Signals and Communication Technology

Mourad Abbas *Editor*

Analysis and Application of Natural Language and Speech Processing

Springer

Signals and Communication Technology

This series is devoted to fundamentals and applications of modern methods of signal processing and cutting-edge communication technologies. The main topics are information and signal theory, acoustical signal processing, image processing and multimedia systems, mobile and wireless communications, and computer and communication networks. Volumes in the series address researchers in academia and industrial R&D departments. The series is application-oriented. The level of presentation of each individual volume, however, depends on the subject and can range from practical to scientific.

Indexing: All books in "Signals and Communication Technology" are indexed by Scopus and zbMATH

For general information about this book series, comments or suggestions, please contact Mary James at mary.james@springer.com or Ramesh Nath Premnath at ramesh.premnath@springer.com.

Mourad Abbas
Editor

Analysis and Application of Natural Language and Speech Processing

Springer

Editor
Mourad Abbas
High Council of Arabic
Algiers, Algeria

ISSN 1860-4862 ISSN 1860-4870 (electronic)
Signals and Communication Technology
ISBN 978-3-031-11034-4 ISBN 978-3-031-11035-1 (eBook)
https://doi.org/10.1007/978-3-031-11035-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The increased access to powerful processors has made possible significant progress in natural language processing (NLP). We find more research in NLP targeting diverse spectrum of major industries that use voice recognition, text-to-speech (TTS) solutions, speech translation, natural language understanding (NLU), and many other applications and techniques related to these areas.

This book presents the latest research related to natural language processing and speech technology and sheds light on the main topics for readers interested in this field. For TTS and automatic speech recognition, it is demonstrated how to explore transfer learning in order to generate speech in other voices from TTS of a specific language (Italian), and to improve speech recognition for non-native English. Language resources are the cornerstone for building high-quality systems; however, some languages, as Arabic, are considered under-resourced compared to English. Thus, a new Arabic linguistic pipeline for NLP is presented to enrich the Arabic language resources and to solve common NLP issues, like word segmentation, POS tagging, and lemmatization. Arabic named entity recognition, a challenging task, has been resolved within this book using transformer-based-CRF model.

In addition, the readers of this book will discover conceptions and solutions for other NLP issues such as language modeling, question answering, dialog systems, and sentence embeddings.

Mourad Abbas

Contents

ITAcotron 2: The Power of Transfer Learning in Expressive TTS Synthesis

Anna Favaro, Licia Sbattella, Roberto Tedesco, and Vincenzo Scotti

Abstract A text-to-speech (TTS) synthesiser has to generate intelligible and natural speech while modelling linguistic and paralinguistic components characterising human voice. In this work, we present ITAcotron 2, an Italian TTS synthesiser able to generate speech in several voices. In its development, we explored the power of transfer learning by iteratively fine-tuning an English Tacotron 2 spectrogram predictor on different Italian data sets. Moreover, we introduced a conditioning strategy to enable ITAcotron 2 to generate new speech in the voice of a variety of speakers. To do so, we examined the zero-shot behaviour of a speaker encoder architecture, previously trained to accomplish a speaker verification task with English speakers, to represent Italian speakers' voiceprints. We asked 70 volunteers to evaluate intelligibility, naturalness, and similarity between synthesised voices and real speech from target speakers. Our model achieved a MOS score of 4.15 in intelligibility, 3.32 in naturalness, and 3.45 in speaker similarity. These results showed the successful adaptation of the refined system to the new language and its ability to synthesise novel speech in the voice of several speakers.

1 Introduction

The development of text-to-speech (TTS) synthesis systems is one of the oldest problems in the natural language processing (NLP) area and has a wide variety of applications [14]. Such systems are designed to output the waveform of a voice

A. Favaro (✉)
Center for Language and Speech Processing (CLSP), Johns Hopkins University, Baltimore, MD, USA
e-mail: afavaro1@jhu.edu

L. Sbattella · R. Tedesco · V. Scotti
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milano, MI, Italy
e-mail: licia.sbattella@polimi.it; roberto.tedesco@polimi.it; vincenzo.scotti@polimi.it

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_1

uttering the input text string. In the last years, the introduction of approaches based on deep learning (DL), and in particular the end-to-end ones [11, 20, 22, 25], led to significant improvements.

Most of the evaluations carried out on these models are performed on languages with many available resources, like English. Thereby, it is hard to tell how good these models are and whether they are general across languages. With this work, we propose to study how these models behave with less-resourced languages, leveraging the transfer learning approach.

In particular, we evaluated the effectiveness of transfer learning on a TTS architecture, experimenting with the English and Italian languages. Thus, we started from the English TTS Tacotron 2 and fine-tuned its training on a collection of Italian corpora. Then, we extended the resulting model, with speaker conditioning; the result was an Italian TTS we named ITAcotron 2.

ITAcotron 2 was evaluated, through human assessment, on intelligibility and naturalness of the synthesised audio clips, as well as on speaker similarity between target and different voices. In the end, we obtained reasonably good results, in line with those of the original model.

The rest of this paper is divided into the following sections: In Sect. 2, we introduce the problem. In Sect. 3, we present some available solutions. In Sect. 4, we detail the aim of the paper and the experimental hypotheses we assumed. In Sect. 5, we present the corpora employed to train and test our model. In Sect. 6, we explain the structure of the synthesis pipeline we are proposing and how we adapted it to Italian from English. In Sect. 7, we describe the experimental approach we followed to assess the model quality. In Sect. 8, we comment on the results of our model. Finally, in Sect. 9, we sum up our work and suggest possible future extensions.

2 Background

Every TTS synthesiser represents an original imitation of the human reading capability, and, to be implemented, it has to cope with the technological and imaginative constraints characterising the period of its creation.

In the mid-1980s, the concomitant developments in NLP and digital signal processing (DSP) techniques broadened the applications of these systems. Their first employment was in screen reading systems for blind people, where a TTS was in charge of reading user interfaces and textual contents (e.g. websites, books, etc.), converting them into speech. Even though the early screen readers (e.g. JAWS[1]) sounded mechanical and robotic, they represented a valuable alternative for blind people to the usual braille reading.

Since the quality of TTS systems has been progressively enhanced, their adoption was later extended to other practical domains such as telecommunications services,

[1] https://www.freedomscientific.com/products/software/jaws.

language education systems, talking books and toys, and video games. In 2004, Yamaha Corporation released its first version of Vocaloid [16]. It is a voice synthesising software product, based on diphone concatenation that allows creating a virtual singer by specifying the text and the melody of a song.

In general, TTS can be conceived as a decoding problem where a highly "compressed" input sequence (text) has to be "decompressed" into audio. However, linguistic units (e.g. phonemes, characters, words) are discrete, whereas speech signals are continuous and longer than textual input sequences; this mismatch causes prediction errors to accumulate rapidly. Besides, the meaning expressed by an utterance is typically undermined by its textual counterpart since the same textual sequence can correspond to several pronunciations and speaking styles.

Nowadays, the hectic development of embodied agent technologies such as embodied conversational agents (ECA) that adopt mimics, gestures, and speech to communicate with the human user makes the modelling of human-computer dialogues a research hotspot. To endow an ECA with a human-like conversational behaviour, a TTS system cannot just synthesise understandable speech at a fast rate. Instead, it needs to account for further speech nuances in order to reproduce elements and peculiarities of human conversations [8].

Thus, to synthesise human-like speech, a TTS has to explicitly or implicitly model many factors that are not attested in the textual input. This requirement is invoked by the presence of paralinguistic components characterising human dialogic exchanges. On the whole, the synthesised speech should express the correct message (intelligibility) while sounding like human speech (naturalness) and conveying the right prosody (expressiveness) [31]. This is what makes the development of high-quality TTS systems a challenging task.

In our daily conversational exchanges, paralinguistic components (the so-called prosody) are exploited as a whole to attribute mental states and an independent mental life to our interlocutors. It follows that, if the ultimate goal is to develop an ECA that can successfully hold a conversation and be mistaken for a human, all of these components should be taken into account within the modelling pipeline [24].

Prosody is the systematic arrangement of various linguistic units into a single utterance or group of utterances, which occurs in the process of human speech production [7]. Its implementation encompasses both segmental and suprasegmental features of speech, and it aims at conveying linguistic and non-linguistic information.

Speech prosody mainly plays the roles of [28]:

- Disambiguating the verbal component of communication (i.e. augmentative prosody) [21]
- Conveying emotions, intentions, and attitudes in communication (i.e. affective prosody) [23]

The lack of explicit control on specific speech traits characterises architectures based on DL. This usually prevents them from reproducing accurately prosodic phenomena, both locally and globally. However, the end-to-end learning approach allows for the introduction of rich conditioning on various prosodic attributes. Thus, besides generating a comprehensible synthetic product, Seq-2-Seq models enable the synthesis of speech in multiple voices, in various styles, and with different emotional nuances. In the following, we present some modern approaches to TTS, which mostly inspired our work.

3 Related Work

Modern, DL-based TTS pipelines are composed of two main blocks: a *spectrogram predictor* and a *vocoder* [14]. These components take care of, respectively, converting a string of characters into a (mel-scaled) spectral representation of the voice signal and converting the spectral representation to an actual waveform. Optionally, input text—apart from normalisation—undergoes phonemisation to present the input to the spectrogram predictor as a sequence of *phonemes* rather than *graphemes*.

Recent end-to-end solutions for spectrogram prediction are built with an *encoder-decoder* architecture [20, 22, 25, 32]. The encoder maps the input sequence to a hidden continuous space, and the decoder takes care of generating, autoregressively, the spectrogram from the hidden representation. To produce the alignment between encoder and decoder, an *attention mechanism* [2] is introduced between these two blocks.

Among the available architectures for spectrogram prediction, *Tacotron* [32], and in particular its advanced version, Tacotron 2 [25], seems to be the most flexible and re-usable.

Many works have been developed to introduce conditioning into Tacotron, obtaining a fine-grained control over different prosodic aspects. The *Global Style Token* (GST) approach enabled control over the speaking style in an unsupervised manner [33]. Another controllable aspect is the speaker voice, introduced through additional *speaker embeddings* extracted through a speaker verification network [13]. Finally, [27] proposed a methodology to control the *prominence* and *boundaries* by automatically deriving prosodic tags to augment the input character sequence. It is also possible to combine multiple techniques into a single conditioned architecture, as shown by Skerry-Ryan et al. [26].

Neural vocoders completed the DL-based TTS pipeline improving consistently the quality of synthesised voice [15, 17, 30, 34]. These vocoders substituted the Griffin-Lim algorithm [9], which was characterised by artefacts and poor audio quality, especially if compared with newer neural approaches. These components, different from the spectrogram predictors, do not strictly depend on the input language—their primary role is to invert a spectral representation into the time domain; thus, they are thought to be *language-agnostic*.

As premised, the available models are primarily trained and evaluated on English corpora, due to data availability. A general solution for data scarcity is to leverage a technique called *transfer learning* [35] that mimics what typically occurs in human learning. In most learning environments, in fact, people do not start from scratch when forming hypotheses and drawing inferences. Rather, knowledge gained from one domain is abstracted and re-used in other domains. The more related a new task is to our previous experience, the easier we can master it.

The lack of sufficiently large data sets makes DL hard to apply and yield satisfying results. The mechanism of transfer learning [29] attempts to cope with this issue by re-using the hidden representations learnt performing a task on a domain similar to the target one. The idea is to leverage the knowledge derived solving one or more source tasks and use it to improve the results in a new target task. Techniques that enable knowledge transfer represent progress towards making machine learning as efficient as human learning.

For our work, we applied a variant of transfer learning called *fine-tuning*, where we used the pre-trained weights of the network as initialisation for the actual training on the new task [35].

4 Aim and Experimental Hypotheses

Whereas recent research mainly focuses on further refining the intelligibility and naturalness of the synthesised product, speech expressiveness requires for its part explicit modelling. The lack of a similar control makes the prosody of the synthesised speech feel like an “average, anonymous voice style”, preventing it from fully displaying the range of prosody variations occurring in human speech.

In this work, we address the modelling of speech expressiveness by presenting a customised TTS architecture, *ITAcotron 2*, which is able not only to generate intelligible and natural Italian speech but also to emulate the voice of a given speaker.

A speech synthesiser of this kind can endow an ECA with a personalised voice, which contributes to increasing human engagement in the ongoing conversation [3, 19]. Furthermore, such a system can have a wider range of useful applications, such as speech-to-speech translation, navigation systems, and fairy tale reciting. It could also allow people who have lost their voices to recover their ability to communicate naturally even though they can’t provide a satisfying amount of previously recorded voice samples.

Note that, in achieving this goal, we distanced ourselves from the potential misuses accompanying the development of this technology. Impersonating someone’s voice without their consent represents a clearer example of such a drift. In developing our system, we stuck to AI Google’s principles,[2] and we hope that future users and developers will act in full compliance with these guidelines.

The experimental hypotheses underlying our research are the following: Firstly, we examined the possibility of performing language adaptation using transfer learning. To do so, we fine-tuned a customised TTS model, previously trained on

[2] https://ai.google/principles.

English data, porting it to the Italian language. Then, we conducted a listening task to collect subjective ratings expressing the intelligibility and naturalness of the synthesised Italian speech.

Secondly, we investigated the feasibility of modelling fine-grained, speaker-dependent characteristics while generating new speech from text. These features crucially contribute to the uniqueness of each utterance for the fact that every speaker is provided with a unique vocal identity. Thus, we represented speakers' *voiceprints* as fixed-dimensional embeddings (i.e. d-vectors) to condition the speech generation with the purpose of synthesising different voices from a single model.

Finally, since we employed a speaker encoder architecture that was previously trained on an English verification task to extract speakers' voiceprints, we implicitly tested the feasibility of performing language adaptation. In fact, if the speaker encoder was able to derive discriminative features for representing English speakers, it should have been able to do the same for Italian speakers. Thus, we designed this experiment to prove whether the speaker encoder was language-agnostic and applicable to a language that differs from the source one (English), without being re-trained or fine-tuned. We did so because we wanted to observe the zero-shot behaviour of this network in the new language (Italian). We apply the same strategy to the neural vocoder, by using a network previously trained with English recordings and not refined on the new language.

5 Corpora

In this work, we considered three different corpora of Italian speech, containing recited utterances. We reported the main statistics about the corpora in Table 1. All clips were re-sampled at 22.050 Hz.

Mozilla Common Voice[3] (MCV) is a publicly available corpus of crowd-sourced audio recordings [1]. Contributors can either donate their voice by reading prompted sentences or validate clips by listening to others' recordings. Clips in this corpus have a native sample rate of 48.000 Hz.

Table 1 Statistics on the considered corpora for the Italian fine-tuning of the spectrogram predictor: Mozilla Common Voice (MCV), VoxForge (VF), and Ortofonico (Ort)

	Time [h]			Clips			Speakers		
Corpus	Train	Validation	Test	Train	Validation	Test	Train	Validation	Test
MCV	79.1	26.5	26.4	50,322	16,774	16,775	5151	3719	3743
VF	13.6	1.8	1.8	7176	913	918	903	584	597
Ort	2.9	0.4	0.3	1436	164	159	20	20	20

[3] https://commonvoice.mozilla.org.

VoxForge[4] (VF) is a multilingual open-source speech database that includes audio clips collected from volunteer speakers. Clips in this corpus have a native sample rate of 16.000 Hz.

Ortofonico (Ort) is a subset of the CLIPS[5] corpus of Italian speech, collected for a project funded by the Italian Ministry of Education, University and Research. Audio recordings come from radio and television programmes, map task dialogues, simulated conversations, and text excerpts read by professional speakers. Clips in this corpus subset have a native sample rate of 22.050 Hz.

Apart from the three presented corpora, we used some clips from a private collection of audiobooks in the human evaluation step. We reported further details in Sect. 7.

6 ITAcotron 2 Synthesis Pipeline

The model we propose is called ITAcotron 2. It is an entire TTS pipeline, complete with speaker conditioning, based on Tacotron 2 [13, 25]. The pipeline is composed of a phonemiser, a speaker encoder (used for the conditioning step), a spectrogram predictor, and a neural vocoder. We reported a scheme of the pipeline in Fig. 1.

The core of the model we are presenting is the spectrogram predictor. We referred to the Tacotron 2 implementation and weights provided by Mozilla[6] [10]. The

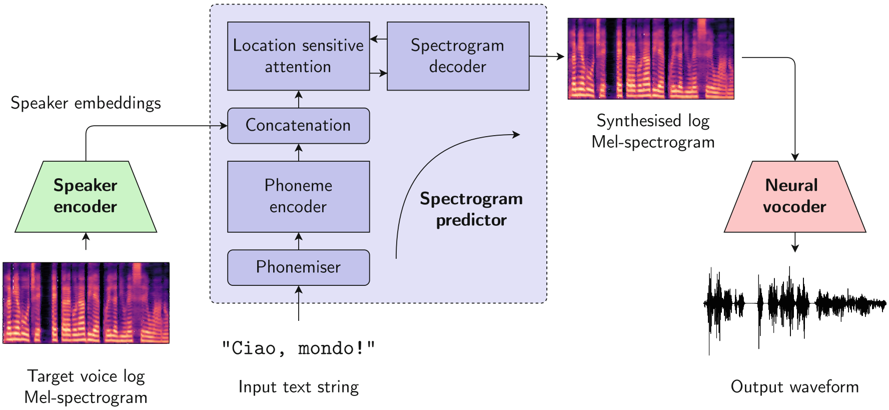


Fig. 1 ITAcotron 2 speech synthesis pipeline

[4] http://www.voxforge.org.

[5] http://www.clips.unina.it.

[6] Repository link, https://github.com/mozilla/TTS; reference commit link, https://github.com/mozilla/TTS/tree/2136433.

model uses a *phoneme encoder* to represent the input sequence to utter and an *autoregressive decoder* to generate the target spectrogram; an intermediate attention mechanism provides the input-output alignment. With respect to the original implementation, we only extended the employed phonemiser[7] to accommodate Italian's accented vowels as additional input characters. Code and pre-trained weights for speaker encoder and vocoder came from the Tacotron 2 same source.

We divided the fine-tuning process of the spectrogram predictor into two steps. In this way, we iteratively improved the output quality.

The former used only the data coming from the MCV corpus, which constituted the majority of the available data. Due to the low quality of the input audio recordings, we leveraged this step mostly to drive the network's weights towards the target language. The noisy and sometimes poorly uttered clips of this corpus resulted in synthesised clips of poor quality, which sometimes were impossible to understand. This fine-tuning was performed for 52.271 update steps (identified through validation) on mini-batches containing 64 clips each; other hyper-parameters were left unchanged from the reference implementation.

The latter fine-tuning leveraged both VF and Ort corpora. Audio clips in these corpora were of a noticeable higher quality than those of MCV in terms of audio clearness and speaker articulation. As a result, the outputs of this final stage had significantly less background noise, and the content was highly intelligible. We performed this second fine-tuning for 42.366 update steps (identified through validation) on mini-batches containing 42 clips each; other hyper-parameters were left unchanged from the reference implementation.

To achieve speaker conditioning, we concatenated the encoder representation of the spectrogram predictor with a *speaker embedding*. These embeddings were extracted from a speaker verification model [4], similar to that of the reference work by [13]. For the vocoder, instead, we adopted the more recent *full-band MelGAN* (FB-MelGAN) vocoder [34].

Notice that while we fine-tuned the spectrogram synthesis network, we did not apply the same process to the speaker embedding and neural vocoder networks. We did so because we wanted to observe the zero-shot behaviour of these networks in the new language. In this way, we could assess whether the two models are language-agnostic.

7 Evaluation Approach

Similar to [13], we divided the evaluation process of our fine-tuned model into two listening experiments:

[7] https://pypi.org/project/phonemizer/.

- Evaluation of *intelligibility and naturalness* (I&N) of the speaker-conditioned synthesised samples
- Evaluation of *speaker similarity* (SS) of the speaker-conditioned synthesised samples

For both experiments, we asked subjects to rate different aspects—in a 1 to 5 scale, with 0.5 increments [12]—of several audio clips. We divided the 70 participants into 20 experimental groups for both listening tasks. We prompted participants of each group with the same clips.

In the I&N experiment, we assigned each group with 4 clip pairs, for a total of 160 among all groups. Each clip pair was composed of a real clip (ground truth) coming from one of the corpora (including an additional private corpus of audiobooks) and a synthetic clip generated in the voice of the ground truth, but with different speech content (i.e. the same voice uttered a different sentence). At this step, we asked subjects to rate the intelligibility and naturalness of each clip, separately. Clips were presented in a random order (to avoid biases) and were rated right after listening.

In the SS experiment, we assigned each group with 16 clips, split into 4 subsets, for a total of 160 among all groups. We divided the SS experiment into three tasks. Each task was composed of a synthetic clip and three real clips. Subjects compared the synthetic clip to each of the other three real clips:

1. A real clip containing an utterance in the voice of the same speaker of the synthetic utterance (the *same-speaker* comparison task)
2. A real clip containing an utterance in the voice of a different speaker having the same gender of the speaker of the synthetic utterance (the *same-gender* comparison task)
3. A real clip containing an utterance in the voice of a different speaker having different gender of the speaker of the synthetic utterance (the *different-gender* comparison task)

At this step, we asked subjects to rate how similar the synthetic voice was to the one we paired it with (knowing that the fixed clip was synthetic and the other three real). Real clips were presented in a random order (to avoid biases), and subjects rated the similarity right after listening to a synthetic-real pair.

8 Results

In this section, we report results on the two experiments described in the previous section, by providing both quantitative and qualitative analyses.

We report the mean opinion score (MOS) of each task in Table 2. The overall scores were satisfying and reflected the intentions and the expectations underlying this research.

Table 2 Results of the listening tasks. MOS values are reported as $\mu \pm \sigma$. All values are computed with a support of 280 samples

Task	(Comparison) Task	Model	MOS	95% confidence interval
Intelligibility and naturalness	Intelligibility	ITAcotron 2	4.15 ± 0.78	$[4.07,\ 4.23]$
		Ground truth	4.43 ± 0.74	$[4.36,\ 4.50]$
	Naturalness	ITAcotron 2	3.32 ± 0.97	$[3.22,\ 3.41]$
		Ground truth	4.28 ± 0.86	$[4.20,\ 4.37]$
Speaker similarity	Same speaker	ITAcotron 2	3.45 ± 1.07	$[3.34,\ 3.56]$
	Same gender	ITAcotron 2	2.78 ± 1.01	$[2.68,\ 2.89]$
	Different gender	ITAcotron 2	1.99 ± 1.08	$[1.91,\ 2.08]$

Concerning the I&N evaluation, the first thing that jumps to the eye is the high intelligibility score, very close to real clips. This high score provides clear evidence of how easy it was to understand the linguistic content of the synthetic clips. The naturalness score, however, is lower than that of intelligibility, meaning that it is still possible to distinguish between real and synthetic clips.

Concerning the SS evaluation, instead, the thing that jumps to the eye is the progressive drop in the MOS value. This reduction is precisely the expected behaviour [18]—changing the speaker should lead to lower similarity, especially when the two speakers have different gender. Thus, the MOS obtained for the same-speaker task was the best one, with a promising absolute value; in other words, the system was able to provide a good imitation of the speaker's voice. Then, changing speaker, the MOS dropped, meaning that the synthetic voice was able to express the "personality" of the speaker it was imitating, and so it was clearly distinguishable from other voices). Finally, as expected, a further drop was observed by the different-gender similarity evaluations.

The figures we obtained are close to those obtained by the reference work [13] on similar tasks, for English. However, we choose not to report a direct comparison against the work mentioned above as it focuses on English and the tasks are not perfectly comparable with ours. Nevertheless, obtaining similar scores is a hint that our approach seems sound. More detailed results on the two experiments we conducted are presented in the following.

8.1 Speech Intelligibility and Naturalness

This experiment was meant to evaluate the language adaptation hypothesis by assessing the degree of intelligibility and naturalness exhibited by the synthesised speech. MOS evaluation distributions, for real and synthesised speech, are reported as histograms in Fig. 2.

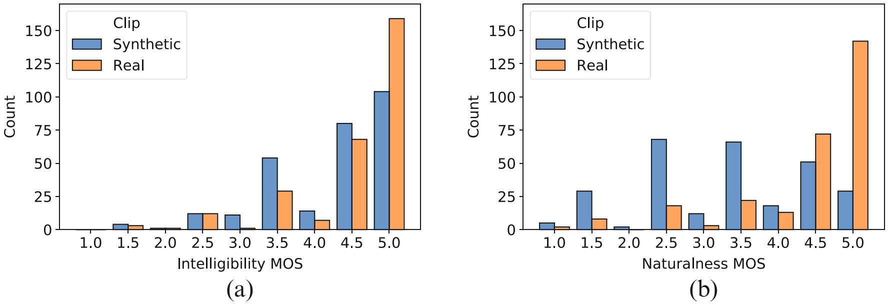


Fig. 2 MOS distributions of real and synthetic clips. (**a**) Intelligibility MOS distributions. (**b**) Naturalness MOS distributions

Table 3 Results of Welch's t-test with degrees of freedom (DoF) computed as $n_1 - n_2 - 2$, where n_1 and n_2 are the sample sizes of the first and second sample. The t-value is the value used to produce the probability value (p-value) based on Student's t-distribution. All scores are computed with 558 DoF

	Sample			
Task	First	Second	t-value	p-value
Intelligibility	ITAcotron 2	Ground truth	4.375	1.448e–6
Naturalness	ITAcotron 2	Ground truth	12.414	2.237e–31
Speaker similarity	Same speaker	Same gender	7.606	1.208e–13
	Same speaker	Different gender	16.054	5.812e–48
	Same gender	Different gender	8.962	4.820e–18

Table 2 reports sample size, mean, standard deviation, and 95% confidence interval computed with empirical bootstrap [5] for real and synthetic audio MOS distributions. Results from Welch's unequal variance t-test for intelligibility are reported in Table 3.

Ground-truth recordings obtained a higher intelligibility MOS ($\mu = 4.43$, $\sigma = 0.74$) than did audio clips synthesised by our model ($\mu = 4.15, \sigma = 0.78$), $t(558) = 4.37$, $p < 0.05$. Our proposed model achieved 4.1 intelligibility MOS compared to 4.34 of the ground truth. This confirmed the system ability to synthesise speech with highly intelligible content.

The speech content of the data sets used to train our synthesiser, especially in the second fine-tuning, was in most of the cases clearly and smoothly comprehensible. This could have influenced positively the output intelligibility. Moreover, using a significant amount of hours and speakers in training could have lead the model both to improve its generalisation ability and to distinguish easily useless from useful spectral information at prediction time. These aspects could have jointly increased the intelligibility of the synthesis.

In exploratory listening sessions, we noticed that the model learnt to smoothly generate out of vocabulary words, long input texts, and complex syntactic structures such as long-distance dependencies and topicalised sentences. Indeed, and surprisingly, there were cases in which its generative performance seemed to be enhanced by the presence of such complexities. This could be originated by the adoption of a double decoder that helped to reduce attention alignment problems at inference time.

However, we also observed that the model was not able to synthesise interrogative direct sentences from text. This may be caused by the fact that interrogative sentences were not present in any training sets. Otherwise, the model could have learnt to reproduce properly the suprasegmental prosody features which distinguish an assertion from a direct question.

With respect to naturalness, the MOS distributions for real and synthesised speech are reported as histograms in Fig. 2b. In addition, to visualise clearly the differences between these two score distributions, two paired plots are presented in Fig. 3b.

Sample size, mean, standard deviation, and confidence interval computed with empirical bootstrap for both real and synthetic distributions are summarised in Table 2. Results from Welch's unequal variance *t*-test are reported in Table 3.

Ground-truth recordings obtained a higher naturalness MOS ($\mu = 4.28$, $\sigma = 0.86$) than did audios synthesised by our model ($\mu = 3.32$, $\sigma = 0.97$), $t(558) = 12.41$, $p < 0.05$. Our proposed model achieved 3.32 naturalness MOS compared to 4.18 of the ground truth. This might have been due to an evident drawback of the ICV data set which presents a high level of background noise that the synthesiser had learnt to reproduce.

In exploratory listening sessions, we noticed that the naturalness of the synthesised voice mainly varied depending on which speaker embedding was adopted to condition the generative process. Moreover, the model learnt to pause naturally when punctuation marks, such as comma or full stop, were inserted in the input text. This was probably the consequence of not having removed punctuation in speech transcripts when training our system. Thus, the model seems to have learnt some prosodic aspects connected to the presence of punctuation marks.

In analysing the experimental results for both intelligibility and naturalness, we were also interested in discovering whether a linear correlation existed between the amount of training data associated with the voices used to synthesise the experimental stimuli (i.e. independent variable) and the quality of the speech synthesised using those voices (i.e. dependent variable). We expected that the more the system was exposed to speech data belonging to a given voice, the more it would have been able to synthesise high-quality sentences in that voice.

About intelligibility, for each of the 62 voices used in the experiment, we computed the total amount of training sentences associated with each of them and its average intelligibility MOS from the scores it received in the listening test. Then, we represented each voice as a point in a Cartesian plane, where x axis stood for its average intelligibility MOS and y axis stood for the amount of time (in minutes) that voice was seen during training.

Pearson product-moment correlation and Spearman's rank correlation coefficient were computed to detect whether a correlation could be identified between the amount of training data each voice is assigned to and the intelligibility MOS each

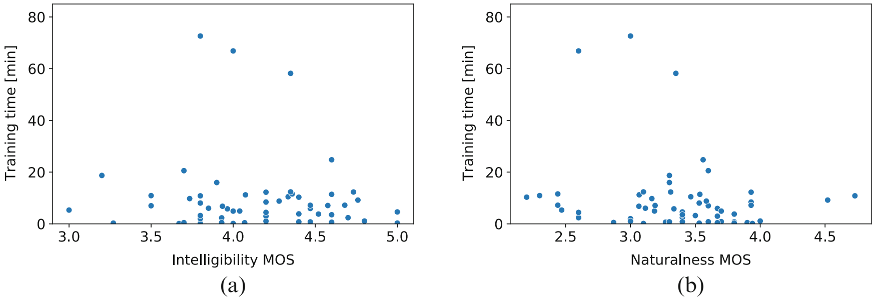


Fig. 3 I&N scores vs duration of synthetic clips. (**a**) Voice-wise average intelligibility MOS vs. duration time in training. (**b**) Voice-wise average naturalness MOS vs. duration time in training

voice received in the listening test. The p-value for the Pearson correlation between these two variables was above the significance level of 0.05, which means that the correlation coefficient was not significant ($r = -0.07, n = 62, p = 0.55$). The same occurred for Spearman's ρ correlation coefficient ($r_s = 0.03, n = 62, p = 0.83$).

The scatterplot of Fig. 3a summarises the results. Overall, no correlation was found between the amount of training data associated with a given voice and the intelligibility MOS it received in the subjective listening test.

The same correlation was investigated for speech naturalness. We represented each of the 62 voice in a Cartesian plane, where x axis stood for its average naturalness MOS and y axis stood for the amount of time (in minutes) that voice was seen in training.

We derived Pearson product-moment correlation and Spearman's rank correlation coefficient to detect whether there existed a correlation between the amount of training data each voice is presented with and the naturalness MOS each voice received when evaluated. With respect to Person correlation, the p-value between these two variables was 0.16 ($r = -0.18$, $n = 62$). Since it was greater than the significance level of 0.05, there was inconclusive evidence about the significance of the association between these two variables. The same occurred for Spearman's ρ correlation coefficient ($r_s = -0.21, n = 62, p = 0.10$).

The scatterplot in Fig. 3b summarises these results. Overall, no correlation was found between the amount of training data associated with a given voice and the naturalness MOS it received in the subjective listening test.

8.2 Speaker Similarity

The second experiment was meant to evaluate the effectiveness of the strategy we used to condition our TTS generative pipeline. Namely, we wanted to assess whether the model developed the capability of disentangling speaker-dependent characteristics from linguistic component when synthesising new speech.

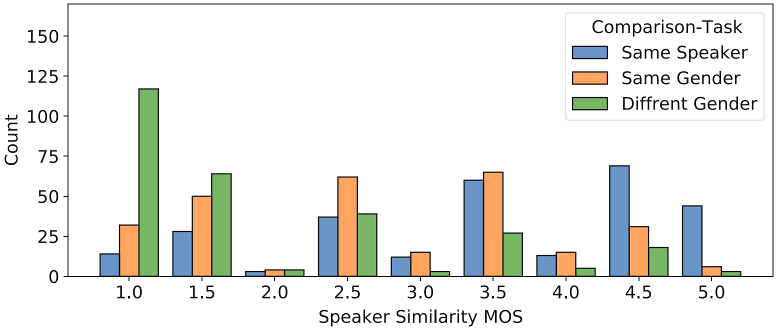


Fig. 4 SS comparison task similarity MOS distributions

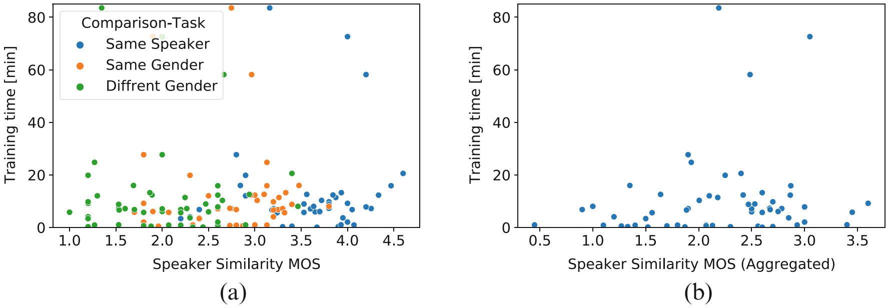


Fig. 5 Speaker similarity vs duration. (**a**) Separate. (**b**) Combined

Moreover, we were also interested in verifying whether the speaker encoder adopted to extract speaker discriminative characteristics was language-agnostic and thus applicable to Italian speech samples.

Figure 4 reports the MOS distributions of same-speaker, same-gender, and different-gender similarity judgements as histograms and box plots, respectively. In addition, to clearly visualise the differences between these three score distributions, in Fig. 5, two paired plots are presented.

We report mean, standard deviation, and confidence interval computed with empirical bootstrap for SS comparison task MOS distributions in Table 2.

Results from three Welch's unequal variance t-tests for speaker similarity are reported in Table 3. Same-speaker similarity judgements obtained a higher MOS ($\mu = 3.45$, $\sigma = 1.07$) than same-gender similarity judgements ($\mu = 2.78$, $\sigma = 1.01$), $t(558) = 7.60$, $p < 0.05$. Same-gender similarity judgements obtained a higher MOS ($\mu = 2.78$, $\sigma = 1.01$) than different-gender similarity judgements ($\mu = 1.99$, $\sigma = 1.08$), $t(558) = 8.96$, $p < 0.05$. Same-speaker similarity judgements obtained a higher MOS ($\mu = 3.45$, $\sigma = 1.07$) than different-gender similarity judgements ($\mu = 1.99$, $\sigma = 1.08$), $t(558) = 16.05$, $p < 0.05$.

On one hand, same-speaker similarity average score amounts to 3.45, which indicates a reasonable resemblance between the voices of audios synthesised in a given voice and their real counterparts. On the other hand, same-gender similarity MOS is 2.78, while different-gender similarity MOS is 1.99. It means that the raters recognised correctly the extent in which voices of speakers of the same or different genders differed from each other.

These results confirmed the overall model ability to decouple speech content from speaker-dependent characteristics and to learn high-fidelity speaker representations that can be exploited to generate speech in a desired voice. In general, our system was able to transfer effectively speakers' gender for all the 57 voices used in the experiment (Table 2).

In exploratory listening sessions, we noticed mismatches on regional accent, between synthesised and target voices. This could have influenced negatively the similarity judgements since participants were not told how to judge accents, so they could have rated poorly because of accent mismatches rather than because of low model quality. Accent mismatches could have been caused by the use of a speaker encoder trained only on English-accented speech. Since English and Italian do not match in terms of accent, this could have prevented the system from properly and systematically transferring accents from target to synthesised audios.

However, some raters reported that the accent of the synthesised speech exhibited a clear resemblance with the accent of the original recording from the same speaker. This effect was larger on IVF data set which contains more marked regional accents. It follows that, to better exert a control on the synthesised rendition, a further refinement is required to decouple speaker individual characteristics from prosody and linguistic features.

As for the previous experiment, in analysing the results, we were interested in verifying whether a correlation existed between the amount of training recordings associated with the voices used to synthesised the experimental stimuli (i.e. independent variable) and the degree of resemblance (i.e. dependent variable) that particular voice exhibited with its real counterpart. We expected that the more the system had been exposed to speech data belonging to a given voice, the more it would have been able to transfer speaker-dependent characteristics at inference time.

In assessing whether such correlation occurred, we adopted both a disaggregated and an aggregated approach to represent the similarity MOS score associated with each voice.

In the first approach, we intended the similarity score as three distinct scores. Namely, for each of the 57 voices used in the experiment, we computed comparison task average similarity MOS, separately. Additionally, we computed the amount of training data (in minutes) associated with each of these voices. It follows that each voice used in the experiment was represented in a Cartesian plane, where the x axis stood for either its comparison task average similarity score and y axis stood for the amount of time that voice was seen in training.

We derived the Pearson product-moment correlation and Spearman's rank correlation coefficient between comparison task average scores and the corresponding amount of recordings associated with each voice in training. The p-value for the Pearson correlation between the same-speaker similarity MOS and the amount of training data was above the significance level of 0.05, which indicated that the correlation coefficient was not significant ($r = 0.21$, $n = 57$, $p = 0.11$). The same occurred for the Pearson correlation computed between same-gender similarity MOS and the amount of training data synthesised ($r = -0.01$, $n = 57$, $p = 0.92$) and between different-gender similarity MOS and the amount of training data synthesised ($r = 0.03$, $n = 57$, $p = 0.79$).

Concerning Spearman's ρ correlation coefficient, the p-value between SS similarity MOS and the amount of training recordings is about 0.40, which indicates that there was a moderate positive correlation [6] between these two variables ($p = 0.002$, $n = 57$). Differently, the p-value for Spearman's correlation between same-gender similarity MOS and the amount of training data synthesised was above the significance level of 0.05, which indicates that the correlation coefficient was not significant ($r_s = 0.22$, $n = 56$, $p = 0.08$). The same occurred between the different-gender similarity MOS and the amount of training data ($r_s = 0.14$, $n = 56$, $p = 0.30$).

A scatterplot in Fig. 5a summarises these findings. It highlights that in general no correlation exists between the amount of training data associated with a given voice and the similarity MOS evaluations it received in the subjective listening test. However, a moderate correlation can be detected between same-speaker similarity MOS and the amount of training data (green dots on the right).

In verifying the existence of such correlation, the other approach we adopted to represent the similarity MOS was to assign to each voice a single score. This score was derived by subtracting from the same-speaker similarity MOS the average between same-gender and different-gender similarity MOS. Thus, each voice used in the experiment was represented in a Cartesian plane, where the x axis stood for its aggregated similarity MOS score and y axis stood for the amount of time the spectrogram predictor processed that voice during training.

We computed the Pearson product-moment correlation and Spearman's rank correlation coefficient between the aggregated similarity MOS and the amount of time each voice was seen in training. The p-value for the Pearson correlation was above the significance level of 0.05, which indicates that the correlation coefficient was not significant ($r = 0.15$, $n = 56$, $p = 0.23$). The same occurred for Spearman's ρ correlation ($r_s = 0.21$, $n = 56$, $p = 0.10$).

A scatterplot in Fig. 5b summarises these results. Overall, no correlation was derived between the amount of training data associated with the speakers used to construct the experimental stimuli and their similarity aggregated MOS.

Since we didn't find significant correlations between intelligibility, naturalness, and speaker similarity evaluations and the amount of training data, we concluded that the model acquired a discrete ability to generalise to speakers for which a low amount of training recordings was provided or even to speakers unseen during training.

9 Conclusion and Future Work

In this paper, we showed our approach to adapt a speech synthesis pipeline from English to Italian. The procedure was language-agnostic, but spectrogram prediction network required fine-tuning data in the target language. To show how some pipeline components can be used without language adaptation, we also introduced a speaker embedding network (to achieve speaker conditioning) and a neural vocoder.

Opinion scores from a human evaluation session showed that the adaptation was successful in terms of intelligibility and naturalness. Concerning speaker conditioning, the result was not as sharp as for the first evaluation; yet, we obtained a satisfying similarity score, matching that of the reference model.

In general, the main issue arising in modelling prosody features implicated in conveying linguistic and non-linguistic information in speech is that they are fully entangled. However, some of these characteristics are speaker-dependent, such as accent and idiolect, while others are speaker-independent such as prosodic phrasing, lexical stress, and tune variation. Thus, if on one side controlling a given prosody phenomenon using a unique latent embedding space would allow a complete control over all linguistic and non-linguistic components, disentangling speaker-dependent and speaker-independent characteristics enables simplified models and better decoupling, from a human perspective, of the controlled aspects.

Traditionally, prosody modelling relied on labelling prosodic phenomena and developing rule-based systems or statistical models from speech data. These strategies allow a high control on speech products, but they require to derive hand-crafted features, which is difficult and time-consuming in the presence of large data sets. In contrast, end-to-end neural TTS systems permit to generate high-fidelity speech with a simplified pipeline and to learn prosody as an inherent part of the model. Even though these unsupervised methods are extensively used, they still miss exerting an accurate and clear control over the output prosody.

Thus, future research will focus on different prosody phenomena, to identify strategies to model micro and macro prosody patterns. To do so, a possibility will be to leverage independent representations, in the form of GST or latent vectors from a variational auto-encoder (VAE), for each of the speech traits of interest. In addition, a multi-head attention mechanism can increase the system parallelisation capability and help to cope with the hardness arising with RNN application in modelling human speech long-distance dependencies.

On the whole, to improve the model ability to regulate its generation in accordance to the mental and emotional status of its interlocutor, we could augment the input with a multi-modal stream of information, encoding the features of the previous conversational turn. For instance, the user's prompt could be represented by feeding the TTS with a visual input encoding the user's facial expression, an acoustic input encoding prosodic information characterising her/his speech, and a linguistic embedding encoding information related to the meaning of her/his words. The TTS should then be trained to align its generation in terms of prosody and linguistic content, in accordance with the previous conversational turn—thing that we, as humans, do in every conversational exchange.

Acknowledgments This work was partially supported by the European Union's Horizon 2020 project *WorkingAge* (grant agreement no. 826232).

Appendix

The source code developed during this project is available at the following link: https://github.com/vincenzo-scotti/ITAcotron_2. Inside the repository, we also provide the links to download the weights of the fine-tuned model ITAcotron 2, for Italian speech synthesis. We remind that the original source code we forked, and the weights of the speaker encoder and neural vocoder, was taken from the reference open-source project developed by Mozilla[8]

References

1. Ardila, R., Branson, M., Davis, K., Kohler, M., Meyer, J., Henretty, M., Morais, R., Saunders, L., Tyers, F.M., Weber, G.: Common voice: A massively-multilingual speech corpus. In: Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11–16, 2020, pages 4218–4222. European Language Resources Association (2020)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings (2015)
3. Cassell, J.: Embodied conversational agents: Representation and intelligence in user interfaces. AI Mag. **22**(4), 67–84 (2001)
4. Chung, J.S., Huh, J., Mun, S., Lee, M., Heo, H.-S., Choe, S., Ham, C., Jung, S., Lee, B.-J., Han, I.: In defence of metric learning for speaker recognition. In: Meng, H., Xu, B., Zheng, T.F. (eds.) Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25–29 October 2020, pages 2977–2981. ISCA (2020)
5. Dekking, F.M., Kraaikamp, C., Lopuhaä, H.P., Meester, L.E.: A Modern Introduction to Probability and Statistics: Understanding why and how. Springer Science & Business Media, Berlin (2005)
6. Fowler, J., Cohen, L., Jarvis, P.: Practical Statistics for Field Biology. Wiley, Hoboken (2013)
7. Fujisaki, H.: Prosody, models, and spontaneous speech. In: Sagisaka, Y., Campbell, N., Higuchi, N. (eds.) Computing Prosody, Computational Models for Processing Spontaneous Speech, pp. 27–42. Springer, Berlin (1997)

[8] Repository link, https://github.com/mozilla/TTS; reference commit link, https://github.com/mozilla/TTS/tree/2136433.

8. Gilmartin, E., Collery, M., Su, K., Huang, Y., Elias, C., Cowan, B.R., Campbell, N.. Social talk: making conversation with people and machine. In Chaminade, T., Nguyen, N., Ochs, M., Lefèvre, F. (eds.) Proceedings of the 1st ACM SIGCHI International Workshop on Investigating Social Interactions with Artificial Agents, ISIAA@ICMI 2017, Glasgow, United Kingdom, November 13, 2017, pp. 31–32. ACM, New Yrok (2017)
9. Griffin, D.W., Lim, J.S.: Signal estimation from modified short-time fourier transform. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '83, Boston, Massachusetts, USA, April 14–16, 1983, pp. 804–807. IEEE, Piscataway (1983)
10. Gölge, E.: Solving Attention Problems of TTS Models with Double Decoder Consistency (2020)
11. Hsu, P.-C., Wang, C.-H., Liu, A.T., Lee, H.-Y.: Towards robust neural vocoding for speech generation: A survey. CoRR **abs/1912.02461** (2019)
12. ITU-T Recommendation: P.910: Subjective video quality assessment methods for multimedia applications (1999)
13. Jia, Y., Zhang, Y., Weiss, R.J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Lopez-Moreno, I., Wu, Y.: Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada, pp. 4485–4495 (2018)
14. Jurafsky, D., Martin, J.H.: Speech and Language Processing, 2nd edn. Prentice-Hall, Hoboken (2009)
15. Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., Kavukcuoglu, K.: Efficient neural audio synthesis. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018. Proceedings of Machine Learning Research, , vol. 80, pp. 2415–2424. PMLR (2018)
16. Kenmochi, H.: Vocaloid and Hatsune Miku phenomenon in Japan. In: Interdisciplinary Workshop on Singing Voice (2010)
17. Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W.Z., Sotelo, J., de Brébisson,A., Bengio, Y., Courville, A.C.: Melgan: Generative adversarial networks for conditional waveform synthesis. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, pp. 14881–14892 (2019)
18. Leung, Y., Oates, J., Chan, S.P.: Voice, articulation, and prosody contribute to listener perceptions of speaker gender: A systematic review and meta-analysis. J. Speech Language Hearing Res. **61**(2), 266–297 (2018)
19. Moridis, C.N., Economides, A.A.: Affective learning: Empathetic agents with emotional facial and tone of voice expressions. IEEE Trans. Affect. Comput. **3**(3), 260–272 (2012)
20. Ping, W., Peng, K., Gibiansky, A., Arik, S.Ö., Kannan, A., Narang, S., Raiman, J., Miller, J.: Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, April 30–May 3, 2018, Conference Track Proceedings. OpenReview.net (2018)
21. Prieto, P., Borràs-Comes, J., Roseano, P.: Interactive Atlas of Romance Intonation (2010)
22. Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., Liu, T.-Y.: Fastspeech: Fast, robust and controllable text to speech. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, pp. 3165–3174 (2019)
23. Schuller, B., Batliner, A.: Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing. Wiley, Hoboken (2013)
24. Schuller, D., Schuller, B.W.: The age of artificial emotional intelligence. Computer **51**(9), 38–46 (2018)

25. Shen, J., Pang, R., Weiss, R.J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R.J., Saurous, R.A., Agiomyrgiannakis, Y., Wu, Y.: Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, April 15–20, 2018, pp. 4779–4783. IEEE, Piscataway (2018)
26. Skerry-Ryan, R.J., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R.J., Clark, R., Saurous, R.A.: Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, July 10–15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 4700–4709. PMLR (2018)
27. Suni, A., Kakouros, S., Vainio, M., Simko, J.: Prosodic prominence and boundaries in sequence-to-sequence speech synthesis. CoRR **abs/2006.15967** (2020)
28. Taylor, P.: Text-to-Speech Synthesis. Cambridge University Press, Cambridge (2009)
29. Torrey, L., Shavlik, J.: Transfer learning. In: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, pp. 242–264. IGI Global, Pennsylvania (2010)
30. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.W., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. In: The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, 13–15 September 2016, p. 125. ISCA (2016)
31. Wang, Y., Skerry-Ryan, R.J., Stanton, D., Wu, D., Weiss, R.J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q.V., Agiomyrgiannakis, Y., Clark, R., Saurous, R.A.: Tacotron: A fully end-to-end text-to-speech synthesis model. CoRR **abs/1703.10135** (2017)
32. Wang, Y., Skerry-Ryan, R.J., Stanton, D., Wu, Y., Weiss, R.J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q.V., Agiomyrgiannakis, Y., Clark, R., Saurous, R.A.: Tacotron: Towards end-to-end speech synthesis. In: Lacerda, F. (ed.) Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, August 20–24, 2017, pp. 4006–4010. ISCA (2017)
33. Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R.J., Battenberg, E., Shor, J., Xiao, Y., Jia, Y., Ren, F., Saurous, R.A.: Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, July 10–15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 5167–5176. PMLR (2018)
34. Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., Xie, L.: Multi-band melgan: Faster waveform generation for high-quality text-to-speech. In: IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, January 19–22, 2021, pp. 492–498. IEEE, Piscataway (2021)
35. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13 2014, Montreal, Quebec, pp. 3320–3328 (2014)

Improving Automatic Speech Recognition for Non-native English with Transfer Learning and Language Model Decoding

Peter Sullivan, Toshiko Shibano, and Muhammad Abdul-Mageed

Abstract ASR systems designed for native English (L1) usually underperform on non-native English (L2). To address this performance gap, (1) we extend our previous work to investigate fine-tuning of a pre-trained wav2vec 2.0 model (Baevski et al. (wav2vec 2.0: A framework for self-supervised learning of speech representations (2020). Preprint arXiv:200611477), Xu et al. (Self-training and pre-training are complementary for speech recognition. In: ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3030–3034 (2021))) under a rich set of L1 and L2 training conditions. We further (2) incorporate language model decoding in the ASR system, along with the fine-tuning method. Quantifying gains acquired from each of these two approaches separately and an error analysis allows us to identify different sources of improvement within our models. We find that while the large self-trained wav2vec 2.0 may be internalizing sufficient decoding knowledge for clean L1 speech (Xu et al. (Self-training and pre-training are complementary for speech recognition. In: ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3030–3034 (2021))), this does not hold for L2 speech and accounts for the utility of employing language model decoding on L2 data.

1 Introduction

Although non-native English speakers (L2) outnumber native English speakers (L1) [7], major challenges contribute to a gap between performance of automatic speech recognition (ASR) systems on L2 speech. This is mainly due to influence of L1 pronunciation on the learned language and lack of annotated L2 speech data on which ASR systems can be trained [42, 50]. To meet these challenges, previous

P. Sullivan (✉) · T. Shibano · M. Abdul-Mageed
The University of British Columbia, Columbia, BC, Canada
e-mail: prsull@student.ubc.ca; tshibano@student.ubc.ca; muhammad.mageed@ubc.ca

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_2

work has generally followed two distinct approaches. The first is to make L2 speech representations more closely match those of L1 speech [42]. The second approach leverages L2 speech data to improve model robustness. Due to L2 data scarcity, this second approach necessitates employment of transfer learning or domain adaptation [45, 47].

State-of-the-art ASR models based on self-supervised pre-training such as wav2vec [44] and wav2vec 2.0 [2][1] offer a tantalizing starting point for applying the transfer learning approach we list above, especially due to their strong performance of self-trained wav2vec 2.0 models on ASR in low-resource settings even without a language model [56]. However, challenges remain in identifying how best to apply models such as wav2vec 2.0 in L2 fine-tuning scenarios. In spite of this advantage of a fine-tuned model, it is not clear whether the knowledge it acquires is orthogonal to that of a language model especially on L2 speech. Hence, we are interested in *investigating the practical sufficiency of fine-tuned models on their own and the extent to which they may benefit from external language model decoding on both L1 and L2 speeches*. As such, our main objective in the current work is to investigate a rich set of conditions under which we can fine-tune ASR models for optimal L2 performance and the utility of integrating language model decoding along with fine-tuning in an overall ASR model. Concretely, we pursue this primary objective through the following sub-objectives:

1. Evaluate fine-tuning and language model decoding strategies for adapting pre-trained L1 English ASR models to L2 English.
2. Explore the impact of non-native (L2) accents on performance of these ASR models fine-tuned under various conditions, comparing *multi-accent* training to *single-accent* training.
3. Quantify the impact of L2 fine-tuning on model performance for L1 English speech recognition.
4. Analyze error categories associated with fine-tuning, as well as language model decoding.

Our investigation of the role of language model decoding in L2 ASR performance extends our previous work [46]. We also better contextualize the magnitude of impact of fine-tuned only vs. fine-tuning+LM decoding on the downstream tasks for both L1 and L2 speeches. The rest of the paper is organized as follows: Sect. 2 is an overview of related work. We introduce our methods in Sect. 3. We describe our data in Sect. 4, and Sect. 5 is about our experiments and results. We conclude in Sect. 7.

[1] Although sometimes referred to as “unsupervised,” these models employ a self-supervised objective.

2 Related Work

Because of the difficulty in linguistically annotating corpora for hidden Markov model (HMM)-based ASR [12], researchers have broadly embraced end-to-end (E2E) deep learning architectures either based on Connectionist Temporal Classification (CTC) [12, 13], Attention [4, 5, 14], or hybrids of the two [54, 55]. Recent efforts inspired by work such as BERT [9] have improved on these purely supervised learning baselines through self-supervised pre-training [1, 2, 44] and self-training [56]. These self-supervised wav2vec models represent one line of research in speech representation. Other works include models similar to wav2vec that also use a contrastive loss [37], models using an autoregressive loss function [6, 28], as well as models using a masked language model closer to the original BERT [29].

With these efforts, ASR technologies for native languages have evolved significantly. However, we still observe problems in many applications. In particular, several researchers have emphasized how performance of ASR models drops when the input speech is from non-native speakers whose native languages are different from the models' target languages [31, 41, 42, 52, 53]. For systems developed for English ASR, this can be a real issue due to the large populations of English language speakers who are non-native [7]. In line with this argument, Ping [41] points out the necessity to improve speech recognition technology for L2 speakers given that many people speak more than one language for economic and social reasons. It is hoped that continued efforts aiming at improving ASR for non-native speakers will eventually lead to improved results for many as voice recognition technology becomes increasingly pervasive in our daily lives [41].

There are two distinct approaches to improve current ASR performance on L2 speech: (1) accent conversion as an extension to the active area of research of voice conversion and (2) incorporation of L2 speech data, which is often limited in quantity and quality, during the model training process. The first approach takes inspiration from voice conversion, but instead of focusing on modifying the pitch, it modifies the pronunciation to reduce accents. Additionally, voice conversion models aim to generate results that are speaker-dependent, while accent conversion models deal with generalizing accents from a group of speakers, hence being speaker-independent. With this approach, the resulting model can be used as a pre-processing step to remove accents in the data prior to feeding these data into an ASR model. Bearman et al. [3] adopt this approach but focus on L1 English accents, while Radzikowski et al. [42] work on L2 English accents with speakers' L1 being Japanese. Liu et al. [30] took a step further and turned Hindi-accented English to native American English without utilizing native utterances.

The second approach often employs methods such as domain adversarial training and transfer learning in order to utilize as much available accented speech data as possible. Domain adversarial training (DAT) is a popular approach as it encourages models to learn accent-invariant features [19, 21, 47]. Transfer learning is another popular approach in L2 speech recognition, as it possibly allows a model to gain knowledge from both the base task and the new task, even when the new task has limited data [8, 34, 45]. In the Accented English Speech Recognition Challenge 2020 (AESRC2020), many teams utilize transfer learning to tackle the L2 accent recognition task [45]. In a recent work, Das et al. [8] combine both DAT and transfer learning to achieve robust accented speech recognition performance.

One method that is common in ASR systems is language model decoding, which re-weights output probabilities to account for greater likelihoods of words occurring in the language. Language models such as KenLM [17] give probabilities of tokens occurring in a sequence and thus represent corpus-level statistic of language. Language model decoding can help prevent unlikely sequences of words from being selected (*"the mat chased the rat"*) in favor of more likely predictions (*"the cat chased the rat"*).

While integration of language models has been a standard part of ASR systems, only recent works have been able to reach parity without using an explicit language model, either through knowledge distillation techniques [10], data augmentation [40], or self-training [48, 56]. Language model-free ASR systems are appealing due to the simplicity, but most still struggle with difficult ASR tasks, such as the noisy recordings of LibriSpeech *dev/test-other*. To our knowledge, there has been no work to date examining whether the properties of these systems transfer to L2 ASR.

3 Methods

We provide a background about our main methods in this section. We first introduce transfer learning for ASR and then follow with CTC and language model decoding.

3.1 Transfer Learning

For tasks with limited labeled data, training models from scratch becomes impractical. One approach that has great success is transfer learning. Transfer learning involves taking an existing model trained on one or more tasks from a given domain and transferring its knowledge to a target downstream task or domain [38]. Tasks which share the same label and feature space, but perhaps differ in feature distribution, can allow for a simple transfer learning method called model adaptation [51]. This allows for simply taking an existing model and re-training (i.e., "fine-tuning") it using a smaller domain-specific dataset. Model adaptation for ASR can be performed easily by freezing part of an existing model and re-training the rest on the new domain [26].

One particularly promising base model for transfer learning is wav2vec 2.0 [2], which is composed of a multi-layer convolutional neural network feature extractor and a transformer context network. The network uses a contrastive task for self-supervised pre-training to learn good general representations of audio. Following pre-training, the CNN feature extractor layers of the model are frozen, and the model is fine-tuned on domain-specific tasks by adding a linear layer on top of the transformer context network followed by training with CTC loss [2].

While the original models are strong baselines, the self-trained wav2vec 2.0 Large (LV-60) version of the model [56], which we will refer to as *Wav2Vec 2.0-ST* [56],[2] extends the original work with wav2vec 2.0 by applying a self-training approach. The model is pre-trained on 960 hours of speech data from LibriSpeech [39], followed with self-training on 53.2k hours of Libri-Light [24]. During the self-training process, pseudo-labels are generated using language models trained on the LibriSpeech corpus, allowing for transfer of knowledge from the language model into the ASR model proper ultimately resulting in a model with little need for an external model during inference time [56].

Fine-tuning of pre-trained wav2vec 2.0 is performed with CTC and the transcriptions of the audio segments. For each model, we identify the optimal hyperparameters on the respective Dev set. We choose hyperparameters as follows: For `mask_feature_prob`, we pick from *{0.25, 0.5}*; for `mask_feature_length`, we choose from *{15, 30}*; and for `mask_time_prob`, we use *{0.5, 0.75}* and a batch size of 16. To mimic the tri-state learning rate schedule [2], we set different learning rates for different stages: warm-up (1e–5, 3e–5), constant stage (1e–5, 3e–5), and decay (1e–5, 3e–5, 5e–6). The decay stage is followed by another constant stage (1e–5, 2e–6, 5e–6) to simulate Fairseq's fine-tuning configuration.

3.2 CTC Decoding

Because the output of CTC-trained models is a table of character probabilities for each timestep, this output must be decoded to find the most probable sequence of characters. One simple approach is to use a best path decoding strategy (see top left of Fig. 1), which simply involves outputting the highest probability token for each timestep condensing duplicate tokens and removing CTC blank symbols [11]. Following Graves [11], we can write the decoding as:

$$W^* = \arg\max_W p(W|X) \tag{1}$$

where W^* is our most likely sequence of characters (the labeling) and $p(W|X)$ is our probability of a labeling given a signal X. Then we can write best path decoding as:

$$W^* \approx \mathcal{F}(\pi^*) \tag{2}$$

[2] https://github.com/pytorch/fairseq/tree/master/examples/wav2vec.

where $\mathcal{F}$ is the CTC collapsing function, which removes duplicate letters and *blank* tokens, and $\pi*$ is the highest activation in the CTC output at a given timestep. The simplicity of this method allows for fast predictions, but at the cost of potential errors added through not considering combined probability states. As Graves [11] notes, this matters when the "label is weakly predicted for several consecutive timesteps"(p. 71). Several algorithms have been introduced to fix this shortcoming: *prefix search*, which allows for accounting for the probability of children nodes in the search graph [11]; *token passing*, which allows integration of a dictionary [11]; and *decoding with attention*, which uses a secondary RNN model to correct errors [58].

Many decoding strategies aim to also integrate a language model in the process, which allows for incorporating lexical information into the decoding process. N-gram language models can be formalized as:

$$p(w_i|w_{i-1}, w_{i-2}..., w_{i-n-1}) \tag{3}$$

where w_i is the ith word in the sequence, which we would like to estimate the probability of, and n is our n-gram size. Probabilities are generally calculated from a text corpus either through efficient statistical packages such as KenLM [17] or through training neural networks to generate probability distributions of the tokens. Additional decoding strategies that use language model probability re-weighting include modified beam search strategy [12, 16], weighted finite-state transducers [36, 43], character-level recurrent neural network (RNN) decoding [22, 33], or word-level RNN decoding [18].

In our experiments, we choose to apply the prefix beam search strategy for both decoding and including an external language model (see top right of Fig. 1). Instead of rehashing the full prefix beam search algorithm (see [16]), we focus on the main components needed to understand the hyperparameter optimization process of this decoding strategy. Prefix beam search attempts to find transcriptions which maximize the following equation (see [16]):

$$p_{CTC}(W; X)p_{LM}(W)^{\alpha}|W|^{\beta} \tag{4}$$

Here $p_{CTC}(W; X)$ is our CTC-trained neural network probability of a character sequence W given an input audio X, $p_{LM}(W)$ is the language model probability of sequence W, α is the language model weight term, and β is a word insertion penalty modifier. The algorithm to maximize the value in 4 is similar to normal beam search in the sense that it keeps track of a set of possible contenders $\leq k$, where we call k the beam width [32, 35]. For CTC, the complexities of duplicates and *blank* tokens mean that the actual probability of a given proposed sequence needs to be calculated as follows:

$$p(l; x_{1:t}) = (p_b(l; x_{1:t}) + p_{nb}(l; x_{1:t}))|W(l)|^{\beta} \tag{5}$$

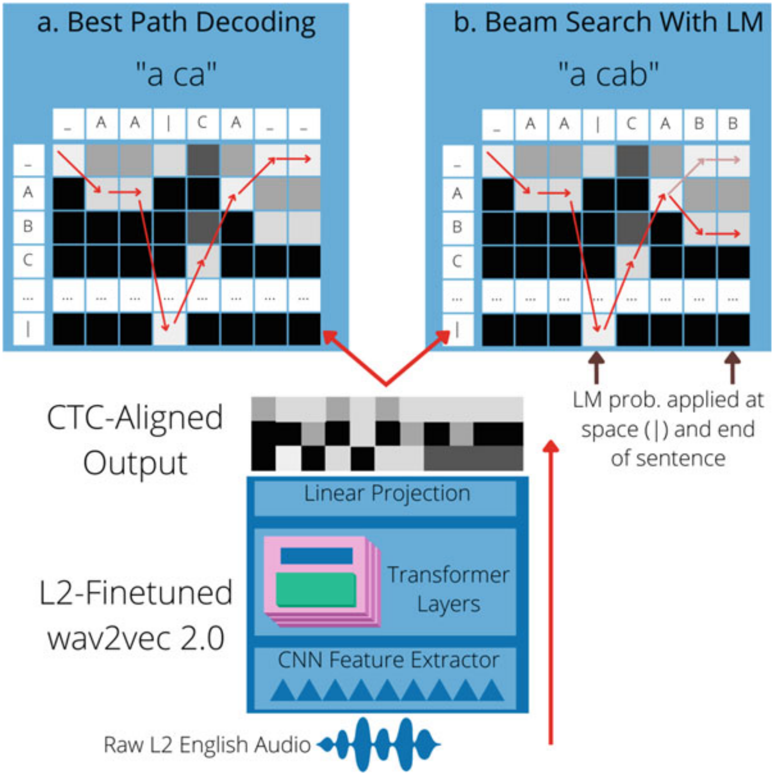


Fig. 1 The overall ASR pipeline. We **(a)** evaluate performance of wav2vec 2.0 without LM using *best path* decoding. We also **(b)** incorporate language model decoding with *beam search* along with the fine-tuned model

where $p(l; x_{1:t})$ is the probability of a given prefix, $p_b(l; x_{1:t})$ is the probability of a *blank* token being appended onto the current sequence, $p_{nb}(l; x_{1:t})$ is the probability of the next token being a character or punctuation (i.e., $non - blank$), and $|W(l)|^\beta$ is our word insertion term based on the words $W(l)$ in our proposed sequence l. A list of these probabilities is kept and updated based on the probabilities of each of the characters in the next segment of the CTC output table. When *space* characters are added to an existing sequence, the language model probability weight $p(W(l^+)|W(l))^\alpha$ is multiplied to the probability of the sequence, where $W(l^+)$ is the new set of words in the sequence. Values of α, which indicate how much to emphasize the language model, and values of β, the word insertion bonus, must be set via a hyperparameter tuning process.

In our experiments with adding language model decoding, we use the *pyctcdecode*[3] implementation of prefix beam search. It functions much the same way as

[3] https://github.com/kensho-technologies/pyctcdecode.

normal prefix beam search, differing only in several minor ways: first by using caching to speed up the decoding process and second by adding a partial word score which penalizes out of vocabulary word matches (based on checking whether the prefix is in a trie of the unigram vocabulary). For hyperparameter tuning, we perform a small grid search using the development set of L2-ARCTIC, with the ranges $\alpha \in \{0.5, 1, 1.5\}$ (considering both downweighting and overemphasizing the LM), $\beta \in \{0.5, 1, 1.5\}$, and $beamwidth \in \{50, 100, 150, 200\}$, with final hyperparameters as $\alpha = 1, \beta = 1.5$, and $beamwidth = 200$. For experiments on LibriSpeech, we similarly set hyperparameters on the development set (dev-other) and find the combination $\alpha = 0.5, \beta = 0.5, andbeamwidth = 100$ works best. To ablate the contribution of the language model, we also conduct an experiment on the full splits of L2-ARCTIC with $\alpha = 0$, effectively neutralizing the impact of the language model, keeping the rest of the hyperparameters the same.

4 Data

4.1 Corpus Information

We choose **L2-ARCTIC**, a non-native English speech corpus [59], for L2 fine-tuning. The recordings are from 24 non-native speakers of English with a total of 6 different L1s, and each of the L1s consists of 2 female speakers and 2 male speakers. The L1s we use for our experiments are Arabic (AR), Hindi (HI), Korean (KO), Mandarin (ZH), Spanish (ES), and Vietnamese (VI). Because L2-ARCTIC is based on the original L1 English corpus, CMU ARCTIC [25] (henceforth **L1-ARCTIC**, for simplicity), we can easily evaluate performance from fine-tuning on same-domain L1 data.

Each speaker in L2-ARCTIC contributed approximately 1 hour of phonetically balanced read speech based on the L1-ARCTIC prompts, which consist of carefully selected sentences (1, 132 sentence prompts) from Project Gutenberg [25]. We note this, as the pre-trained wav2vec 2.0 model we use was first pre-trained on LibriSpeech[4] [39] and then self-trained on Libri-Light[5] [24]. Both corpora rely on audiobooks from the LibriVox project,[6] much of which comes from Project Gutenberg.[7] However, because the ARCTIC corpus was selected to create a good phonological balance of sentences and weighted toward fiction [25], there may be domain mismatch between the sets of texts selected between these different corpora, and we aim to measure this with experiments using L1 fine-tuned models. Finally,

[4] http://www.openslr.org/12/.

[5] https://github.com/facebookresearch/libri-light.

[6] https://librivox.org.

[7] http://www.gutenberg.org.

we ensure there is no overlap in sentences between our L2-ARCTIC dev and test sets and the LibriSpeech training sets.

We also evaluate our fine-tuned models on (1) **LibriSpeech** to compare the fine-tuning with the original performance of *Wav2Vec 2.0-ST*. In addition, we evaluate on (2) **L1-ARCTIC**, identical to our L2-ARCTIC corpus but spoken by four native US English speakers, allowing us to identify any degradation on same-domain L1 speech performance, as well as estimate potential domain mismatch between the LibriSpeech corpus used to train *Wav2Vec 2.0-ST* and ARCTIC. Each of L1-ARCTIC speakers' datasets contains approximately the same number of utterances ($n = \sim 1, 132 * 4$) as each of L2-ARCTIC speakers' datasets.

For the purpose of our experiments, we define *native (L1) accents* as those represented in the LibriSpeech and L1-ARCTIC and *non-native (L2) accents* as those represented in L2-ARCTIC.

4.2 Data Splits

For both L2-ARCTIC and L1-ARCTIC, we split the data into three distinct Train, Dev, and Test sets with an 80:10:10 ratio. Importantly, we ensure there is *no overlap between utterances*. For L2-ARCTIC, we split the data across the following settings (see Fig. 2):

- **Split-1** *(speaker-dependent, multi-accent split)*: All speakers from all accents in the Train set are also included in the Dev and Test sets; however, no utterances are shared between Train, Dev, and Test.
- **Split-2** *(speaker-independent cross-validation splits with multiple accents)*: A speaker from each accent[8] is removed from the Train and Dev sets, but other speakers with the same accent remain in the Train and Dev sets.
- **Split-3** *(speaker-independent zero-shot splits with multiple accents)*: All speakers from one of the accents are entirely removed from the Train and Dev sets. The removed speakers are included in Test.
- **Split-4** *(all-speaker, single-accent split)*: Speakers are broken down by accents (six accents in total), and all speakers in a given accent are split into the Train, Dev, and Test sets (3 data splits x 6 accents).
- **Split-5** *(speaker-independent cross-validation splits with single accent)*: One speaker in each accent is removed from the Train and Dev sets, but the other speakers with the same accent remain in the Train and Dev sets. As there are four speakers per accent, four splits are created for each accent, which are further split into the Train, Dev, and Test sets (3 data splits x 6 accents x 4 speakers).

[8] We use the term "accent" here to loosely refer to variation in speakers with L1 other than English.

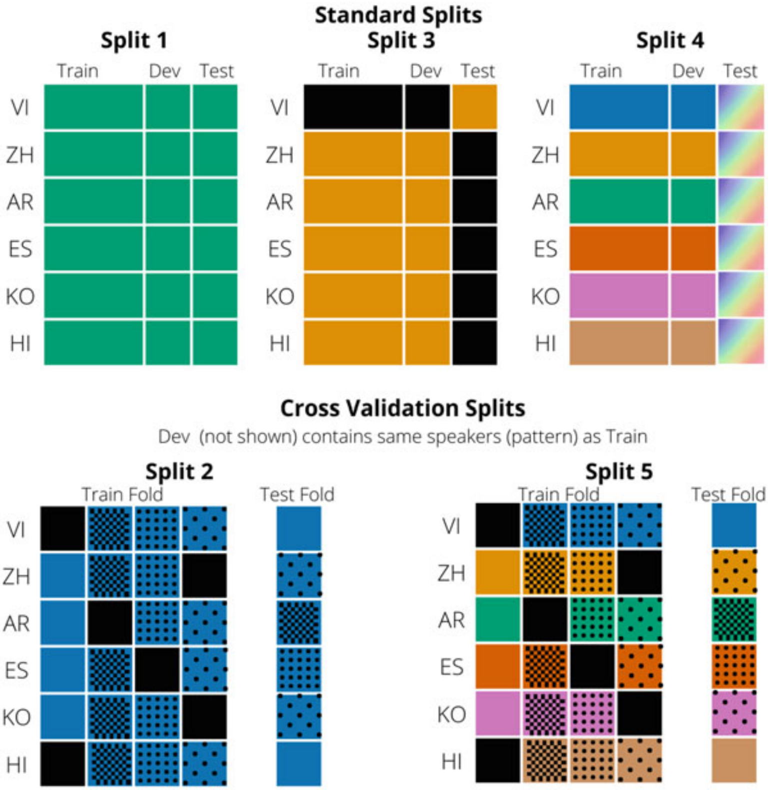


Fig. 2 The various data splits we use in our experiments. Color represents a different run of our training, with the rainbow blocks in Split 4 being present in all runs. For cross-validation splits, we show a single fold as an example. Speakers are indicated by a pattern with "held-out" speakers blacked out in the training set

5 Experiments

For all our wav2vec 2.0 models, we use Fairseq[9] fine-tuning default settings as a reference and convert the hyperparameters to align with HuggingFace's implementation. We evaluate all our models in terms of word error rate (WER). For L2 fine-tuning, we train each model with three random seeds and report the average WER. Our experiment code is available online.[10]

[9] https://github.com/pytorch/fairseq.

[10] https://github.com/UBC-NLP/L2ASR.

5.1 Baselines

We use the following baselines:

- **Baseline-I:** We use *Wav2Vec 2.0-ST* as a baseline, due to its strong performance on L1 English speech. We use the model released via HuggingFace.[11]
- **Baseline-II:** This is *Wav2Vec 2.0-ST*, the same as Baseline-I, fine-tuned on L1-ARCTIC described earlier. The purpose of Baseline-II is to measure potential domain shift between LibriSpeech/Libri-Light and ARCTIC, as well as to measure potential trade-offs from the fine-tuning process itself.

5.2 Multi-Accent Models

With our multi-accent models, we examine performance using multiple accents during training. We introduce each of our models here and present the results acquired with each. We provide a summary of our different data splits and models across accent and speaker dependency categories in Table 1.

Model-1 (Speaker- and Accent-Dependent) The model is fine-tuned with Split-1 data to identify any speaker-dependent training impact, as well as an upper limit on performance. In addition to evaluating on L2-ARCTIC Test, we evaluate on L1-ARCTIC Test and LibriSpeech in order to observe any changes in model performance on L1 English.

As Table 2 shows, our Model-1 achieves best performance on both Dev and Test of **L2-ARCTIC** as compared to our two baselines. On Test, our Model-1 acquires 25.66% improvement over our Baseline-I wav2vec 2.0 system on L2-ARCTIC (9.27 WER for our model vs. 12.47 WER for Baseline-I). This gain is not surprising and simply means that a model with access to L2 data for fine-tuning will improve over models fine-tuned with L1 data (Baseline-II, which is fine-tuned on L1-ARCTIC) or

Table 1 Summary of data splits, fine-tuning, and evaluation setups

	Accent dependency		Speaker dependency		
	Dependent	Independent	Dependent	Independent	
Multi-accent	Model-1 (Split-1)	x		x	
	Model-2 (Split-2)	x			x
	Model-3 (Split-3)		x		x
Single-accent	Model-4 (Split-4)	x	x	x	x
	Model-5 (Split-5)	x			x

[11] https://huggingface.co/facebook/wav2vec2-large-960h-lv60-self.

Table 2 Model-1 performance in word error rate (WER) (lower is better) on non-native accents (L2-ARCTIC) and native accents (L1-ARCTIC, LS_{dev} and LS_{test}). Baseline-I and Baseline-II are reported on the same Dev and Test sets of each corpus for comparison

	L2-ARCTIC		L1-ARCTIC		LS_{dev}		LS_{test}	
Model	Dev	Test	Dev	Test	Clean	Other	Clean	Other
Baseline-I	13.47	12.47	2.30	2.23	**1.69**	**3.55**	**1.86**	**3.89**
Baseline-II	17.29	15.95	**1.26**	**1.30**	2.19	5.13	2.32	5.00
Model-1	**9.78**	**9.27**	1.94	1.86	2.75	5.55	2.82	6.36

not-fine-tuned at all (Baseline-I). Nor is performance on **L1-ARCTIC** surprising: A model fine-tuned with native data (Baseline-II) outperforms one fine-tuned with accented data (our Model-1), both of which outperform a model without fine-tuning (Baseline-I). These results, however, show that in absence of L1 data, L2 data can be valuable for improving ASR model performance even on L1. For **LibriSpeech**, Baseline-I, which is trained on LibriSpeech data, outperforms the two fine-tuned models (our Model-1 and Baseline-II). The reason is that these two latter models are fine-tuned on a domain that is different from LibriSpeech. That is, fine-tuning models on out-of-domain data will, and as we see here does, result in deterioration of performance on in-domain data. We also note that our Model-1's performance on LibriSpeech is worse than that of Baseline-II on both the "Clean" (LS_{Clean}, native speech under quite recording environments) and "Other" (LS_{Other}, both noisy environment and accented recordings) Dev and Test splits. This may be because LibriSpeech is mostly comprised of L1 data and the greater variability on our L2-ARCTIC Train set (24 non-native speakers in our Model-1 vs. 4 native speakers in Baseline-II).

Model-2 (Speaker-Independent, Accent-Dependent) While Model-1 mimics a situation where we have some training data from speakers that we serve (i.e., test on), this is rarely a realistic scenario. We instead switch to a speaker-independent (but still *accent-dependent*) setting, Split-2. We carry out four-fold cross-validation with the 24 speakers in the data, every time using 18 speakers (3 speakers per accent) in Train[12] and 6 speakers in Test (1 per accent). We report the average of the four folds/runs, along with standard deviation.

As Table 3 shows, Model-2 performance is consistent with Model-1. Our Model-2 outperforms the two baselines on both Dev and Test, reaching 9.96 WER on Test compared to 12.47 for Baseline-I and 15.96 for Baseline-II. These results demonstrate that fine-tuning with multiple accents improves the accented ASR system without access to test speaker data.

Model-3 (Speaker- and Accent-Independent) To evaluate performance on *unseen* accents, we adopt a zero-shot strategy by removing one accent at a time from both Train and Dev sets and evaluating on the Test set of the removed accent,

[12] We use 10% of the utterances from these 18 speakers for development (Dev).

Table 3 Model-2 cross-validated performance on L2-ARCTIC Dev and Test sets, alongside Baseline-I and Baseline-II performance on the same cross-validation splits. Mean refers to the average WER over the four runs and SD refers to the standard deviation

	Dev_{L2}		Test_{L2}	
Model	Mean	SD	Mean	SD
Baseline-I	13.47	0.23	12.47	0.84
Baseline-II	17.29	0.41	15.96	1.58
Model-2	**9.57**	0.19	**9.96**	0.64

Table 4 Model-3 setting, where a different accent is removed each run. Test_{all} refers to Test of *all* 24 speakers, and $\text{Test}_{\text{zeroshot}}$ refers to Test of those 4 speakers who have $\text{L1}_{\text{removed}}$ accent. Baseline-I acquires 12.47 on Test_{all}, while Baseline-II acquires 15.95 on the same test set (i.e., Test_{all})

	Baseline-I	Baseline-II	Model-3		
$\text{L1}_{\text{removed}}$	$\text{Test}_{\text{zeroshot}}$	$\text{Test}_{\text{zeroshot}}$	Dev_{L2}	$\text{Test}_{\text{zeroshot}}$	Test_{all}
VI	23.30	28.81	7.96	18.81	9.43
ZH	14.85	19.32	9.02	12.13	9.08
AR	10.95	14.82	9.40	10.10	9.13
ES	10.48	13.48	9.38	8.89	8.98
KO	8.18	10.22	10.10	6.95	9.01
HI	6.93	8.93	10.29	6.67	9.11

Split-3. To evaluate model performance on each accent, we conduct six runs in total with one accent removed at a time.

As Table 4 shows, fine-tuning on accented speech benefits unseen accents and speakers (Model-3 setting). All the multi-accent, zero-shot models outperform Baseline-I and Baseline-II, which means each of the six accents benefit from other accents through this process of transfer learning. Our results also show that, in absence of in-accent data, some unseen accents are easier for the model than others. For example, on $\text{Test}_{zeroshot}$, Vietnamese (VI) is the most challenging (with 18.81 WER), and Hindi (HI) is the least challenging (with only 6.67 WER).

5.3 *Accent-Specific Models*

We evaluate the accent-dependent performance by fine-tuning our models on a single type of L1-specific accent at a time.

Model-4 (Speaker-Dependent, Accent-Dependent) The model is fine-tuned with Split-4 data to identify any accent-dependent training impact on downstream performance, as well as an upper bound on performance when the model is optimized for a single accent. In addition to evaluating on L2-ARCTIC Test, we

Table 5 Model-4 performance on L2 accent (Test_{L2}) and native accent (Test_{L1}, LS_{Clean}, LS_{Other}), compared with Baseline-I, Baseline-II, and Model-1. SD refers to the standard deviation

	Baseline-I	Baseline-II	Model-1	Model-4			
L1	Test_{L2}	Test_{L2}	Test_{L2}	Test_{L2}	Test_{L1}	LS_{Clean}	LS_{Other}
VI	23.30	28.81	15.14	**12.12**	2.02	3.08	6.96
ZH	14.85	19.32	11.49	**8.95**	1.82	2.84	6.22
AR	10.95	14.82	8.90	**6.92**	1.55	2.66	6.24
ES	10.48	13.48	8.92	**6.68**	1.56	2.53	6.11
KO	8.18	10.22	6.60	**4.99**	1.71	2.51	5.63
HI	6.93	8.93	5.51	**4.99**	1.52	2.36	6.05
Mean	12.45	15.93	9.43	7.44	1.70	2.66	6.20
SD	5.97	7.30	3.49	2.72	0.20	0.26	0.43

test the model on L1-ARCTIC Test and LibriSpeech as a means to identify any degradation on L1 English data.

As Table 5 shows, while the multi-accent model (Model-1) outperforms Baseline-I for all six accents, all of the accent-specific models (Model-4 setting) outperform Model-1 on the Test_{L2} setting despite the small amount of data (roughly 5 hours) used for fine-tuning each of the versions of Model-4. On average, Model-4 setting is two points WER better than Model-1. In addition, Model-4 type models (each of which is fine-tuned on one non-native accent) perform reasonably well on L1 data (Test_{L1}, LS_{Clean}, and LS_{Other}). Further, large accent-specific variability is observed across different model types on Test_{L2} ($SD = [2.72 - 7.30]$), compared with native counterparts such as Test_{L1} ($SD = [0.20 - 0.43]$). An interesting result is the apparent difficulty difference between different accents (HI and KO easiest, VI hardest), regardless of model types. We provide sample outputs from Model-4 in Table 11.

As shown in Table 6, we also perform accent-wise zero-shot evaluation. Results of this set of experiments reveal an interesting pattern: While fine-tuning on a single accent generally benefits *at least one other accent*, fine-tuning on the Hindi accent only benefits Hindi (the same accent) and hinders performance on *all the other accents*.

Strong to moderate positive correlations (see Fig. 3) are observed among ZH, KO, and VI (0.79 between ZH and KO, 0.44 between VI and ZH, 0.34 between VI and KO). On the contrary, HI accents have negative correlations with all the other L1s. Strong negative correlations with ZH, KO, and VI (−0.95, −0.73, and 0.67, respectively) suggest that the more we fine-tune on HI accents, the more detrimental to the performance on those three accents (and vice versa; those three accents would have negative impacts on HI performance).

Model-5 (Speaker-Independent and Accent-Dependent) This setup simulates a more realistic scenario where we target a single accent, without access to all speakers during development time. Thus, we use Split-5 data which mimics a speaker-independent setting. We cross-validate each L1 subset with one of the four speakers per fold. The hyperparameters we use are those identified for Model-4. To evaluate the performance on each speaker, we conduct 24 folds in total with

Table 6 Model-4 performance in the zero-shot setting. Bold fonts represent the accent whose WER drops the most in the zero-shot setting. For example, compared with Baseline-I, the VI-specific fine-tuning not only improves performance on VI (i.e., a drop in WER) but also improves on ZH despite ZH being the unseen accent. One notable pattern is that HI-specific fine-tuning only benefits HI-accented speech recognition, while all the other fine-tuning hinder performance on the HI accent

	VI	ZH	AR	ES	KO	HI
Baseline-I	23.30	14.85	10.95	10.48	8.18	6.93
VI-specific	12.12	13.62	13.01	9.95	8.55	9.62
ΔWER	−11.18	−1.23	2.06	−0.53	0.37	2.69
Δ%	−48.00	**−8.31**	18.84	−5.03	4.52	38.77
ZH-specific	20.37	8.95	11.42	9.79	6.82	10.91
ΔWER	−2.93	−5.90	0.47	−0.69	−1.36	3.98
Δ%	−12.58	−39.75	4.26	−6.62	**−16.67**	57.43
AR-specific	23.88	14.86	6.92	9.86	9.16	7.74
ΔWER	0.58	0.01	−4.03	−0.62	0.98	0.81
Δ%	2.47	0.07	−36.83	**−5.92**	11.94	11.69
ES-specific	20.71	13.99	11.00	6.68	7.92	8.66
ΔWER	−2.59	−0.86	0.05	−3.80	−0.26	1.73
Δ%	**−11.13**	−5.81	0.43	−36.23	−3.22	25.01
KO-specific	20.07	12.12	11.66	10.04	4.99	9.09
ΔWER	−3.23	−2.73	0.71	−0.44	−3.19	2.16
Δ%	−13.88	**−18.38**	6.45	−4.23	−39.04	31.17
HI-specific	26.18	18.39	13.51	11.90	10.72	4.99
ΔWER	2.88	3.54	2.56	1.42	2.54	−1.94
Δ%	**12.37**	23.82	23.35	13.55	31.01	−27.99

1 speaker removed at a time and report the average and standard deviation of the fourfolds per each accent. As Table 7 shows, speaker-dependent variability is small for Test$_{\text{all}}$ ($SD = [0.11 - 0.38]$) but large for Test$_{\text{zeroshot-speaker}}$ ($SD = [1.12 - 4.87]$). These results suggest that individual speaker's differences may play an important role in how much performance gain can be obtained by fine-tuning.[13]

5.4 Language Model Decoding

We evaluate the impact of language model decoding in comparison to the fine-tuning techniques already identified. We use a 4-gram KenLM model [17] trained on the

[13] For those speakers whose TOEFL scores are known [59], a strong negative correlation was observed between speaker-specific WERs of Baseline-I and speaker's TOEFL scores, $r(8) \approx -0.77$, $p < 0.01$.

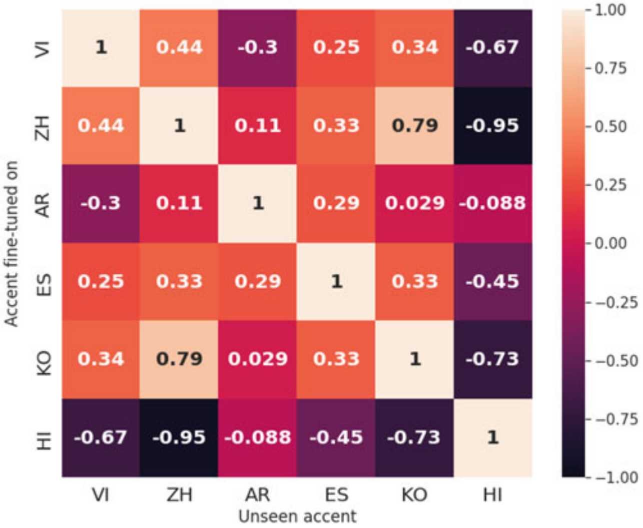


Fig. 3 Model-4 correlations of fine-tuning accent vs. test-accent percent performance change. Here we present the correlations based on Table 6, to show the accents that most benefit each other

Table 7 Model-5 performance on L2 accent. Test$_{\text{all}}$ contains utterances by all speakers within each L1, whereas Test$_{\text{zeroshot-speaker}}$ contains utterances by a single speaker that is absent in the training phase. Mean refers to the average WER over fourfolds for each L1, and SD refers to the standard deviation

	Test$_{\text{all}}$		Test$_{\text{zeroshot-speaker}}$	
L1	Mean	SD	Mean	SD
VI	12.67	0.38	14.28	4.87
ZH	9.65	0.31	11.26	3.03
AR	7.28	0.29	8.56	2.28
ES	6.95	0.26	7.76	3.99
KO	5.22	0.18	5.69	2.20
HI	5.27	0.11	5.79	1.12

concatenated LibriSpeech and ARCTIC training corpora. We find performance gain from language model decoding to be relatively similar to fine-tuning for most splits, with the combination of the two methods even more beneficial (see Tables 8 and 9). To further quantify the results, for each target accent, we calculate the average reduction from adding language model decoding and compare with the average reduction from fine-tuning, calculated as $\Delta WER_{LM} = AVG(B1_{LM} - B1, M_{LM} - M)$, while fine-tuning reduction is $\Delta WER_{FT} = AVG(M - B1, M_{LM} - B1_{LM})$ (see Fig. 4). For more difficult accents (VI, ZH), fine-tuning appears to play a much bigger role in performance improvements, with easier accents benefiting more from the language model decoding (HI, KO).

Table 8 Comparison of models with and without language model decoding on the full L2-ARCTIC Test set. We further ablate this by setting α to 0 to demonstrate performance with beam search, but without language model re-weighting. ΔWER indicates increase (+) or decrease (−) in WER given as a percent relative to the no LM results

	Best path ↓	Beam search ↓	ΔWER ↓
B1	12.47	8.43	−32.40%
$B1_{\alpha=0}$	12.47	12.74	+2.17%
B2	15.95	11.96	−25.02%
$B2_{\alpha=0}$	15.95	16.38	+2.70%
M1	**9.27**	**5.53**	**−40.32%**
$M1_{\alpha=0}$	9.27	9.42	+1.62%

Table 9 Comparison of models with and without language model decoding on the language background splits (subscript). Results in WER and relative decrease (−) in WER ($\Delta WER\%$)

	Best path ↓	Beam search ↓	ΔWER % ↓
$B1_{VI}$	23.30	17.01	−27.00%
$B2_{VI}$	28.81	22.60	−21.56%
$M4_{VI}$	**12.12**	**7.36**	**−39.23%**
$B1_{ZH}$	14.85	9.76	−34.28%
$B2_{ZH}$	19.32	14.28	−26.09%
$M4_{ZH}$	**8.95**	**5.98**	**−33.20%**
$B1_{AR}$	10.95	7.53	−31.23%
$B2_{AR}$	14.82	10.90	−26.45%
$M4_{AR}$	**6.92**	**4.72**	**−31.81%**
$B1_{ES}$	10.48	7.04	−32.82%
$B2_{ES}$	13.48	9.96	−26.11%
$M4_{ES}$	**6.68**	**3.96**	**−40.80%**
$B1_{KO}$	8.18	4.75	−41.93%
$B2_{KO}$	10.22	7.25	−29.06%
$M4_{KO}$	**4.99**	**2.80**	**−43.85%**
$B1_{HI}$	6.93	4.43	−36.08%
$B2_{HI}$	8.93	6.67	−25.31%
$M4_{HI}$	**4.99**	**2.78**	**−44.22%**

In evaluating beam search on its own ($\alpha = 0$), performance degrades slightly compared to the baseline. This indicates that most of the performance gain in decoding is coming from the inclusion of the language model. We note the B2 baseline not only performs worse than B1 baseline but additionally benefits the least from language model decoding (perhaps indicating that it has overfit to the L1-ARCTIC corpus). For L2 ASR, this suggests that simply fine-tuning on domain-specific but L1 accent corpora is counterproductive.

For performance on the LibriSpeech corpus (see Table 10), results are more mixed. As already shown by earlier work [56], *Wav2Vec 2.0-ST* benefits only slightly from language model decoding on the clean split of LibriSpeech ($\Delta WER-$ 0.05). The fine-tuned models show mild benefit from language model decoding,

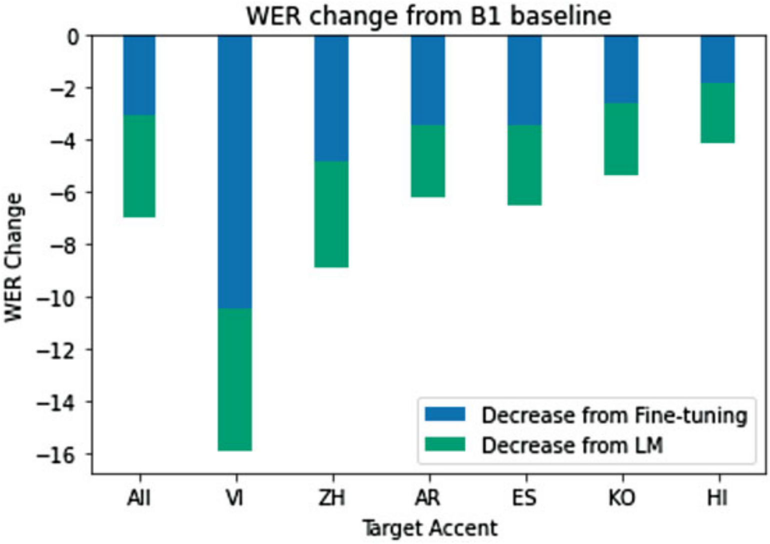


Fig. 4 Here we show the average absolute WER reduction from adding a language model compared with fine-tuning on the different test splits

Table 10 Comparison of models with and without language model decoding on the test-clean and test-other splits of LibriSpeech. Results in WER and relative decrease (−) or increase (+) in WER ($\Delta WER\%$)

	LS test-clean			LS test-other		
	Best path ↓	Beam search ↓	ΔWER % ↓	Best path ↓	Beam search ↓	ΔWER % ↓
B1	1.86	1.81	−2.69%	3.89	3.67	−5.66%
B2	2.32	1.96	−15.52%	5.00	4.34	−13.20%
M1	2.82	2.54	−11.30%	6.36	5.42	−17.35%
M4VI	3.08	2.78	−10.78%	6.96	5.97	−16.53%
M4ZH	2.83	2.56	−10.53%	6.22	5.38	−15.54%
M4AR	2.66	2.42	−10.06%	6.24	5.38	−15.92%
M4ES	2.53	2.31	−9.51%	6.12	5.30	−15.61%
M4KO	2.51	2.24	−12.22%	5.64	4.82	−16.86%
M4HI	2.36	2.12	−11.16%	6.05	5.19	−16.63%

though some (M4VI, M4ES) are not statistically significantly. For the more difficult *test-other*, we however observe across the board improvements in performance for all models using language model decoding. The overall performance gains from adding a language model and beam search to decoding L1 speech are minor in comparison to the benefits in L2 speech decoding, and fine-tuning on L2 speech decreases the performance on L1 speech substantially even when compared with better decoding (M1 results *test-clean* 40.33% and *test-other* 47.68% increase from

B1). This suggests an *L1 vs. L2 trade-off* that cannot entirely be overcome by the combination of fine-tuning and decoding strategies we have tried.

6 Error Analysis

To understand the benefit of fine-tuning and language model decoding, we further analyze the types of error corrected by the respective approaches, using Levenshtein single-character edit operations (as measured from gold standard to predicted utterance) as our proxy for types of errors. For this analysis, we use the L2-ARCTIC development set. An interesting finding of our analysis (see Fig. 5) is that while adding language model decoding to the B1 baseline improves WER on L2-ARCTIC, it increases the number of *deletion* operations, indicative of over-generation. For fine-tuned models (M1 and M4), there is reduction in error types across the board, with particular benefit to *substitution* ($M4$, −43%) and *deletion* operations ($M4$, −55%) and mild benefit to *insertion* operations ($M4$, −30%). For adding a language model on top of the fine-tuning, we see further reduction in the *substitution* operations ($M4_{LM}$, −64%) and *insertion* operations ($M4_{LM}$, −52%), with mild benefit to *deletion* operations ($M4_{LM}$, −65%) (Table 11).

We give examples of some of these errors in Table 12 and use the B1 predictions on the Dev set of L2-ARCTIC to explore them in more detail. When looking at cases of single Levenshtein edit operations, we notice the following patterns: Out of the 18 *deletion* operations, 15 are spelling errors, 1 is a pluralization error, 1 is a tense

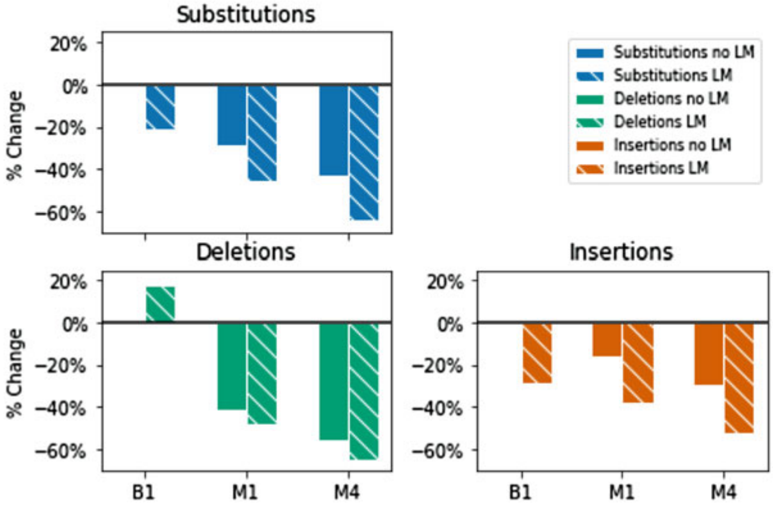


Fig. 5 The change in number of Levenshtein edit operations compared to our baseline B1 with best path decoding and no language model

Table 11 Examples of transcription output of selected utterances from the Test set of Model-4 among all six L1s without a language model. Capitalized words indicate errors. We show samples from two speakers per accent

Model	Model output
Ref	At lake linderman i had one canoe very good peterborough canoe
VI	At LAY LINDEMAN i had one canoe very good PETERBORROUG CANOES
	A lake LNDER MAN i had one canoe very good BIET OF ROCK canoe
ZH	At lake LINGERMAN i had ONCE canoe very good PETERBROUGH canoe
	At lake LINERMAN i had one canoe very good PETERE BROUGHTA canoe
AR	At lake LUNDERBOGH i had one canoe very good BITTERBOROUGH canoe
	At lake LUNDERMAN i had one canoe very good BETTER BORT canoe
ES	At lake linderman i had one canoe a very good PETERBOURN canoe
	At lake linderman i had ONCE canoe very good PIERREBOROUGH canoe
KO	At lake linderman i had one canoe very good peterborough canoe
	At lake LINDEMAN i had ONCE canoe very good PITTEBRAUG canoe
HI	At lake LINDEMAN i had one canoe very good PETERBURGH canoe
	At lake linderman i had one canoe A very good PEACHERBROROU canoe

Table 12 Examples of transcription output of different categories of edit operation. We use a shorthand to indicate the applied method: fine-tuning, $+FT$; language model decoding, $+LM$; or lack thereof $(-)$. Errors capitalized

Edit type	Model	Model output
	Ref	The portuguese boy crawled nearer and nearer
Substitution	$-FT-LM$	The portuguese boy CROWLED nearer and nearer
	$-FT+LM$	The portuguese boy crawled nearer and nearer
	$+FT-LM$	The portuguese boy CROWLED nearer and nearer
	$+FT+LM$	The portuguese boy crawled nearer and nearer
	Ref	Tomorrow it will be strong enough for you to stand upon
Insertion	$-FT-LM$	TO MORROW it will be strong enough for you to stand upon
	$-FT+LM$	TO MORROW it will be strong enough for you to stand upon
	$+FT-LM$	Tomorrow it will be strong enough for you to stand upon
	$+FT+LM$	Tomorrow it will be strong enough for you to stand upon
	Ref	There are no kiddies and half grown youths among them
Deletion	$-FT-LM$	There are no kiddies and half grown YOUTH among them
	$-FT+LM$	There are no kiddies and half grown YOUTH among them
	$+FT-LM$	There are no kiddies and half grown YOUTH among them
	$+FT+LM$	There are no kiddies and half grown youths among them

error, and 1 is a spacing error. For the 18 *substitution* operations, all are indicative of spelling errors. Of the 11 *insertion* operations, 7 are spacing errors and 3 are spelling errors, and 1 is a pluralization error. Using this rough analysis, it appears that while both fine-tuning and language model decoding substantially improve spelling of the model, language model decoding is more effective at ensuring proper spacing of words.

7 Conclusion

We demonstrated the potential of developing accent-independent and accent-dependent models that improve non-native speech recognition simply by fine-tuning the pre-trained wav2vec 2.0 model on a small amount of labeled data. Both our multi- and single-accent models improve performance on L2 English speakers. However, each accent benefits differently: Results of the multi-accent, zero-shot experiments suggest that transfer learning on accent is possible and single-accent models improve the most for the target L2 accents. Comparing the benefit from using a language model in decoding the ASR outputs with simply fine-tuning the models, we find that both these methods yield comparable improvements. We also find that the combination of the two methods greatly closes the gap between L1 and L2 ASR. We summarize our findings as follows:

- Fine-tuning either on single accents (most effective) or groups of accents (more generalizable) significantly improves L2 ASR performance. This is possible through large reductions in calculated substitution and deletion operations.
- Fine-tuning on domain-specific L1 accents is counterproductive to L2 ASR.
- Language model decoding is useful for L2 ASR, even for models with strong language model-free performance on L1 speech, and is particularly good at reducing calculated substitution and insertion operations.

When looking at future research directions, it is important to stress the need the field has for benchmark L2 ASR datasets. Datasets proposed by Wang et al. [53] is one of the few L2 ASR datasets collected with this intention in mind (although these only cover L1 Chinese speakers). Without datasets to cover a wide variety of L2 English accents, relying on accent embeddings [23, 49, 50] and multi-task learning might be a vital addition to L2 ASR work if the goal is wide-accent coverage. While our fine-tuning of *Wav2vec 2.0-ST* did not seem to keep the model's language model-free performance, there may be other directions to take to try to maintain it. These can include looking more closely at fine-tuning techniques, such as Layer Norm and Attention fine-tuning [27] or Adapter fine-tuning [20]. These might be better at preserving this internal language model, as they freeze more of the original model weights. Finally, smaller ASR model size and federated learning—although early results have noted difficulties in applying it to ASR [57], effort is underway to lower the training cost and improve accuracy [15]—might bring about the potential for ASR individualization to be targeted at the level of idiolects, with people able to use ASR models tailored to their personal accent profile.

Acknowledgments We would like to thank Mia Li, Jeremy Zhang, and Haejin Cho for having contributed to an initial phase of this work.

References

1. Baevski, A., Schneider, S., Auli, M.: vq-wav2vec: Self-supervised learning of discrete speech representations (2019). Preprint arXiv:191005453
2. Baevski, A., Zhou, H., Mohamed, A., Auli, M.: wav2vec 2.0: A framework for self-supervised learning of speech representations (2020). Preprint arXiv:200611477
3. Bearman, A., Josund, K., Fiore, G.: Accent conversion using artificial neural networks. Technical Report, Stanford University, Technical Report, Tech. Rep (2017)
4. Chan, W., Jaitly, N., Le, Q., Vinyals, O.: Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 4960–4964 (2016)
5. Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: Advances in Neural Information Processing Systems, pp 577–585 (2015)
6. Chung, Y.A., Hsu, W.N., Tang, H., Glass, J.: An unsupervised autoregressive model for speech representation learning (2019). Preprint arXiv:190403240
7. Crystal, D.: English as a global language. Ernst Klett Sprachen, Stuttgart (2003)
8. Das, N., Bodapati, S., Sunkara, M., Srinivasan, S., Chau, D.H.: Best of both worlds: Robust accented speech recognition with adversarial transfer learning (2021). Preprint arXiv:210305834
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018). Preprint arXiv:181004805
10. Futami, H., Inaguma, H., Ueno, S., Mimura, M., Sakai, S., Kawahara, T.: Distilling the knowledge of bert for sequence-to-sequence ASR (2020). Preprint arXiv:200803822
11. Graves, A.: Connectionist temporal classification. In: Supervised Sequence Labelling with Recurrent Neural Networks, pp. 61–93. Springer, Berlin (2012)
12. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: International Conference on Machine Learning, pp. 1764–1772 (2014)
13. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 369–376 (2006)
14. Gulati, A., Qin, J., Chiu, C.C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu Y., et al.: Conformer: Convolution-augmented transformer for speech recognition (2020). Preprint arXiv:200508100
15. Guliani, D., Beaufays, F., Motta, G.: Training speech recognition models with federated learning: A quality/cost framework. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3080–3084 (2021)
16. Hannun, A.Y., Maas, A.L., Jurafsky, D., Ng, A.Y.: First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs (2014). Preprint arXiv:14082873
17. Heafield, K.: KenLM: Faster and smaller language model queries. In: Proceedings of the sixth Workshop on Statistical Machine Translation, pp. 187–197 (2011)
18. Hori, T., Cho, J., Watanabe, S.: End-to-end speech recognition with word-based rnn language models. In: 2018 IEEE Spoken Language Technology Workshop (SLT), IEEE, pp. 389–396 (2018)
19. Hou, J., Guo, P., Sun, S., Soong, F.K., Hu, W., Xie, L.: Domain adversarial training for improving keyword spotting performance of esl speech. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 8122–8126 (2019)
20. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for NLP. In: International Conference on Machine Learning, PMLR, pp. 2790–2799 (2019)
21. Hu, H., Yang, X., Raeesy, Z., Guo, J., Keskin, G., Arsikere, H., Rastrow, A., Stolcke, A., Maas, R.: Redat: Accent-invariant representation for end-to-end asr by domain adversarial training

with relabeling. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 6408–6412 (2021)

22. Hwang, K., Sung, W.: Character-level incremental speech recognition with recurrent neural networks. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 5335–5339 (2016)
23. Jain, A., Upreti, M., Jyothi, P.: Improved accented speech recognition using accent embeddings and multi-task learning. In: Interspeech, pp. 2454–2458 (2018)
24. Kahn, J., Rivière, M., Zheng, W., Kharitonov, E., Xu, Q., Mazaré, P.E., Karadayi, J., Liptchinsky, V., Collobert, R., Fuegen, C., et al.: Libri-light: A benchmark for asr with limited or no supervision. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 7669–7673 (2020)
25. Kominek, J., Black, A.W., Ver, V.: CMU ARCTIC databases for speech synthesis (2003)
26. Kunze, J., Kirsch, L., Kurenkov, I., Krug, A., Johannsmeier, J., Stober, S.: Transfer learning for speech recognition on a budget (2017). Preprint arXiv:170600290
27. Li, X., Wang, C., Tang, Y., Tran, C., Tang, Y., Pino, J., Baevski, A., Conneau, A., Auli, M.: Multilingual speech translation with efficient finetuning of pretrained models (2020). Preprint arXiv:201012829
28. Ling, S., Liu, Y., Salazar, J., Kirchhoff, K.: Deep contextualized acoustic representations for semi-supervised speech recognition. In: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 6429–6433 (2020)
29. Liu, A.T., Yang, S.W., Chi, P.H., Hsu, P.C., Lee, H.Y.: Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 6419–6423 (2020)
30. Liu, S., Wang, D., Cao, Y., Sun, L., Wu, X., Kang, S., Wu, Z., Liu, X., Su, D., Yu, D., et al.: End-to-end accent conversion without using native utterances. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 6289–6293 (2020)
31. Livescu, K., Glass, J.: Lexical modeling of non-native speech for automatic speech recognition. In: 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100), IEEE, vol 3, pp. 1683–1686 (2000)
32. Lowerre, B.T.: The Harpy Speech Recognition System. Carnegie Mellon University (1976)
33. Maas, A., Xie, Z., Jurafsky, D., Ng, A.Y.: Lexicon-free conversational speech recognition with neural networks. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 345–354 (2015)
34. Matassoni, M., Gretter, R., Falavigna, D., Giuliani, D.: Non-native children speech recognition through transfer learning. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 6229–6233 (2018)
35. Meister, C., Vieira, T., Cotterell, R.: If beam search is the answer, what was the question? (2020) Preprint arXiv:201002650
36. Miao, Y., Gowayyed, M., Metze, F.: Eesen: End-to-end speech recognition using deep RNN models and WFST-based decoding. In: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), IEEE, pp. 167–174 (2015)
37. Oord, A.V.D., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding (2018). Preprint arXiv:180703748
38. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2009)
39. Panayotov, V., Chen, G., Povey, D., Khudanpur, S.: Librispeech: An ASR corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 5206–5210 (2015)
40. Park, D.S., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E.D., Le, Q.V.: Specaugment: A simple data augmentation method for automatic speech recognition (2019). Preprint arXiv:190408779

41. Ping, T.T.: Automatic speech recognition for non-native speakers. PhD Thesis, Université Joseph-Fourier-Grenoble I (2008)
42. Radzikowski, K., Wang, L., Yoshie, O., Nowak, R.: Accent modification for speech recognition of non-native speakers using neural style transfer. EURASIP J. Audio Speech Music Process. **2021**(1), 1–10 (2021)
43. Sak, H., Senior, A., Rao, K., Irsoy, O., Graves, A., Beaufays, F., Schalkwyk, J.: Learning acoustic frame labeling for speech recognition with recurrent neural networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 4280–4284 (2015)
44. Schneider, S., Baevski, A., Collobert, R., Auli, M.: wav2vec: Unsupervised pre-training for speech recognition (2019). Preprint arXiv:190405862
45. Shi, X., Yu, F., Lu, Y., Liang, Y., Feng, Q., Wang, D., Qian, Y., Xie, L.: The accented english speech recognition challenge 2020: Open datasets, tracks, baselines, results and methods. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 6918–6922 (2021)
46. Shibano, T., Zhang, X., Li, M.T., Cho, H., Sullivan, P., Abdul-Mageed, M.: Speech technology for everyone: Automatic speech recognition for non-native english with transfer learning (2021). Preprint arXiv:211000678
47. Sun, S., Yeh, C.F., Hwang, M.Y., Ostendorf, M., Xie, L.: Domain adversarial training for accented speech recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 4854–4858 (2018)
48. Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Sriram, A., Liptchinsky, V., Collobert, R.: End-to-end ASR: from supervised to semi-supervised learning with modern architectures (2019). Preprint arXiv:191108460
49. Turan, M.A.T., Vincent, E., Jouvet, D.: Achieving multi-accent asr via unsupervised acoustic model adaptation. In: INTERSPEECH 2020 (2020)
50. Viglino, T., Motlicek, P., Cernak, M.: End-to-end accented speech recognition. In: Interspeech, pp. 2140–2144 (2019)
51. Wang, D., Zheng, T.F.: Transfer learning for speech and language processing. In: 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), IEEE, pp. 1225–1237 (2015)
52. Wang, Z., Schultz, T., Waibel, A.: Comparison of acoustic model adaptation techniques on non-native speech. In: 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings (ICASSP'03), IEEE, vol. 1, pp. I–I (2003)
53. Wang, Y., Luan, H., Yuan, J., Wang, B., Lin, H.: Laix corpus of chinese learner english: Towards a benchmark for l2 english asr. In: INTERSPEECH, pp. 414–418 (2020)
54. Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang. X., Zhang, F., et al.: Transformer-based acoustic modeling for hybrid speech recognition. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal, Processing. (ICASSP), IEEE, pp. 6874–6878 (2020)
55. Watanabe, S., Hori, T., Kim, S., Hershey, J.R., Hayashi, T.: Hybrid CTC/attention architecture for end-to-end speech recognition. IEEE J. Selec. Top. Signal Process. **11**(8), 1240–1253 (2017)
56. Xu, Q., Baevski, A., Likhomanenko, T., Tomasello, P., Conneau, A., Collobert, R., Synnaeve, G., Auli, M.: Self-training and pre-training are complementary for speech recognition. In: ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 3030–3034 (2021)
57. Yu, W., Freiwald, J., Tewes, S., Huennemeyer, F., Kolossa, D.: Federated learning in ASR: Not as easy as you think. In: Speech Communication; 14th ITG Conference, VDE, pp 1–5 (2021)
58. Zenkel, T., Sanabria, R., Metze, F., Niehues, J., Sperber, M., Stüker, S., Waibel, A.: Comparison of decoding strategies for ctc acoustic models (2017). Preprint arXiv:170804469
59. Zhao, G., Sonsaat, S., Silpachai, A.O., Lucic, I., Chukharev-Hudilainen, E., Levis, J., Gutierrez-Osuna, R.: L2-ARCTIC: A non-native english speech corpus. In: INTERSPEECH Perception Sensing Instrumentation Lab (2018)

Kabyle ASR Phonological Error and Network Analysis

Christopher Haberland and Ni Lao

Abstract Training on graphemes alone without phonemes simplifies the speech-to-text pipeline. However, models respond differently to training on graphemes of different writing systems. We investigate the impact of differences between Latin and Tifinagh orthographies on automatic speech recognition quality on a Kabyle Berber speech corpus. We train on a corpus represented in a Latin orthography marked for vowels and gemination and subsequently transliterate model output to a consonantal Tifinagh orthography not marked for these features, which results in 10% absolute improvement in word error rate over a model trained on the unmarked orthography. We find that this performance gain is primarily due to a reduced error rate for graphemes marked for vocalic and voiced consonantal phonemes. However, this overall improvement is tempered by a reduction in recognition quality for other phonemes, especially allophonic spirantized consonants that are replete in the Kabyle language and many Berber dialects more widely. We also introduce new methods to characterize the disparity in performance between ASR models by analyzing outputs in terms of phonological networks. To our knowledge, this is the first work analyzing phonological networks of artificial neural network speech model outputs. Our results suggest that inputs written in defective orthographies lead to worse recognition quality for modern speech-to-text architectures compared to those fully marked for vowels and gemination.

Ni Lao contributed to this chapter while he was working at SayMosaic Inc.

C. Haberland (✉)
USAA, San Antonio, TX, USA
e-mail: crh2ke@virginia.edu

N. Lao
Google, Mountain View, CA, USA

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_3

1 Introduction

Graphemic modeling units and their correspondence with the spoken word can vary between different language communities [1], and even a single language community may have multiple orthographic conventions for application in different contexts [2] (diglossia). Minority languages in particular have often undergone less standardization [3], contributing to a greater tendency to be written in multiple orthographies. Improving speech technologies to support minority and "low-resource" languages and orthographies is crucial to ensuring their vitality and their users' access to information in the digital era [4]. Poor quality of low-resource language systems can compel users to interact with ASR systems in languages of which they are non-native, diminishing use of their native language. Furthermore, high error rates for a low-resource language ASR systems disadvantage monolingual speakers of the low-resource language that have a limited ability to switch to systems in more prevalent languages with better recognition quality.

Modern speech-to-text (S2T) models are trained on audio data paired with sequences of modeling units [5], which may be graphemes, phonemes, or other representations [6] that represent the linguistic constituents. Training models on phonemes constitutes a general paradigm in the creation of S2T systems [7] especially in the context of low-resource languages [8]. Training on phonemes can be advantageous for decoding out-of-vocabulary words or words from an external language [9], but manual annotation of speech data can be prohibitively expensive for low-resource languages [4].

ASR pipelines often include a component to automatically generate phoneme-based training data through grapheme to phoneme (G2P) conversions [10, 11] by training supervised models [12–14] or constructing rule-based systems [15]. There is an emerging trend toward G2P conversion with minimal intervention and preparation to streamline the end-to-end learning process. Several systems intend to streamline the G2P process using different methods, including self-training [16], ensembles of varying degrees of supervision [17], and leveraging open dictionaries of high-resource languages [18]. For low-resource languages, training S2T systems with graphemes alone obviates the G2P step in the S2T pipeline and the need for language-specific expert annotations [19]. There is also evidence that neural speech models implicitly learn phonemes at intermediate hidden layers when training on graphemes [20]. A number of different methods, such as diagnostic probing techniques and Representational Similarity Analysis, have been applied to show phonological learning by peering into neural model internals [21]. However, further research is required to show how S2T model quality responds to training on graphemes of various writing systems and orthographies in practice.

In this chapter, we study the impact of using a fully featured orthography instead of a consonantal orthography on S2T performance for Kabyle, a Berber language of northern Algeria. We chose to experiment with this language to augment the discussion surrounding orthographic choice on S2T quality that has been conducted primarily on Semitic languages that are comparatively more resourced, such as

Arabic. While several previous studies [22–24] have demonstrated the effect of training and decoding using defective (*i.e., omitting vocalic information*) and non-defective orthographies separately, our study is the first to compare a neural speech model's performance between (a) training and decoding in a defective orthography and (b) training on a non-defective orthography and decoding into its defective representation. Our study is also the first to analyze the nature of phonemic errors made by neural ASR models trained on a corpus in a defective and non-defective orthography to understand any systematic difference of types of errors made by models trained on these orthographies. The results demonstrate the importance of including vocalic graphemic inputs for improved S2T recognition of vowels and voiced consonants. To our knowledge, this result represents the first S2T system trained on a Tifinagh-encoded corpus of a Berber language. Our study is also unique in being the first to apply phonological network methods to characterize the differences between phonological networks between neural ASR model outputs to compare them against those of their respective gold vocabularies. We find that phonological networks of learned ASR vocabularies are significantly denser and less modular than gold vocabularies and publish our network data to encourage further investigations of phonological networks of ASR models.

2 Background

2.1 ASR Modeling Units

The investigation of orthographic choices on S2T system performance parallels psycholinguistic and cognitive science research on humans' linguistic and conceptual comprehension from auditory and visual information. A significant body of research aims to uncover how different G2P correspondences across writing systems may predict reading level achievement and interactions with dyslexia [25, 26]. For example, the work [27] assesses the reading abilities of children diagnosed with dyslexia when taught a novel orthography consisting of new G2P mappings. The work [28] studies the effect of diacritization and non-diacritization of dyslexic and non-dyslexic readers' processing of the Arabic script and found spelling knowledge of study participants to be the most significant predictor of processing speed.

S2T learning solely with graphemes has a long history [29]. Recently, studies have focused on identifying the differences between training on phonemic and graphemic inputs. The authors in [30] report that the phonemic–graphemic performance gap closes when model architecture and hyperparameters are attuned to the specific data input. Rao and Sak [31] found improved performance of graphemically trained models in multi-accented corpora and in trials of increased input data scale. Other work has tested derivatives of graphemes, such as bytes [32], wordpieces [31], and context-dependent graphemes (i.e., chenones) [19, 33]. Wang et al. [33] achieved state-of-the-art error rates on English data with graphemically derived modeling units for English.

2.2 Diacritization

Imputation of diacritics to augment defective model inputs has been, and continues to be, another active area of research, especially in the context of Arabic speech-to-text system design [34–38]. Diacritic imputation systems are designed to help computational models resolve heterophonic homographs or congruent graphemic sequences that have multiple phonemic interpretations, in orthographies that do not mark certain features. Sequences of this type are prevalent in consonantal writing systems, such as that used for Arabic, in which roughly one-third of tokens may be pronounced differently when not diacritized [39].

There has been work investigating diacritization's effect on speech modeling in languages that are written in defective orthographies or those not marked for certain phonemes. Afify et al. [40] used HMMs to demonstrate that training on voweled graphemes could increase performance over training on unvowelled graphemes on Arabic broadcast transcripts, even when decoding into unvowelled text. However, to the authors' knowledge, this has not been demonstrated in modern neural speech models with CTC decoding. More recently, [22] showed that training neural acoustic models upon voweled graphemes generally improved WER over unvowelled graphemes *when decoding into the same orthography*. The authors of [41] pre-annotate training transcripts with phonetic information deduced from graphemic context with rules to improve system performance. Alshayeji et al. [23] and Al-Anzi and AbuZeina [24] compare diacritized and non-diacritized input with various S2T model architectures and hyperparameters and observe higher WER for diacritized trials, though they do not train on diacritized data and decode on non-diacritized data.

Augmenting inputs via transliteration has been shown to improve S2T systems or machine translation performance. The authors of [42] transliterate model output as a post-process to improve the recognition of code-switched speech. Le and Sadat [43] and Cho et al. [44] model the G2P task as a neural sequence-to-sequence model and record improvements in named entity recognition and code-switched speech for Vietnamese and mixed Korean–Chinese scripts, respectively. While these studies use neural G2P models, rule-based systems have been developed for under-resourced languages [45, 46].

2.3 Berber Language Tools

To date, there have been limited efforts that apply neural speech models to Berber languages. OCR techniques have been applied to Tifinagh recently [47, 48], and [49] produced a pronunciation dictionary for speech modeling of phonemes. However, to the best of our knowledge, the ASR research community has not documented the training of Berber S2T models aside from those produced from the CommonVoice initiative [50] trained with a Latin-script corpus. We cite [51] as a limited exception, who describe a speech recognition system for Tarifit to recognize spoken numbers in noisy environments.

2.4 Phonological Networks

Phonological network analysis stems from investigations of how humans organize, internalize, recall, and reproduce words during linguistic processing. The authors of [52] were among the first to computationally model and understand the network effects of word similarity on auditory word recognition. Vitevitch [53] and later authors apply the methodology to understand the properties of phonological networks for specific language vocabularies and also to compare them. The authors in [54] use phonological networks to attempt to discover the differences between lexicon structures of different languages, identifying a robustness of connectivity of the networks in response to node removal and small giant components compared to complex networks in other domains. These authors also reveal general trends and commonalities between various languages across phonological network statistics.

Shoemark et al. [55] further improves the methodology and argues the need to control for network size (vocabulary size), to establish baselines for random networks governed by similar properties, and account for morphological processes by casting vocabularies as sets of lemmata. This resulted in more robust measures for phonological network comparison across languages while controlling for expected network variability, word length, and phonological inventory size. They found that most languages exhibit very similar network statistics trends as a result of the way phonological networks are defined but note certain cross-linguistic differences in the average shortest path length (ASPL) and small-world property between languages at different sampled vocabulary sizes. However, the authors did not find conclusive evidence that phonological networks represented "deeper organization within language" as [54] stated.

Beyond comparative linguistics, studies have also attempted to study monolingual phonological networks in the context of language acquisition. Siew [56] uses Louvain optimization to find communities among the phonological network analyzed by Vitevitch [53] and finds that larger communities are more likely to contain short, frequent, and highly connected words and low average age of acquisition ratings and a clustering effect of similar phonological segments in each community. The authors in [57] find an inverse preferential attachment effect as new words are acquired in language learners' networks. Siew and Vitevitch [58] extends the phonological network to cover orthographic differences to uncover joint effects on visual word recognition and spoken word recognition. Neergaard et al. [59] studies monolingual and bilingual speakers of Mandarin and English to understand the differences in structure, cohesion, and interconnectedness of elicited phonological networks. Turnbull [60] describes the graph-theoretic properties of the most common type of phonological networks applied in this literature. Despite considerable efforts to apply connectivist-theoretical methods in the realms of psycholinguistics and comparative linguistics, phonological networks have not heretofore been applied to the analysis of computational speech recognition models to compare lexical network structures with individual and language-wide vocabularies. To our knowledge, no prior work has described phonological networks of artificial neural network speech model outputs.

3 The Kabyle Language and Berber Writing Systems

Kabyle is a Berber language spoken in northern Algeria that has historically been written in Latin, Arabic, and Tifinagh scripts. Contemporary Kabyle is most widely written in a Latin orthography popularized by the linguist Mouloud Mammeri in a 1976 grammar of the language, though the Arabic and Tifinagh scripts are still promoted among certain groups within Algeria society [61]. Souag [61] contends that the Latin script predominates over the others in modern usage.

The alphabetic Neo-Tifinagh orthographies came into use after language planning initiatives for the Berber languages in the mid-twentieth century carried out by organizations such as Morocco's IRCAM (Amazigh), the Nigerien APT (Tuareg) [62], and the Académie berbère (Kabyle) [61]. The consonantal Tifinagh orthographies are not commonly used to write Kabyle. However, we transliterate Kabyle into a consonantal Tifinagh orthography to expand the incomplete literature on decoding into defective orthographies, which has primarily focused on Semitic languages. To our knowledge, no prior study has trained or decoded a speech model for a Berber language using Tifinagh inputs or a consonantal Tifinagh orthography.

We outline the fundamental differences between the Latin Kabyle orthography and the consonantal Tifinagh orthography: the first is that the Latin marks for gemination via digraphs, unlike the traditional Tifinagh. In some dialects, singletons are often spirantized as opposed to their geminated counterparts (e.g., "tt" from "t"). In the Latin orthography, these doubled consonants are phonemically "tense" and correlate with increased pronunciation length [63], register a fortis-lenis contrast that includes devoicing, and can form minimal pairs [64]. A consonantal Tifinagh orthography introduces additional heterophonic homography by graphically equating tense sounds with their non-tense counterparts.

The second fundamental difference is of vowel denotation. Although vowels are written in all contexts in Neo-Tifinagh orthographies, they are not marked save for word-final positions in the traditional Tifinagh orthographies [65, 66]. From the set of Tifinagh characters that may represent vowels, only "ⴰ" exclusively represents non-glide vowels (for "a," "ə"[1]) while "ⵓ" ("u") and "ⵉ" ("i") also represent semivowels ("w" and "j," respectively). These latter two graphemes are analogous to the *matres lectionis* of Semitic language scripts [67].

A final difference is that certain Tifinagh orthographies make use of ligatures that elide certain sequences of adjacent graphemes. The number of attested ligatures across the many varieties of traditional Tifinagh is vast [66], and most are not supported by Unicode.[2] We test the effect of ligatures by encoding those used in the Ahaggar orthography [65] as distinct characters in trial (1c) described in Section 5.

[1] We do not find attestations of "ⵓ" in the traditional Tifinagh orthographies described in [65]. We transliterate word-final "e" (primarily in loan-words) as "ⴰ,".

[2] https://www.unicode.org/charts/PDF/U2D30.pdf.

4 Approach

4.1 Mozilla CommonVoice

We use the original CommonVoice Kabyle corpus for all experiments.[3] The audio-transcript pairs come from Mozilla's CommonVoice crowdsourced initiative [50], which has collected data for over 54 languages at the time of writing. All corpora are released with train/dev/test subsets, and a unique speaker may appear in only a single set among each split. Most utterances are derived from Wikipedia, but some have been added by annotators through the language community's Pontoon page.[4] We removed special symbols and normalized Unicode characters of similar graphical appearance to ensure that characters intended to represent a single grapheme were treated as such.[5]

4.2 Mozilla DeepSpeech

For S2T model training, we use Mozilla's DeepSpeech pipeline, which is based on the DeepSpeech framework [68] and is maintained by a large community. After parameter tuning, we found that the default hyperparameters worked well. For all experiments, we used models of 1024 hidden units and trained for 50 epochs, with a learning rate of 0.0001 and a dropout of 0.3. We used batch sizes of 32, 16, and 16 for train, dev, and test sets, respectively. We used the default trigram settings for training the LM with KenLM [69] in our experiments.

4.3 Transliterator

To convert the Latin-script CommonVoice corpus to the Tifinagh orthographies in our experiments, we use the `Graph Transliterator` Python package[70]. This constructs a directed tree of ranked transition rules (e.g., **mm** -> ⵎ (not ⵎⵎ) because **mm** -> ⵎ ranks before **m** -> ⵎ) to convert between Latin and Berber orthographies. We write rules for two distinct defective orthographies modeled after [65]'s description of the Ahaggar variant of Tiginagh—one with ligatures and one without. In cases where multiple Unicode graphemes represent the same phonemes across Berber languages and orthographies (e.g., ⴽ, ⵗ), we opted to use the symbol closest to that described in [65]. Heterophonic homographs in the Latin corpus

[3] Accessed April 2020, 4th ed.

[4] https://pontoon.mozilla.org/projects/common-voice/.

[5] For example, ɛ and € were converted to ɛ (U+025B).

Table 1 Kabyle commonvoice data statistics

Split	Downloaded	Processed	Length
Train	37,056	35,715	35 hrs, 24 min
Dev	11,482	11,100	10 hrs, 52 min
Test	11,483	11,125	11 hrs, 42 min

Table 2 Normalization and transliteration examples

Original	Normalized	Tifinagh Transliteration
D tasnareft taserdasit i yettreṣṣin deg Lezzayer.	**d tasnareft taserdasit i yettreṣṣin deg lezzayer**	ⴷ ⵜⵙⵏⵔⴼⵜ ⵜⵙⵔⴷⵙⵜ ⵉ ⵉⵜⵔⵙⵏ ⴷⴳ ⵍⵣⵉⵔ
Teĉĉiḍ iles-ik waqila?	**teččiḍ iles ik waqila**	ⵜⵞⴹ ⵍⵙ ⴾ ⵓⵈⵍⴰ
Σerḍeγ-t-id ad yekkes lxiq, yeẓẓel iḍarren.	**ɛerḍeɣ t id ad yekkes lxiq yeẓẓel iḍarren**	ⵄⵔⴹⵗ ⵜ ⴷ ⴷ ⵉⴾⵙ ⵍⵆⵈ ⵉⵥⵍ ⴹⵔⵏ
Tawaγit d lmehna d-yeγdel ṭrad γef tmurt.	**taɣaɣit d lmeḥna d yeɣḍel ṭrad ɣef tmurt**	ⵜⵓⵗⵜ ⴷ ⵍⵎⵂⵏⴰ ⴷ ⵉⵗⴹⵍ ⵟⵔⴷ ⵗⴼ ⵜⵎⵔⵜ

remain as such in the transliterated Tifinagh (e.g. ‘d’ represents both ‘d’ and ‘ð’, and is transliterated as “V” and not the IRCAM “V.” All Kabyle phonemes that do not have distinct graphemes in the orthography described in [65] are represented with a corresponding Neo-Tifinagh symbol (e.g. -> ⵞ, -> ⵕ) (Tables 1 and 2).

4.4 Sequence Alignment

We sought to investigate which, and to what degree, phonemic classes are affected by different training orthographies. To facilitate this analysis, we required a tool to align the graphemic output sequences from the ASR systems, such that the aligned character pairs represented the audio data at the same time periods in the input data. We considered multiple techniques for matching the output sequences between the gold input and the inferences of the two models. One potential approach was to use an acoustic alignment model (e.g., the Montreal Forced Aligner [71] or DSAlign [72]), though this method risked substantial error propagation for our analysis. We also considered extracting time-aligned CTC model internals to understand the exact timesteps at which outputs were predicted with respect to the gold data. However, we felt that we could achieve the same results with Sound-Class-Based Phonetic Alignment (SCA) [73] with substantially reduced effort. Sound-Class-Based Phonetic Alignment (SCA) [73] was possible due to the relatively high degree of transparency or unambiguous correspondence between graphemes and phonemes [74] of the Kabyle Latin script. To implement SCA, we use the *prog_align* function contained in the `LingPy` package [75], which constructs a similarity matrix and applies a Neighbor-Joining algorithm (see [76]) to construct a guide tree to successively align phonemic sequences. A dynamic programming routine finds a least-cost path through the matrix to align the multiple sequences according to similar sound classes. We alter the default SCA sound class matrix values to ensure that Tifinagh *matres lectionis* graphemes ((‘j’ | ‘ⵉ’) => ‘I’, (‘w’ | ‘ⵓ’) => ‘Y’) could align with both vowels and semi-glides from the Latin gold transcripts. We find that this approach gives accurate alignment for phonemic sequences. We found no apparent errors after manually inspecting a thousand aligned phoneme pairs.[6]

5 Experimentation and Results

We present our result comparing S2T performance when training on orthographies of varying degrees of phonemic informativeness and analyzing phonemic confusion using sequence alignment techniques.

5.1 Experiments

First, we test the hypothesis that training and testing upon an orthography unmarked for vowels, as opposed to marked, yields lower ASR word error rates. Because the Tifinagh input only registers *matres lectionis* at the end of words, we expect that most intra-word vocalic signals are lost during the training process on the Tifinagh orthography compared to training on the Kabyle Latin script. Experiment 1 compares the effect of training and testing upon the Latin-based orthography and transliterated Tifinagh orthography in a set of trials listed in Table 4 (1a–c). In 1a, the Latin corpus is used for training and testing. The outputs were evaluated against Latin gold utterances in the test split. In 1b, we train in the same manner but test by applying a transliterator to convert the Latin test set into the consonantal Tifinagh orthography without ligatures. The corpus used to train the language model (LM) is composed of the transliterated utterances of the original corpus. In the third setup (1c), we repeat experiment 1b using a transliterator that models the ligatures described in Section 3. Examples of the ligatured Tifinagh are shown in Table 3.

Secondly, we test the hypothesis that learning from an orthography marked for vowels and decoding on an orthography unmarked for vowels result in lower word error rates compared to training and testing on either of the marked or unmarked orthographies alone. In experiment 2, we test the hypothesis that training on the

Table 3 Modelling unit experiment (1c) input example. Note: ⵆ and ⴹ are stand-in single-character substitutions for ligatures that are not represented in Unicode and are not graphically representative of the traditional graphemes for these ligatures

Non-ligatured	ⵏⵅⵊⵓⵔ	ⵉⵣⵅⵏ	ⵣⵟⵓ	ⴱⴷⵜ	ⵛⵏⵅⴰ	ⵂⵜ	ⵜⵗⵏ	ⴷⵉ	ⵜⵎⵏⵜⵍⵜ
Ligatured	!ⵊⵓⵔ	ⵉⵣⵅⵏ	ⵣⵟⵓ	ⴱⴷⵜ	ⵛ!ⴰ	ⵂⵜ	ⵜⵗⵏ	ⴷⵉ	ⵜⵎⵆⴹ

[6] https://github.com/berbertranslit/berbertranslit.

Table 4 The impact of orthography and language modeling. Group 1: trained and tested on the same orthography types. Group 2: Latin to Tifinagh transliteration at test time given a Latin model. Group 3: the same as Group 1 but without language modeling

Exp.	Train orthography	Transliteration	LM	Test orthography	CER (%)	WER (%)
1a	Latin	No	Yes	Latin	29.9	49.9
1b	Tifinagh	No	Yes	Tifinagh	35.8	57.9
1c	Tifinagh (ligatured)	No	Yes	Tifinagh (ligatured)	33.7	57.4
2	Latin	Yes	Yes	Tifinagh	29.7	47.4
3a	Latin	No	No	Latin	34.9	78.3
3b	Tifinagh	No	No	Tifinagh	38.8	77.9
3c	Latin	Yes	No	Tifinagh	35.6	72.1

plene (fully marked) Latin orthography and subsequently decoding into and testing against the defective Tifinagh orthography yield lower error rates compared to both training and testing on the Tifinagh orthography. We train all components on the Latin script and obtain Latin-script output for test utterances as in 1a. However, we then transliterate the output and test against gold utterances transliterated into Tifinagh, as in 1b. Because our main goal is to study the acoustic model and we do not want a small LM training corpus to negatively affect the experimental result, we build the LM in DeepSpeech on all train, dev, and test utterances of the normalized CommonVoice Kabyle Latin-script data for experiments 1 and 2.

Finally, we train the S2T model without an LM as a post-process to specifically understand the sensitivity of the neural speech component. Trials 3a–c replicate 1a–c but do not apply LM post-processing to help understand the effect of our interventions on the neural ASR component.

5.2 Results

We report the results of all three sets of trials in Table 4. 1a and 1b show that the original Kabyle input encoded in the plene Latin orthography yields lower error rates than when training and testing on the transliterated Tifinagh alone (CER: −5.9% and WER: −8%). However, this reduction is less pronounced when the ligatured Tifinagh orthography is used (1c) (CER: −3.8% and WER: −7.5%).

Trial 2 exhibits improved recognition when training on the Latin orthography and subsequently transliterating to and testing against Tifinagh. This arrangement reduces CER by 0.2% and WER by 2.5% with respect to trial 1a in which the plene orthography was used for both training and testing. Compared to training and testing in the defective orthography (1b), Trial 2 shows a 10.5% absolute decrease in WER and 6.1% absolute decrease in CER.

Trial 3 shows that, without the language model, the WER for training upon and testing against Latin orthography (3a) is greater than when using the Tifinagh orthography (3b) by 0.4%. However, the CER for the former procedure with respect to the latter is less by 3.9%, likely due to the increased difficulty of predicting more characters. Applying a Tifinagh tansliterator to the Latin trained model (3c) resulted in a WER reduction of 6.2 and 5.8% with respect to 3a and 3b. 3c exhibits an improved CER compared to the Tifinagh-only trial (3b) (−3.2%), although it is 0.7% higher when compared to the Latin-only trial (3a).

5.3 Phonemic Confusion Analysis

To understand the orthographies' effects on the speech model, we conduct an analysis by alignment between the gold utterances and the predictions from experiments 3b and 3c. This analysis is inspired by recent studies by Kong et al. [77], Alishahi et al. [78], and Belinkov et al. [6], to explore the nature of neural learning of phonemic information. More specifically, we use the `LingPy` [75] package to determine phone error rates as described in Sect. 4.4. We translate all graphemes of the gold utterances and their predicted counterparts into sequences of G2P IPA representations and tabulate phoneme class confusions using PHOIBLE's sound classes [79]. To understand the models' differential abilities in detecting spirantized consonants, we establish a "spirantized" feature that is attributed to the consonants "t," "d," "k," "g," and "b" that do not present in the contexts where non-continuant stops are the norm. We follow Chaker's description [80] of predictable Kabyle spirantized contexts to estimate this number across the corpus, as spirantized and non-spirantized consonants are commonly homographic in the Latin script. We modify the SCA model to ensure that *matris lectionis* characters are more easily aligned to their respective vowels in the gold Latin-text transcripts. Table 5 shows example aligned sentences produced by this procedure. By analyzing the aligned utterances, we tabulate estimated confusions between the gold and predicted alignments.

We count phonemic disagreements between the models as a proportion of gold target contexts of the aligned matching phoneme. To understand which model achieves better performance for word-final vowel recognition that is denoted in the Tifinagh orthography, we analyze the counts of all gold contexts in which vowels or semi-vowels appear (always word-finally) against the counts of aligned model inferences at these contexts. Table 6 shows that the model trained on the Latin orthography and subsequently transliterated (3c) achieves higher recognition of the pure vowel grapheme compared to the model trained on the unvowelled traditional Tifinagh (3b).

Table 7 compares the errors across several different phonemic classes. We do not consider the "continuant" and "delayedRelease" features, as the distinction between allophonic and phonemic fricativity is difficult to determine for Kabyle from graphemes alone. Although the PHOIBLE database includes these features as "syllabic," we tally counts for the "approximate," "sonorant," and "dorsal," and "periodic glottal source" features without "syllabic" phonemes so as to better analyze the contribution of non-syllabic features. McNemar's asymptotic test with continuity correction [81] affirms the significance of the difference between 3b and 3c ($P < 0.025$ for all features except the "geminate" feature).

Table 5 Alignment of the same sentence produced by different models in Table 4. * indicates a missing space in the alignment. $^+$ indicates Tifinagh-transliterated gold.

Group - train/decode	Raw		Alignment (in IPA representation)																	
3a - Latin/Latin	Gold	yuweḍ ɣer lebɣi s	j	u	w	ə	ḍ	ʁ	ə	r		l	ə	b	*	ʁ	*	i		s
	Pred	yuweḍ ɣaleb ɣ is	j	u	w	ə	ḍ	ʁ	a	-	*	l	ə	b		ʁ		i	*	s
3b - Tifinagh/Tifinagh	Gold$^+$	ⵉⵓⴹ ⵖⵔ ⵍⴱⵖⵉ ⵙ	j		w		ḍ	ʁ		r		l		b		ʁ		j		s
	Pred	ⵉⵓⴹ ⵖⵍⴱⵖ ⵙ	j		w		ḍ	ʁ		-	*	l		b		ʁ		-		s
3c - Latin/Tifinagh	Gold$^+$	ⵉⵓⴹ ⵖⵔ ⵍⴱⵖⵉ ⵙ	j		w		ḍ	ʁ		r		l		b	*	ʁ		j		s
	Pred	ⵉⵓⴹ ⵖⵍⴱ ⵖ ⵙ	j		w		ḍ	ʁ		-	*	l		b		ʁ		-		s

Table 6 Comparison of model performance for different word-final vowels. The columns represent phoneme pairs (Tifinagh grapheme : Latin IPA). Trial 3c shows considerably higher recognition of vowels.

	ⴰ : a/ə	ⵉ : i (j)	ⵓ : u/ (w)	All vowels
The number of word-final vowels in gold	7430	6557	1341	15,328
C_w: The portion (%) of all word-final phonemes	11.7%	10.3%	2.1%	13.0%
C_2: The portion (%) of C_w either 3b (x)or 3c is correct	23.7%	28.1%	30.4%	26.2%
C_3: Both 3b and 3c are incorrect	38.2%	46.8%	34.9%	41.6%
C_{3b}: The portion (%) of C_2 for which 3b is correct	18.5%	13.2%	13.7%	15.6%
C_{3c}: The portion (%) of C_2 for which 3c is correct	**81.5%**	**86.8%**	**86.3%**	**84.5%**

We bold the higher percentage between C3b and C3c

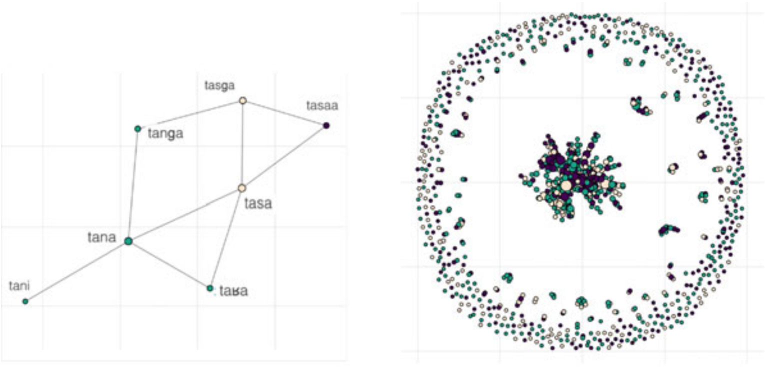
Table 7 Comparison of model performance for different phonemic features. C_p represents the portion (%) of G2P mappings the feature comprises the total number of G2P mappings in the corpus. See the definition of C_2, C_3, C_{3b}, and C_{3c} in Table 6. 3c is correct for more disagreements for all features except for the coronal, strident, and trill features. We use McNemar's asymptotic test with continuity correction [81] to test the null hypothesis that there is no difference between the performance of C_{3b} and C_{3c} with respect to different sound classes. χ_1^2 values are particularly high for voiced and syllabic phonemes. We bold the higher between C_{3b} and C_{3c} when $\chi_1^2 > 18.5$ (corresponding to $P = 0.001$)

	C_p	C_2	C_3	C_{3b}	C_{3c}	χ_1^2
Syllabic (vowels) (word-final)	6.1%	26.2%	41.6%	15.6%	**84.4%**	1902.8
Periodic glottal (voiced) (– syllabic)	36.4%	18.5%	29.2%	42.2%	**57.8%**	407.9
Dorsal (– syllabic)	11.8%	17.7%	28.7%	38.2%	**61.8%**	294.6
Sonorant (– syllabic)	24.0%	18.1%	26.2%	44.1%	**55.9%**	151.4
Nasal	11.5%	17.6%	24.1%	42.0%	**58.0%**	130.1
Spirantized stops (+ voiced)	3.3%	20.1%	34.3%	36.0%	**64.0%**	129.8
Continuant (– syllabic)	28.4%	17.1%	27.5%	45.5%	**54.5%**	98.5
Approximate (– syllabic)	12.6%	18.6%	28.2%	45.9%	**54.0%**	38.2
Consonants	53.1%	16.6%	29.7%	47.9%	**52.1%**	38.9
Non-spirantized stops (+ voiced)	0.5%	24.1%	24.1%	34.8%	**65.2%**	27.1
Labial	11.7%	16.1%	49.8%	35.0%	**65.0%**	26.3
Labiodental	1.5%	17.8%	30.3%	40.7%	**59.3%**	23.7
Spread glottis	0.4%	20.3%	47.1%	34.6%	**65.4%**	18.8
Retracted tongue root	2.1%	16.7%	60.0%	45.8%	54.2%	6.0
Lateral	4.3%	18.6%	30.0%	47.4%	52.6%	5.3
Non-spirantized stops (– voiced)	1.2%	22.7%	45.9%	46.4%	53.6%	3.3
Geminate	8.6%	9.0%	56.8%	49.6%	50.4%	0.13
Strident	8.0%	10.5%	33.5%	**53.8%**	46.2%	12.3
Coronal	37.1%	16.4%	29.1%	**51.9%**	48.1%	22.3
Trill	5.0%	16.3%	30.7%	**55.7%**	44.3%	26.0
Spirantized stops (– voiced)	6.7%	16.9%	20.7%	**70.1%**	29.9%	458.1

5.4 Phonological Network Analysis

We sought to understand the differences of the models based on the phonological similarity of their predicted vocabularies. Specifically, we first tokenize the vocabularies of the speech models' unique lexical tokens from their predicted output, as well as the vocabularies of the gold data as encoded in both Latin and the transliterated Tifinagh. We model nodes as surface-form tokens as they appear in their respective texts; we do not lemmatize outputs to study morphological effects on the phonological network as conducted by Shoemark et al. [55] as we are not aware of any available Kabyle lemmatizers. To construct a phonological network, we then assign an edge to any pair of nodes that are one edit away from each other (Fig. 1). That is, for any pair of tokens for which a single change, addition, or subtraction could cause both tokens to be the same token, an edge is formed. For example, for a given vocabulary set, "afət" and "aqət" are linked by an undirected edge, just as "afət" and "fət" are likewise assigned an edge. However, "aqət" and "fət" are not assigned an edge since they differ by an edit distance that is greater than one.

We analyze each gold corpus and speech model's inferred vocabulary as a self-contained phonological network and follow [53] and [55] in reporting common network statistics to characterize the properties of the graph. For each vocabulary network, we report the average degree, degree assortativity coefficient, error assortativity coefficient, and average shortest path length. We control for vocabulary size by computing the average statistics of 200 randomly sampled networks of 6000 nodes. We also obtain size-controlled modularity statistics for each network by (1) obtaining 3 randomly sampled networks of 4000 nodes for each gold and model phonological network, (2) conducting the Clauset–Newman–Moore modularity maximization algorithm to split and bin nodes into communities, and (3) computing the average modularity statistic given these communities. All statistics were obtained using the Python `networkx` package, v.2.6.3 [82] (Table 8).



(a) (b)

Fig. 1 Visualizations of phonological network structures on combined gold and ASR model vocabularies. Rendered with the Python package `bokeh`, v. 2.4.2 [84]. (**a**) Example module within a phonological network of Latin ASR gold and model output tokens. (**b**) Subgraph of the unionized network of Tifinagh-encoded gold and model vocabularies

Table 8 Descriptive statistics across gold vocabulary and model vocabulary phonological networks. **Bold** statistics are reported from averages across equally sized, randomly sampled subgraphs over multiple trials as reported in Sect. 5.4. **Group** denotes the vocabulary set analyzed, **Orthography** denotes the encoded orthography, **Size** denotes the length of the vocabulary, **% Giant Comp.** denotes the percentage of nodes in the largest component of the graph, **Avg. Degree** is the average degree of all nodes in the graph, **ASPL** is the average shortest path length of the giant component, **DAG** stands for the degree assortativity coefficient, **Err. AC** is the attribute assortativity coefficient of the binary feature of whether the node was outside of the gold vocabulary, and **Mod.** is the modularity of from Clauset–Newman–Moore community groupings. $^{+}$ indicates a transliterated vocabulary

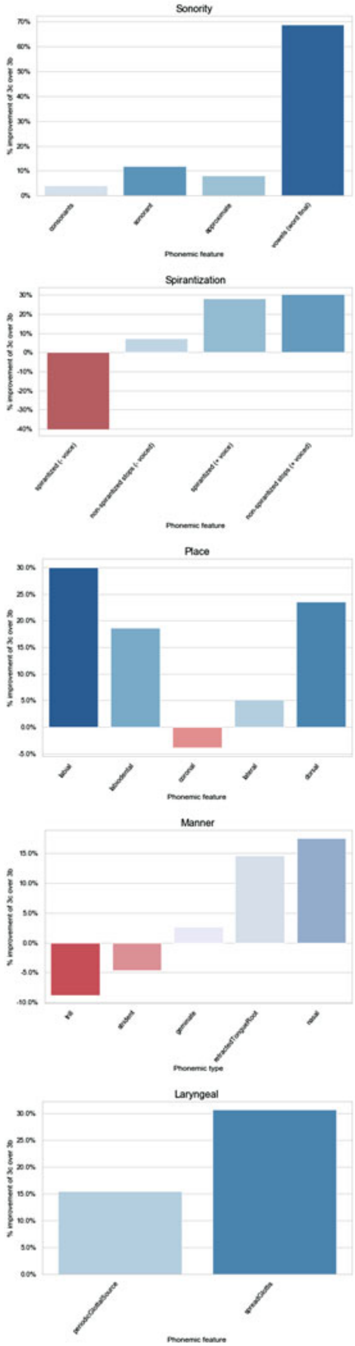
Group	Orthography	Size	% Giant comp.	Avg. deg.	ASPL	DAC	Err. AC	Mod.
Gold	Latin	12,860	48.9%	1.9 (**0.9**)	10.0 (**5.9**)	0.66 (**0.59**)	–	(**0.96**)
Gold	Tifinagh^{+}	9320	84.1%	8.4 (**5.4**)	5.5 (**5.8**)	0.54 (**0.53**)	–	(**0.66**)
3a	Latin	13,985	70.8%	6.4 (**2.7**)	6.5 (**4.3**)	0.56 (**0.54**)	0.11	(**0.79**)
3b	Tifinagh^{+}	8481	97.1%	22.6 (**16.0**)	3.9 (**4.1**)	0.40 (**0.40**)	0.10	(**0.53**)
3c	Tifinagh^{+}	7396	95.3%	19.8 (**16.0**)	4.1 (**4.2**)	0.45 (**0.45**)	0.16	(**0.54**)

6 Discussion

Performance when training on fully featured inputs (3c) to decode word-final vowels improves when compared 3b in which intra-word vowels are hidden from the model. The results suggest that sonorous and vocalized phonemes benefit more from model training on the voweled text. When only one model between 3b and 3c is correct, we see that "approximate," "sonorant," and "period glottal" phonemes exhibit comparatively high disagreement, surpassed only by the phonemes with positive "lateral" and "syllabic" features. The model may share information across these features, and in particular, voicing. All of these features record higher recognition rates in the case of 3c. While the difference in error rates for sonorous and voiced consonants between 3b and 3c does not exactly trend according to the sonority hierarchy [83], the number of disagreements between the models does follow this trend. These findings suggest that the model in 3c is leveraging correlates of sonority for phoneme recognition (Fig. 2).

A surprising finding was an improved ability of the 3b model in classifying non-tense/non-geminated phonemes modeled to be spirantized. This is interesting in that spirantized consonants are often homographic with non-spirantized consonants, so we are able to understand the variability of each model's recognition for a homograph corresponding to multiple sounds. The fact that non-spirantized consonants were better recognized by the Latin-trained 3c suggests that it is spirantization, not occlusivity, that is correlative with 3c's decreased performance in recognition of such phonemes. The reason for this disparity is unclear and may deserve additional investigation. It is notable that all contexts in which stops are modeled to be non-spirantized in our confusion analysis follow consonants. Model 3c, therefore, may better be able to recognize a non-spirantized consonants since its input would otherwise often include a vowel between the characters in question. However, the

Fig. 2 Comparison of the relative error difference between 3b and 3c



magnitude of the advantage of 3b over 3c in recognizing unvoiced spirantized stops is highly significant, especially in light of the fact that both voiced and unvoiced *non*-spirantized stops were more likely to be recognized by 3c when the two models disagreed (Table 7).

The models exhibit different rates of correctly detecting coronal and dorsal consonants. We hypothesize that this difference is a function of heterogeneous distributions in the context of vowels and geminate consonants. Further inspection of the data may also uncover imbalanced distributions between dorsal and coronal consonants with respect to word-internal vowels that are omitted in the consonantal orthography tested in this work. The improvement in the "spread glottis" feature between 3b and 3c is notable, though it is difficult to generalize given the low prevalence of graphemes representing phonemes possessing this feature. The other major orthographic difference of the Latin text compared to Tifinagh is that of marked gemination by means of digraphs. However, our results do not suggest significant differences in the models' abilities to correctly recognize these phonemes. The portion of alignments in which both models failed exhibits a wide range. Graphemes denoting the "retracted tongue root" feature were least likely to be correctly aligned. This feature, however, comprises a relatively low portion of the total number of alignments, and the models might simply not have enough instances to be able to detect the difference of this feature well. The observations we present may not hold for languages that observe some level of intra-word vowel denotation, for example, Arabic and other languages whose consonantal writing systems attest *matres lectionis* characters that present medially. To the authors' knowledge, there are no consonantal writing systems in widespread use that do not employ medial *matres lectionis* in the same way as consonantal Tifinagh. Nevertheless, the results characterize effects that may generalize to non-voweled orthographies as input to a non-Semitic language.

Our phonological network analysis reveals a stark contrast between the average degrees of the Latin and Tifinagh groups. As the phoneme vocabulary size is larger, the hyperlexica [60] of the Latin vocabulary is larger, and this effect outweighs the fact that the size of the Latin networks is larger to contribute to a high average degree. The average degrees of the ASR model output networks is greater by roughly a factor of 3 with respect to the gold networks. We believe this reflects the consolidation of choices elected by the models toward gold tokens, causing dense, closed structures to emanate from the gold signal. This interpretation is supported by a comparatively larger portion of the tokens in the ASR models' networks membership in the giant component of the graph. We note that the ASPL of the speech models' output is all roughly the same, whether encoded in Tifinagh or Latin outputs. However, it is generally shorter than those of the gold vocabularies', which structurally reflects a consolidation and narrowing in the choices to which the models converge as viable output emissions. We observe a higher average modularity statistic of Latin networks compared to that of the Tifinagh networks, reflecting the greater dispersion of highly connected modules in the network with a larger possible emission set. We find that the error assortativity coefficient trends with the models' error rates, which may reflect a tendency of erroneous tokens predicted by the higher performing ASR models to be more similar to each other.

7 Future Work

Our study experiments with the DeepSpeech architecture using a single set of hyperparameters for a single data set and language. Future work can investigate the interactions of model architectures, hyperparameters, data scales, G2P mappings, and statistics of orthographic informativeness on S2T performance. Additionally, future work could study the incidence of particular features of phonological features in modular communities in a phonological network context. An interesting direction would be to explore how other features specific to ASR modeling goals, such as a token's character edit distance from nearest neighbors, classification status as erroneous or licit, and its frequency in the gold corpora, vary with respect to specific network structures. We would also like to understand network statistics across different epoch checkpoints to observe how the network connectivity changes during the training process of the neural model.

8 Conclusion

Our study is the first to document S2T performance on Tifinagh inputs and shows that the choice of orthography may be consequential for S2T systems trained on graphemes. We amplify findings of prior studies focused on Semitic languages by showing that a Berber S2T model intended to output unvowelled graphemes benefits from training on fully featured inputs. Our research suggests that ensuring data inputs are fully featured would improve ASR model quality for languages that conventionally use consonantal orthographies, like Syriac, Hebrew, Persian, and Arabic vernaculars. Using phonological networks, we have also introduced a new way to analyze the similarities between ASR model outputs trained on different orthographies with respect to their respective gold vocabularies.

References

1. Turki, H., Adel, E., Daouda, T., Regragui, N.: A conventional orthography for maghrebi Arabic. In: Proceedings of the International Conference on Language Resources And Evaluation (LREC), Portoroz, Slovenia (2016)
2. Zitouni, I.: Natural Language Processing of Semitic Languages. Springer, Berlin (2014)
3. Jaffe, A.: Introduction: non-standard orthography and non-standard speech. J. Socioling. **4**, 497–513 (2000)

4. Cooper, E.: Text-to-Speech Synthesis Using Found Data for Low-Resource Languages. Columbia University (2019)
5. Davel, M., Barnard, E., Heerden, C., Hartmann, W., Karakos, D., Schwartz, R., Tsakalidis, S.: Exploring minimal pronunciation modeling for low resource languages. In: Sixteenth Annual Conference Of The International Speech Communication Association (2015)
6. Belinkov, Y., Ali, A., Glass, J.: Analyzing phonetic and graphemic representations in end-to-end automatic speech recognition (2019). Preprint ArXiv:1907.04224
7. Yu, X., Vu, N., Kuhn, J.: Ensemble self-training for low-resource languages: grapheme-to-phoneme conversion and morphological inflection. In: Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pp. 70–78 (2020)
8. Besacier, L., Barnard, E., Karpov, A., Schultz, T.: Automatic speech recognition for under-resourced languages: a survey. Speech Commun. **56**, 85–100 (2014)
9. Hu, K., Bruguier, A., Sainath, T., Prabhavalkar, R., Pundak, G.: Phoneme-based contextualization for cross-lingual speech recognition in end-to-end models (2019). Preprint ArXiv:1906.09292
10. Kubo, Y., Bacchiani, M.: Joint phoneme-grapheme model for end-to-end speech recognition. In: ICASSP 2020-2020 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP), pp. 6119-6123 (2020)
11. Chen, Z., Jain, M., Wang, Y., Seltzer, M., Fuegen, C.: Joint grapheme and phoneme embeddings for contextual end-to-end ASR. In: INTERSPEECH, pp. 3490–3494 (2019)
12. Rao, K., Peng, F., Sak, H., Beaufays, F.: Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4225–4229 (2015)
13. Jyothi, P., Hasegawa-Johnson, M.: Low-resource grapheme-to-phoneme conversion using recurrent neural networks. In: 2017 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP), pp. 5030–5034 (2017)
14. Arora, A., Gessler, L., Schneider, N.: Supervised Grapheme-to-Phoneme Conversion of Orthographic Schwas in Hindi and Punjabi (2020). Preprint ArXiv:2004.10353
15. Abbas, M., Asif, D.: Punjabi to ISO 15919 and Roman transliteration with phonetic rectification. In: ACM Transactions On Asian And Low-Resource Language Information Processing (TALLIP), vol. 19, pp. 1–20 (2020)
16. Hasegawa-Johnson, M., Goudeseune, C., Levow, G.: Fast transcription of speech in low-resource languages (2019). Preprint ArXiv:1909.07285
17. Yu, X., Vu, N., Kuhn, J.: Ensemble self-training for low-resource languages: Grapheme-to-phoneme conversion and morphological inflection. In: Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pp. 70–78 (2020). https://www.aclweb.org/anthology/2020.sigmorphon-1.5
18. Deri, A., Knight, K.: Grapheme-to-phoneme models for (almost) any language. In: Proceedings of the 54th Annual Meeting Of The Association For Computational Linguistics (Volume 1: Long Papers), pp. 399-408 (2016)
19. Le, D., Zhang, X., Zheng, W., Fügen, C., Zweig, G., Seltzer, M.: From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition. In: 2019 IEEE Automatic Speech Recognition And Understanding Workshop (ASRU), pp. 457–464 (2019)
20. Krug, A., Knaebel, R., Stober, S.: Neuron activation profiles for interpreting convolutional speech recognition models. In: NeurIPS Workshop on Interpretability and Robustness in Audio, Speech, and Language (IRASL) (2018)
21. Chrupała, G., Higy, B., Alishahi, A.: Analyzing analytical methods: The case of phonology in neural models of spoken language (2020). Preprint ArXiv:2004.07070
22. Alhanai, T.: Lexical and language modeling of diacritics and morphemes in Arabic automatic speech recognition. Massachusetts Institute of Technology (2014)
23. Alshayeji, M., Sultan, S., et al., Diacritics effect on arabic speech recognition. Arab. J. Sci. Eng. **44**, 9043–9056 (2019)

24. Al-Anzi, F., AbuZeina, D.: The effect of diacritization on Arabic speech recogntion. In: 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), pp. 1–5 (2017)
25. Daniels, P., Share, D.: Writing system variation and its consequences for reading and dyslexia. Sci. Stud. Read. **22**, 101–116 (2018)
26. Rafat, Y., Whitford, V., Joanisse, M., Mohaghegh, M., Swiderski, N., Cornwell, S., Valdivia, C., Fakoornia, N., Hafez, R., Nasrollahzadeh, P., et al.: First language orthography influences second language speech during reading: Evidence from highly proficient Korean-English bilinguals. In: Proceedings of the International Symposium on Monolingual and Bilingual Speech, pp. 100–107 (2019)
27. Law, J., De Vos, A., Vanderauwera, J., Wouters, J., Ghesquière, P., Vandermosten, M.: Grapheme-phoneme learning in an unknown orthography: A study in typical reading and dyslexic children. Front. Psychol. **9**, 1393 (2018)
28. Maroun, L., Ibrahim, R., Eviatar, Z.: Visual and orthographic processing in Arabic word recognition among dyslexic and typical readers. Writing Syst. Res., **11**(2), 142–158 (2019)
29. Eyben, F., Wöllmer, M., Schuller, B., Graves, A.: From speech to letters-using a novel neural network architecture for grapheme based ASR. In: 2009 IEEE Workshop On Automatic Speech Recognition & Understanding, pp. 376-380 (2009)
30. Wang, Y., Chen, X., Gales, M., Ragni, A., Wong, J.: Phonetic and graphemic systems for multi-genre broadcast transcription. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5899–5903 (2018)
31. Rao, K., Sak, H.: Multi-accent speech recognition with hierarchical grapheme based models. In: 2017 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP), pp. 4815–4819 (2017)
32. Li, B., Zhang, Y., Sainath, T., Wu, Y., Chan, W.: Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5621–5625 (2019)
33. Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang, X., Zhang, F., et al.: Others transformer-based acoustic modeling for hybrid speech recognition. In: ICASSP 2020-2020 IEEE International Conference On Acoustics, Speech And Signal Processing (ICASSP), pp. 6874–6878 (2020)
34. Schone, P.: Low-resource autodiacritization of abjads for speech keyword search. In: Ninth International Conference on Spoken Language Processing (2006)
35. Ananthakrishnan, S., Narayanan, S., Bangalore, S.: Automatic diacritization of Arabic transcripts for automatic speech recognition. In: Proceedings of the 4th International Conference on Natural Language Processing, pp. 47–54 (2005)
36. Alqahtani, S., Diab, M.: Investigating input and output units in diacritic restoration. In: 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 811–817 (2019)
37. Alqahtani, S., Mishra, A., Diab, M.: Efficient convolutional neural networks for diacritic restoration (2019). Preprint ArXiv:1912.06900
38. Darwish, K., Abdelali, A., Mubarak, H., Eldesouki, M.: Arabic diacritic recovery using a feature-rich biLSTM model (2020). Preprint ArXiv:2002.01207
39. Maroun, M., Hanley, J.: Diacritics improve comprehension of the Arabic script by providing access to the meanings of heterophonic homographs. Reading Writing **30**, 319–335 (2017)
40. Afify, M., Nguyen, L., Xiang, B., Abdou, S., Makhoul, J.: Recent progress in Arabic broadcast news transcription at BBN. INTERSPEECH. **5**, 1637–1640 (2005)
41. Alsharhan, E., Ramsay, A.: Improved Arabic speech recognition system through the automatic generation of fine-grained phonetic transcriptions. Inf. Process. Manag. **56**, 343–353 (2019)
42. Emond, J., Ramabhadran, B., Roark, B., Moreno, P., Ma, M.: Transliteration based approaches to improve code-switched speech recognition performance. In: 2018 IEEE Spoken Language Technology Workshop (SLT), pp. 448–455 (2018)
43. Le, N., Sadat, F.: Low-resource machine transliteration using recurrent neural networks of asian languages. In: Proceedings of the seventh Named Entities Workshop, pp. 95–100 (2018)

44. Cho, W., Kim, S., Kim, N.: Towards an efficient code-mixed grapheme-to-phoneme conversion in an agglutinative language: A case study on to-Korean Transliteration. In: Proceedings of the The 4th Workshop on Computational Approaches to Code Switching, pp. 65–70 (2020)
45. Ahmadi, S.: A rule-based Kurdish text transliteration system. ACM Trans. Asian Low-Resour. Lang. Inf. Process. **18**, 1–8 (2019)
46. Abbas, M., Asif, D.: Punjabi to ISO 15919 and Roman transliteration with phonetic rectification. ACM Trans. Asian Low-Resour. Lang. Inf. Process. **19** (2020). https://doi.org/10.1145/3359991
47. Sadouk, L., Gadi, T., Essoufi, E.: Handwritten tifinagh character recognition using deep learning architectures. In: Proceedings of the 1st International Conference on Internet of Things and Machine Learning, pp. 1–11 (2017)
48. Benaddy, M., El Meslouhi, O., Es-saady, Y., Kardouchi, M.: Handwritten tifinagh characters recognition using deep convolutional neural networks. Sensing Imaging **20**, 9 (2019)
49. Lyes, D., Leila, F., Hocine, T.: Building a pronunciation dictionary for the Kabyle language. In: International Conference on Speech and Computer, pp. 309–316 (2019)
50. Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F., Weber, G.: Common voice: A massively-multilingual speech corpus (2019). Preprint ArXiv:1912.06670
51. Zealouk, O., Hamidi, M., Satori, H., Satori, K.: Amazigh digits speech recognition system under noise car environment. In: Embedded Systems And Artificial Intelligence, pp. 421–428 (2020)
52. Luce, P., Pisoni, D.: Recognizing spoken words: the neighborhood activation model. Ear Hearing **19**, 1 (1998)
53. Vitevitch, M.S: What can graph theory tell us about word learning and lexical retrieval? J. Speech Lang. Hear. Res. **51**(2), 408–422 (2008)
54. Arbesman, S., Strogatz, S., Vitevitch, M.: The structure of phonological networks across multiple languages. Int. J. Bifurcat. Chaos **20**, 679–685 (2010)
55. Shoemark, P., Goldwater, S., Kirby, J., Sarkar, R.: Towards robust cross-linguistic comparisons of phonological networks. In: Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pp. 110–120 (2016)
56. Siew, C.: Community structure in the phonological network. Front. Psychol. **4**, 553 (2013)
57. Siew, C., Vitevitch, M.: An investigation of network growth principles in the phonological language network. J. Exper. Psychol. General **149**, 2376 (2020)
58. Siew, C., Vitevitch, M.: The phonographic language network: using network science to investigate the phonological and orthographic similarity structure of language. J. Exper. Psychol. General. **148**, 475 (2019)
59. Neergaard, K., Luo, J., Huang, C.: Phonological network fluency identifies phonological restructuring through mental search. Sci. Rep. **9**, 1–12 (2019)
60. Turnbull, R.: Graph-theoretic properties of the class of phonological neighbourhood networks. In: Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, pp. 233–240 (2021)
61. Souag, L.: Kabyle in Arabic script: A history without standardisation. In: Creating Standards, pp. 273. De Gruyter, Boston (2019)
62. Blanco, J.: Tifinagh & the IRCAM: Explorations in cursiveness and bicameralism in the tifinagh script. Unpublished Dissertation, University of Reading (2014)
63. Louali, N., Maddieson, I.: Phonological contrast and phonetic realization: The case of Berber stops. In: Proceedings of the 14th International Congress Of Phonetic Sciences, pp. 603–606 (1999)
64. Elias, A.: Kabyle "Double" Consonants: Long or Strong? UC Berkeley (2020). Retrieved from https://escholarship.org/uc/item/176203d
65. Elghamis, R.: Le tifinagh au Niger contemporain: Étude sur lécriture indigène des Touaregs. Unpublished PhD Thesis, Leiden: Universiteit Leiden (2011)
66. Savage, A.: Writing Tuareg–the three script options. Int. J. Sociol. Lang. **2008**, 5–13 (2008)

67. Posegay, N.: Connecting the dots: The shared phonological tradition in Syriac, Arabic, and Hebrew Vocalisation. In: Studies In Semitic Vocalisation And Reading Traditions, p. 191–226 (2020)
68. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al.: Deep speech: Scaling up end-to-end speech recognition (2014). Preprint ArXiv:1412.5567
69. Heafield, K., Pouzyrevsky, I., Clark, J., Koehn, P.: Scalable modified Kneser-Ney language model estimation. In: Proceedings of the 51st Annual Meeting Of The Association For Computational Linguistics (Volume 2: Short Papers), pp. 690-696 (2013). https://www.aclweb.org/anthology/P13-2121
70. Pue, A.: Graph transliterator: a graph-based transliteration tool. In: J. Open Source Softw. **4**(44), 1717 (2019). https://doi.org/10.21105/joss.01717
71. McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., Sonderegger, M.: Montreal forced aligner: trainable text-speech alignment using Kaldi. Interspeech **2017**, 498–502 (2017)
72. Tilmankamp, L.: DSAlign. GitHub Repository (2019). https://github.com/mozilla/DSAlign
73. List, J.: Sequence comparison in historical linguistics. Düsseldorf University Press (2014)
74. Marjou, X.: OTEANN: Estimating the transparency of orthographies with an artificial neural network. In: Proceedings of the Third Workshop On Computational Typology And Multilingual NLP, pp. 1–9 (2021). https://aclanthology.org/2021.sigtyp-1.1
75. List, J., Greenhill, S., Tresoldi, T., Forkel, R.: LingPy. A Python library for quantitative tasks in historical linguistics. Max Planck Institute for the Science of Human History (2019). http://lingpy.org
76. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biol. Evolut. **4**, 406–425 (1987)
77. Kong, X., Choi, J., Shattuck-Hufnagel, S.: Evaluating automatic speech recognition systems in comparison with human perception results using distinctive feature measures. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5810–5814 (2017)
78. Alishahi, A., Barking, M., Chrupała, G.: Encoding of phonology in a recurrent neural model of grounded speech (2017). Preprint ArXiv:1706.03815
79. Moran, S., McCloy, D. (Eds.): PHOIBLE 2.0. Max Planck Institute for the Science of Human History (2019). https://phoible.org/
80. Chaker, S.: Propositions pour la notation usuelle a base latine du Berbère. In: INALCO-CRB, p. e0245263 (1996)
81. Edwards, A.: Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. Psychometrika **13**, 185–187 (1948)
82. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using NetworkX. Los Alamos National Lab. (LANL), Los Alamos, NM (2008). https://github.com/networkx/networkx/releases/tag/networkx-2.6.3
83. Ladefoged, P., Johnson, K.: A Course in Phonetics. Nelson Education, Toronto (2014)
84. Bokeh Development Team: Bokeh: Python library for interactive visualization. (2022) https://bokeh.org/

ALP: An Arabic Linguistic Pipeline

Abed Alhakim Freihat, Gábor Bella, Mourad Abbas, Hamdy Mubarak, and Fausto Giunchiglia

Abstract This paper presents ALP, an entirely new linguistic pipeline for natural language processing of text in Modern Standard Arabic. In contrary to the conventional pipeline architecture, we solve common NLP operations of word segmentation, POS tagging, and named entity recognition as a single sequence labeling task. Based on this single component, we also introduce a new lemmatizer tool that combines machine-learning-based and dictionary-based approaches, the latter providing increased accuracy, robustness, and flexibility to the former. In addition, we present a base phrase chunking tool which is an essential tool in many NLP operations. The presented pipeline configuration results in a faster operation and is able to provide a solution to the challenges of processing Modern Standard Arabic, such as the rich morphology, agglutinative aspects, and lexical ambiguity due to the absence of short vowels.

1 Introduction

Natural language understanding tasks, such as information retrieval [1], word sense disambiguation [2, 3], question answering [4], or semantic search [5], are usually built on top of a set of basic NLP preprocessing operations. These operations are supposed to bring text to a more canonical form with dictionary words (lemmas) and named entities clearly identified. The precise solutions applied depend greatly on the language; however, state-of-the-art approaches typically involve a pipeline

A. A. Freihat (✉) · G. Bella · F. Giunchiglia
University of Trento, Trento, Italy
e-mail: abed.freihat@unitn.it; Gabor.Bella@unitn.it; fausto.giunchiglia@unitn.it

M. Abbas
High Council of Arabic Language, Algiers, Algeria

H. Mubarak
Hamad Bin Khalifa University, Ar-Rayyan, Qatar
e-mail: hmubarak@hbku.edu.qa

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_4

of components, such as a part-of-speech tagger, a morphological analyzer, a lemmatizer, and a named entity recognizer (NER). Compared to English, both lemmatization and NER are harder for Arabic text: for the former because of the inflectional complexity and ambiguity inherent to written language and for the latter mainly because Arabic does not mark named entities by capitalization.

There has been extensive research on each of the tasks mentioned above. In the case of Arabic POS tagging, the approaches are typically based on statistical classifiers such as SVM [6, 7], sometimes combined with rule-based methods [8] or with a morphological analyzer [9–11]. The idea of POS tagging applied to unsegmented words has been investigated in [10] and in [12].

For NER, several solutions and tools have been reported. They can be classified as rule-based systems such as the approach presented in [13], machine-learning-based ones such as [14, 15], and hybrid systems such as [16]. The correlation between NER and POS tagging is illustrated in [17].

For Arabic lemmatization, while several approaches were proposed, few tools are actually available. Existing tools typically combine multiple techniques to achieve efficient lemmatization. The *Alkhalil* lemmatizer [18] first applies morpho-syntactic analysis to the input sentence in order to generate all potential word surface forms. Then, among these, only one form is selected per word using a technique based on hidden Markov models. The accuracy of the tool is reported to be about 94%. Another lemmatizer is MADAMIRA [19] which relies on preliminary morphological analysis on the input word that outputs a list of possible analyses. As a second step, it predicts the correct lemma using language models. The accuracy of the tool is 96.6%. The FARASA lemmatizer [20] uses a dictionary of words and their diacritizations ordered according to their number of occurrences. The accuracy reported for FARASA is 97.32%. Besides these tools, there are other proposed approaches: for example, [21] proposes a pattern-based approach, while [22] and [23] present rule-based solutions.

For Arabic chunking tools, the only available research on phrase chunking is the work done by Mona Diab who introduced a chunking tool as a component of MADAMIRA. The adopted approach and details about the tools are described in [24]. The reported accuracy of the tools is 96.33%. In terms of overall NLP pipeline architecture, most existing solutions perform the aforementioned tasks as a cascade of several processing steps. For example, POS tagging in FARASA [20, 25] and in MADAMIRA supposes that word segmentation has been done as a previous step. Segmentation, in turn, relies on further preprocessing tasks such as morphological analysis in MADAMIRA.

Likewise, NER and lemmatization are often implemented as separate downstream tasks that rely on the results of POS tagging, base phrase chunking, and morphological analysis. In several Arabic pipelines in the literature [7], however, upstream tasks such as POS tagging are implemented in a coarse-grained manner, which amounts to delegating the resolution of certain cases of ambiguity to downstream components. For example, by using single VERB and PART tags, the POS tagger in [6] avoids challenging ambiguities in Arabic verbs and particles, respectively. Consequently, an additional downstream component is needed for morphological disambiguation, e.g., to find out whether تعرف is an imperative (تَعَرَّفْ/*recognize*), past (تَعَرَّفَ/*recognized*), or present tense verb (تَعْرِفُ/*you know or she knows*); whether the noun سحب is singular (in which case it means *withdrawal*) or plural (meaning *clouds*); or whether أن is an accusative (أَنَّ) or a subordinate particle (أَنْ).

Good-quality Arabic annotated corpora for machine learning are few and far between. The *Penn Arabic Treebank* [26] is a non-free, half-a-million-word annotated corpus destined for POS tagging and syntactic parsing, upon which a large number of research results are based. The KALIMAT corpus,[1] while freely available, is a silver standard corpus on which a POS tagging accuracy of 96% was reported [27].

In this paper, we present an entirely new linguistic pipeline for natural language processing of text in Modern Standard Arabic. Our goal was to provide an open-source tool that simultaneously maximizes accuracy, speed of execution, as well as the resolution of difficult cases of ambiguity within the Arabic text. This way, NLP tasks downstream of ALP are also expected to work in a more accurate and robust manner as they need to deal with less amount of ambiguity.

One of the general design principles we used to achieve these goals was to reduce the number of individual NLP components within the pipeline. Thus, ALP consists of just two components: the first one is a *preprocessor* that performs word segmentation, POS tagging, and named entity recognition as a single processing task, without any other auxiliary processing tool [28]. The second component uses these results to perform lemmatization [29].

In an effort to improve accuracy with respect to state-of-the-art tools, we decided to implement a solution that is independent from implicit NLP design choices embedded in the annotations of existing corpora. Thus, we hand-annotated an over two-million-token corpus that forms the basis of ALP. The pipeline can be tested online[2] and is freely available for research upon request.

The rest of the paper is organized as follows: Sect. 2 presents the main cases of lexical and morphological ambiguity in Arabic that ALP was designed to tackle. Section 3 introduces the general architecture of ALP and its components. Section 4 provides details and the rationale behind the tag sets we used for annotation. Section 5 presents the annotation methods we used for the two-million-token corpus. Section 6 provides evaluation results on ALP, and Sect. 7 reflects on future work.

[1] https://sourceforge.net/projects/kalimat/.

[2] http://www.arabicnlp.pro/alp/.

2 Ambiguity in Arabic

To illustrate the challenging cases that low-level NLP tasks such as word segmentation or lemmatization typically need to solve, in the following, we list some common examples of lexical and morphological ambiguity in Arabic.

2.1 Ambiguity in Word Segmentation

Certain words can be segmented into morphemes in more than one valid way. In such cases, the correct segmentation can only be determined in context. In Table 1, we list some common examples of ambiguity that occur at the segmentation level.

2.2 Ambiguity in POS Tagging

While correct segmentation decreases the ambiguity in Arabic text, polysemy and the lack of short vowels result in morphemes having multiple meanings with distinct parts of speech. In Table 2, we show some examples of this kind.

Table 1 Ambiguity examples at the segmentation level

Ambiguity	Example
Nouns vs conjunction+pronoun	وهن/*weakness* vs وهن/*and they (feminine)*
Noun vs conjunction+verb	وحل/*mud* vs وحل/*and (he) solved*
Noun vs conjunction+noun	وصفة/*receipt* vs وصفة/*and (a) character*
Noun vs singular noun+pronoun	كتابي/*two books (in genitive)* vs كتابي/*my book*
Noun vs preposition+noun	لسعة/*sting* vs لسعة/*for capacity*
Proper noun vs preposition+noun	بعقوبة/*a city in Iraq* vs بعقوبة/*with punishment*
Proper noun vs conjunction+noun	وهران/*a city in Algeria* vs وهران/*and two cats*
Proper noun vs definite article+noun	الباب/*a city in Syria* vs الباب/*the door*
Noun vs interrogative particle+negation particle	ألم/*pain* vs ألم/*did I not*
Adjective vs noun+pronoun	جانبي/*lateral* vs جانبي/*my side*
Adjective vs preposition+noun	بحرية/*nautical* vs بحرية/*to freedom*
Verb vs conjunction+pronoun	فهم/*(he) understood* vs فهم/*and they (masculine)*
Verb vs conjunction+verb	وفر/*saved* vs وفر/*and (he) escaped*
Verb vs verb+pronoun	علمنا/*we knew* vs علمنا/*(he) taught us*
Verb vs interrogative particle+verb	أتذكر/*(I) remember* vs أتذكر/*do (you) remember*

Table 2 Ambiguity examples at the POS tagging level

Ambiguity	Example
Verb vs noun	حمل/*carried* vs حمل/*carrying*
Verb vs comparative	أثقل/*overburdened* vs أثقل/*heavier*
Verb vs adjective	سهل/*facilitate* vs سهل/*easy*
Verb/noun vs particle	لم/*gathered* vs لم/*not*
Verb vs number	تسع/*expanded* vs تسع/*nine*
Verb vs proper noun	طلعت/*rose* vs طلعت/*Talat*
Noun vs number	سبع/*lion* vs سبع/*seven*
Noun vs proper noun	إحسان/*philanthropy* vs إحسان/*Ehsan*
Adjective vs noun	نوعية/*qualitative* vs نوعية/*quality*
Adjective vs proper noun	جميل/*nice* vs جميل/*Jamil*
Interrogative particle vs relative pronoun	According to their position in the sentence
Particle ambiguity in أن	أنْ/*subordinating* vs أنَّ/*accusative*
Particle ambiguity in إن	إنْ/*conditional* vs إنَّ/*accusative*
Particle ambiguity in لم	لمْ/*negation* vs لمَ/*interrogative*
Particle ambiguity in ما	ما/*negation* vs ما/*interrogative*

Even with correct segmentation and POS tagging, challenging cases of ambiguity still remain on the level of fine-grained POS tags, mostly due to MSA words overwhelmingly being written without diacritics. In the following, we list some examples of ambiguity with which we deal on the fine-grained level.

2.2.1 Verb Ambiguities: Passive vs Active Voice

Many verbs in Arabic have the same form in the active or passive voice cases. Verbs like سجّل/*reported or has been reported* can be only through the context disambiguated.

2.2.2 Verb Ambiguities: Past vs Present Tense

The same verb word form that denotes a verb in first-person singular present denotes (another) verb in third-person singular masculine past. Consider, for example, the verb أجمل which can be أُجْمِلُ/*(I) illustrate* can also be أَجْمَلَ/*(he) illustrated.*

A third-person singular feminine present verb form denotes (another) verb in third-person singular masculine past. Consider, for example, the verb تحمل which can be تَحْمِل/*(she) carries* can be also تَحَمَّلَ/*(he) sustained.*

2.2.3 Verb Ambiguities: Imperative

The imperative verb form (second person singular masculine) can be read as a past tense verb (third person singular masculine). For example, the verb تعرف which may be an imperative verb (تَعَرَّفْ/*recognize*) or a past tense verb (تَعَرَّفَ/*(he) recognized*).

The imperative verb form (second person plural masculine) can be read as a past tense verb (third person plural masculine). It can be also a present tense verb (third person plural masculine). For example, the verb تعرفوا which may be an imperative verb (تَعَرَّفُوا/*recognize*), a past tense verb (تَعَرَّفُوا/*(they) recognized*), or a present tense verb in cases like كَيْ تَعْرِفُوا/*so that (you) know.*

The imperative verb form (second person singular feminine) can be read as a present tense verb (second person singular feminine), after some particles. For example, the same form تعرفي can be an imperative verb (second person singular feminine like in (تَعَرَّفِي/*recognize*) or a present tense verb (second person singular feminine) after subordination particles such as in the case (كَيْ تَعْرِفِي/*so that you know*).

2.2.4 Noun Ambiguities: Singular vs Plural

In Arabic, there are several word forms that denote (different) singular and plural nouns. For example, the word سحب denotes the singular noun سَحْب/*dragging* and the plural noun سُحُب/*clouds.*

2.2.5 Noun Ambiguities: Dual vs Singular

The ا accusative case ending in Arabic leads to dual singular ambiguity. For example, the word form كتابا may be read as singular noun كِتَاباً/*one book* or dual كِتَابَا/*two books* (in genitive dual cases such as كِتَابَا العُلُوم).

2.2.6 Noun Ambiguities: Dual vs Plural

Dual form nouns and masculine plural noun in general are ambiguous. For example, the word مؤمنين can be read as مؤمِنَيْن/*dual form* or as مؤمِنِيْن/masculine plural form.

2.2.7 Noun Ambiguities: Feminine vs Masculine Singular

There are cases in which the same word form denotes singular but with different gender. For example, the word قدم can be feminine قَدَم/*foot* or masculine قِدَم/*antiquity*.

2.3 Ambiguity in Named Entity Recognition

Besides the ambiguity cases that we have presented in the previous section, we present below two examples of ambiguity related to NER, referring the reader to [30] for a more detailed treatise on the matter.

2.3.1 Inherent Ambiguity in Named Entities

It is possible for a word or a sequence of words to denote named entities that belong to different classes. For example, واشنطن denotes both a person and location. It is also frequent that organizations and establishments are named after person names. For example, جامعة الملك عبد الله للعلوم التقنية/King Abdullah University of Science and Technology.

2.3.2 Ellipses

Ellipses (omitting parts of nominal phrases and entity names) contribute to the high ambiguity of natural languages. Considering the lack of orthographic features in Arabic, ellipses increase the ambiguity. For example, a text about البحر الأبيض المتوسط/*The Mediterranean Sea* mentions it explicitly at the beginning of the text. After that, it may omit البحر الأبيض/the White Sea and refers to it by المتوسط/*the Mediterranean*. This word is used mostly as an adjective (which means *the average*), and there is no orthographic triggers that may disambiguate the entity from the adjective token.

2.4 Ambiguity in Lemmatization

Lexical ambiguity is pervasive in conventional written Arabic due to the absence of short vowels. For example, the past tense verb صرت could be vocalized as صِرْتُ with the corresponding lemma صَارَ/*become* or as صَرًّ with the corresponding lemma صَرَّ/*grit*. Nouns can also be ambiguous: the word سبل can be read as سُبُل/*ways* or سَبَل/ears.

As choosing the correct lemma is ultimately a word sense disambiguation problem, such cases put considerable stress on the quality of lemmatization. Tools that are capable of outputting multiple solutions in an order of preference are in this sense more robust as they potentially allow the disambiguation problem to be delayed to subsequent syntactic or semantic processing steps.

2.5 Ambiguity in Phrase Chunking

Ambiguity in phrase chunking is related to ambiguity at POS tagging and named entity recognition ambiguities. For example, the two tokens محمود الذهب could be read in two different ways. The first can be read as one nominal phrase محمود/*Mahmud* الذهب/*Althahab* (family name) in a sentence like رأيت محمود الذهب/*I saw Mahmud Althahab*. They can also be read as two separate nominal phrases محمود/*Mahmud* الذهب/*the gold* as in the following sentence: باع محمود الذهب/*Mahmud sold the gold*.

We have also the named entity boundary ambiguity problem. For example, the two nouns محمد/*Mohammed* and محمود/*Mahmud* can constitute two different noun sequences. While the sentence رأى محمد محمود/*Mohammed saw Mahmud* contains two nominal phrases, the sentence غادر محمد محمود//*Mohammed Mahmud left* contains only one. We believe that solving this kind of ambiguity needs extending the verb tags with the verb *transitivity/intransitive* information which is planned as a future work.

3 Pipeline Architecture

The pipeline specific to our method is shown in Fig. 1 and is composed of the following main steps:

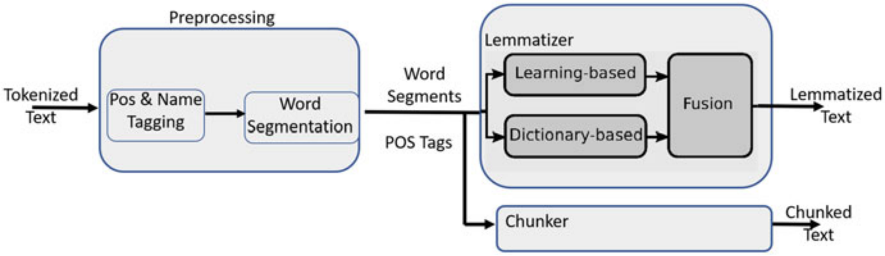


Fig. 1 The high-level NLP pipeline architecture for lemmatization and chunking

1. *Prepossessing:* taking white-space tokenized Arabic text in input, we pre-annotate the text through the following operations:
   (a) *POS and name tagging:* tokens are annotated by a machine-learning-based sequence labeler that outputs POS, named entity, and word segment tags.
   (b) *Word segmentation:* using the POS output, cliticized words are segmented into a proclitic, a base word, and an enclitic, making the subsequent lemmatization step simpler.
2. *Lemmatization:* the segmented and pre-annotated text is fed into the following lemmatizer components:
   (a) *Dictionary-based lemmatizer:* words are lemmatized through dictionary lookup.
   (b) *Machine-learning-based lemmatizer:* words are lemmatized by a trained machine learning lemmatizer.
   (c) *Fusion:* the outputs of the two lemmatizers are combined into a single output.
3. *Chunking:* the input of the chunker is similar to the input of the lemmatizer. The output is a list of base chunks.

The input of the annotator is expected to be UTF-8-encoded, white-space tokenized but otherwise unannotated text in Modern Standard Arabic. We are also supposing that sentences have been previously split by the usual sentence end markers (".", "!", "?", "…") and newlines.

3.1 Preprocessing: POS, NER, and Word Segment Tagging

The first component of ALP is a single *preprocessor* component that tackles three conventionally distinct NLP tasks: part-of-speech tagging, named entity tagging, and word segmentation. The common underlying goal of these preprocessing tasks is to reduce the ambiguity of words by extracting information from their morphology and context. Consequently, our combined tagger uses a machine-learning-based sequence labeling approach.

Performing the three operations in a single step presents several advantages:

- It is faster to execute than running several machine learning models in series.
- It is easier to reuse as part of a natural language understanding application.
- It does not suffer from the problem of cumulative errors that are inherent to solutions that solve the same tasks in series.

3.1.1 POS Tagging

The training corpus of the ALP preprocessor was annotated with fine-grained POS tags that, besides the high-level category, also provide number, gender, tense, and other information (see Sect. 4.1 for details). This detailed output can be effectively used by downstream components—such as the ALP lemmatizer—for solving a large number of cases of lexical ambiguity due to missing vocalization. There remain, however, some cases of word sense disambiguation that POS tagging alone cannot deal with, such as transitive/ditransitive verb ambiguity. For example, verbs such as علم (عَلِمَ/*knew* or عَلَّمَ/*taught*) remain ambiguous according to our current annotation tag set.

3.1.2 Named Entity Recognition

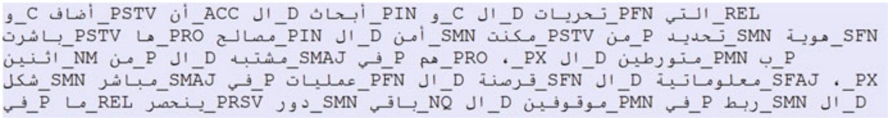
The ALP preprocessor does not mark named entities as nouns or proper nouns; rather, it annotates them directly with named entity tags (see Sect. 4.3 for details). This way the need for a separate NER component is avoided.

Based on the NER tags output by the ALP annotator, identifying the start and end of a named entity is a trivial task. Then through subsequent word segmentation, clitics can be removed from the entity and the canonical name obtained.

3.1.3 Word Segmentation

Word segmentation serves a double goal: to reduce the amount of distinct word forms, resulting in smaller and more robust lemmatizers, and to reduce lexical ambiguity due to multiple possible interpretations. For example, segmentation reduces the number of possible word forms of the lemma قلم from several hundreds of cliticized nouns { قلم, قلمي,بقلمي,وبقلمي, القلم,. . .} to six forms { قلم,قلمان,قلما,قلمين,قلمي,أقلام} only. On the other hand, word segmentation reduces the lexical ambiguity in cases such as لسعة which may be single word (*sting*) or a cliticized word (*for capacity*).

The actual segmentation of words is executed based on the clitic tags provided by the POS tagger. The input of the method is a word and its corresponding tag. The



```
التي_REL تحريات_PFN ال_D و_C أبحاث_PIN ال_D أن_ACC أضاف_PSTV و_C
هوية_SFN تحديد_SMN من_P مكنت_PSTV أمن_SMN ال_D مصالح_PIN ها_PRO باشرت_PSTV
ب_P متورطين_PMN ال_D ،_PX هم_PRO في_P مشتبه_SMAJ ال_D من_P اثنين_NM
،_PX معلوماتية_SFAJ ال_D قرصنة_SFN ال_D عمليات_PFN في_P مباشر_SMAJ شكل_SMN
ال_D ربط_SMN في_P موقوفين_PMN ال_D باقي_NQ دور_SMN ينحصر_PRSV ما_REL في_P
```

Fig. 2 An example output of the word segmenter

output is a list of tokens that correspond to the PROCLITIC, the BASETAG, and the ENCLITIC tags. Given that clitics are linguistically determined, segmentation becomes a simple string splitting task. An example of the output of a segmentation tool we implemented is shown in Fig. 2.

3.2 Lemmatization

Lemmatization returns the canonical (dictionary) forms of inflected words of a text. As such, it is a frequent upstream processing step before any analysis of lexical semantics (the meaning of words). For morphologically rich languages, lemmatization is usually a complex task, e.g., due to the presence of irregular cases. In Arabic, this includes broken plurals and irregular verbs. State-of-the-art lemmatizers typically apply finite-state transducers and/or lemmatization dictionaries to such cases, such as *AraComLex* [21] or the *OpenNLP lemmatizer*.[3] For regular cases, such as singular nouns and adjectives, regular plural nouns, and regular verbs, lemmatization is reduced to a straightforward task of *normalization* that removes inflectional prefixes and suffixes. For example, the verb form يحملوا is normalized into the lemma حمل. In the case of singular and regular plural nouns, it is sufficient to remove the plural suffixes and case endings. For example, the lemma of the dual noun form ولدين is ولد.

The ALP lemmatizer operates over an input pre-annotated by previous preprocessing steps, taking the segmented and POS-annotated text in input. It is built from the following components:

1. *A machine-learning-based lemmatizer:* words are lemmatized by a supervised classifier.
2. *A dictionary-based lemmatizer:* words are lemmatized through dictionary lookup.
3. *A fusion lemmatizer:* the outputs of the two lemmatizer components above are combined into a single output.

Both the learning-based and the dictionary-based lemmatizers were implemented using the Apache OpenNLP toolkit.[4]

[3] https://opennlp.apache.org/docs/1.8.4/manual/opennlp.html.

[4] http://opennlp.apache.org/docs/1.9.0/manual/opennlp.html#tools.cli.lemmatizer.

3.2.1 Learning-Based Lemmatizer

The principal component of our lemmatization approach is a machine-learning-based classifier. It takes word segments and their corresponding POS tag in input, also taking context (words and tags) into account. The learning-based approach is justified by the inherent ambiguity of diacritic-free Arabic words whose meanings are typically deduced, by humans and machines alike, from context. While the preliminary POS tagging resolves a great deal of ambiguity, some cases still remain such as the verb form يكون which may represent the verb كان or the verb كون.

3.2.2 Dictionary-Based Lemmatizer

A downside of learning-based lemmatization is that more rare and exceptional cases, such as أسنة (*spears*), may not be covered by its training corpus, which leads to lemmatization mistakes. The addition of new cases requires the re-training of the classifier. Another inconvenience is that classifiers—such as OpenNLP that we used—typically commit on a single output result, which may or may not be correct. In case of such ambiguity, from the full set of possible lemmas, further NLP processing steps may be able to provide correct results based on, e.g., syntactic or semantic analysis. In order to support these cases, we complement the learning-based lemmatizer by a dictionary-based one. The dictionary lemmatizer can be run independently, but we also provide a simple fusion method that combines the results of the two lemmatizers as described below.

3.2.3 Fusion Lemmatizer

While the learning-based lemmatizer outputs for each word a single candidate lemma, from the dictionary, multiple solutions could be retrieved even for a single part of speech (e.g., the verb form يسلم can be a verb form of the verb سلم *gave up* or أسلم *converted to Islam*). The goal of the simple fusion component is to produce a final result from these solutions. The final output is a list of one or more lemmas in a decreasing order of confidence.

The idea underlying the fusion method is that we *usually* trust the dictionary to be capable of providing a correct solution space (a small set of possible lemmas), while we *usually* trust the classifier to return the most likely lemma from the previous set. However, in the case of out-of-corpus words, the classifier may return incorrect results extrapolated from similar examples, such as returning the lemma تهمن for the word form يتهمن. Thus, whenever a lemma is returned by the classifier that is not included in the dictionary, it will still be included as a solution but with a lower confidence.

Accordingly, our simple fusion method is as follows. We take as input the results output by the two lemmatizers, namely, $L_{\mathrm{DIC}} = \{l_1, \ldots, l_n\}$ for the dictionary-based one and $L_{\mathrm{CL}} = \{l\}$ for the classifier-based one, and output L_{F}, the fusion result. We start by comparing the results of the two lemmatizers:

1. If $|L_{\mathrm{DIC}}| = 1$ and $l_1 = l$, i.e., the outputs are identical, then the solution is trivial, and we return either output and we are done: $L_{\mathrm{F}} = \{l\}$.
2. Otherwise, two further cases are distinguished:

   (a) If $l \in L_{\mathrm{DIC}}$, that is, the dictionary contains the classification output, then we prioritize the result of the classifier by making it first (i.e., the preferred lemma): $L_{\mathrm{F}} = \{l, l_1, \ldots, l_n\}$.

   (b) Otherwise, we add the classifier result as the last element: $L_{\mathrm{F}} = \{l_1, \ldots, l_n, l\}$.

3.3 Base Chunker

Base chunker splits sentences into groups of base chunks. These chunks in turn form sentence phrases or may be parts of sentence phrases. It operates over an input pre-annotated by previous preprocessing steps, taking the segmented and POS-annotated text in input. The output of the component is a list of chunks. These chunks may be one of the following base phrases:

1. Nominal phrases: Base nominal phrases are the longest possible sequence of adjacent words that constitute a phrase which does not contain coordinations or relative clauses. The length of base nominal phrases may range from one token to ten tokens or more. An example for such long phrases is النوم المزمنة هارتموت رينتمايسترعضو الرابطة الألمانية العامة لاضطرابات/ *member of the German General Association of Chronic Sleep Disorders Hartmut Rintmäster.*
2. Verbal phrases: Verbal phrases are phrases that contain a verb, two verbs, and an optional nominal phrase, prepositional phrase, or adverbial phrase complement. The verb part of the phrase may be also preceded by a particle such as لا in the following example: لا يزالون يعدون الترتيبات/*are still making arrangements.* Notice that the boundary of verbal phrases is implicit in case of the object complement. This means only the verbal part of the phrase will be annotated as a verbal phrase. The complement part will keep its original annotation.
3. Predicative adjective phrases: A predicate adjective is a predicate that follows a nominal phrase. It may range from one to several adjectives such as the phrases مناسب/*appropriate* and غير مناسب/*inappropriate*.
4. Prepositional phrases: Prepositional phrases are phrases that contain a preposition followed by a nominal phrase such as في المدسة/*at school.*

5. Adverbial phrases: Adverbial phrases are modifiers that follow an adjective like جدا in جميل جدا or a verb such as أسبوعيا in يسافر أسبوعيا/*travels weekly*. In the case of verbs, it is not necessary that the adverb follows the verb directly. It is possible to find intermediate phrases between the verb and its modifiers. Consider, for example, the phrase يسافر إلى فرنسا أسبوعيا/*travels to France weekly*.

4 Annotation Schema

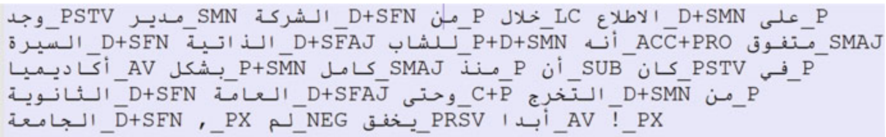
This section presents the tag set used for the preprocessing component of the pipeline and the design choices behind it.

We annotated a single large training corpus with complex and fine-grained tags that encode information with respect to part of speech, word segments, and named entities. On a high level, the tag set is composed as follows:

```
<TAG>       ::= <PREFIX> <BASETAG> <POSTFIX>
<BASETAG>   ::= <POSTAG> | <NERTAG>
<PREFIX>    ::= <PREFIX> | <PROCLITIC> "+" | ""
<POSTFIX>   ::= <POSTFIX> | "+" <ENCLITIC> | ""
```

A tag is thus composed of a mandatory base tag and of zero or more (i.e., optional) proclitics and enclitics concatenated with the "+" sign indicating word segments. A base tag, in turn, is either a POS tag or a NER tag, but not both (in other words, we do not annotate named entities by part of speech). For example, the full tag of the word وبكتابه is a noun tag preceded by two proclitic tags (conjunction and preposition) and followed by a pronoun enclitic tag. The choice of our base and clitic tags was inspired by the coarse-grained tags used in MADA 2.32 [7] and 3.0 [19], as well as by the more fine-grained tags used in the Qur'an Corpus [31]. For compatibility with other NLP tools, mapping our tags to MADA 2.32, MADA 3.0, and Stanford [32] or any other tag sets is straightforward.

In Fig. 3, we provide an example of a complete annotated sentence.



```
وجد_PSTV مدير_SMN الشركة_D+SFN من_P خلال_LC الاطلاع_D+SMN على_P
السيرة_D+SFN الذاتية_D+SFAJ للشاب_P+D+SMN أنه_ACC+PRO متفوق_SMAJ
أكاديميا_AV بشكل_P+SMN كامل_SMAJ منذ_P أن_SUB كان_PSTV في_P
الثانوية_D+SFN العامة_D+SFAJ وحتى_C+P التخرج_D+SMN من_P
الجامعة_D+SFN ،_PX لم_NEG يخفق_PRSV أبدا_AV !_PX
```

Fig. 3 An example of annotated sentence

4.1 Annotation of POS Tags

The POS tag set consists of 58 tags classified into 5 main categories:

```
<POSTAG> ::= <NOUN> | <ADJECTIVE> | <VERB> | <ADVERB>
             | <PREPOSITION> | <PARTICLE>
```

Nouns The noun class has 13 tags as shown in Table 4. The first nine tags are fine-grained annotations of common nouns that we classify according to their number (singular, dual, or plural) and gender (masculine, feminine, or irregular). We use the feature *irregular* to annotate the irregular plural nouns. As it is the case in MADA, we consider quantifiers, numbers, and foreign words as special noun classes. Following the Qur'an corpus, we consider pronouns, demonstrative pronouns, and relative pronouns as special noun classes (Table 3).

Adjectives The adjective class has nine tags as described in Table 4. Similar to nouns, the first seven tags are fine-grained annotations of adjectives that we classify according to their number and gender. As it is the case in MADA, we consider comparative adjectives and numerical adjectives as special adjective classes.

Verbs The verb class contains five tags as described in Table 4. The first four tags are fine-grained annotations of verbs that we classify according to their passive marking (active or passive) and tense (past or present). Annotating future tense in Arabic is explained in the particle class. For imperative verbs, we use the tag *IMPV*.

Adverbs It is not clear in the modern Arabic linguistics community whether *adverb* belongs to the Arabic part of speech system or not. In this study, we follow FARASA and MADA in considering adverbs as a category of the Arabic part of speech system, where we consider adverbs as *predicate* modifiers that we classify in three classes as shown in Table 4.

Prepositions and Particles This class contains 28 preposition and particle tags from the Qur'an corpus tag set that we list in Table 5.

Table 3 Clitic tags

No. of clitics	No. of tags	Examples
0	78	*SMN* كتاب
1	163	*C+PRSV* ويفعع
2	105	*C+PIN+PRO* وأصدقاؤه
3	12	*C+FUT+PRSV+PRO* وسيكتبه

Table 4 Noun, adjective, verb, and adverb tags

Tag	Explanation	Example
SMN	Singular masculine noun	رجل , جبل ,كتاب
SFN	Singular feminine noun	بنت , قرية , جريدة
DMN	Dual masculine noun	كتابي , كتابا , كتابين , كتابان
DFN	Dual feminine noun	قريتي , قريتا , قريتين , قريتان
PMN	Plural masculine noun	موظفي ,موظفو , موظفين , موظفون
PFN	Plural feminine noun	كتابات ,حالات , إسهامات , موظفات
PIN	Plural irregular noun	رجال ,بنات , قرى , كتب
FWN	Foreign noun	سيناتور , لابتوب , موبايل , بنسلين
NQ	Quantifiers	أي , بعض , كل , جميع
NM	Numbers	اثنين , اثنان , ١ ,واحد
PRO	Pronouns	ـها , هي , ـه ,هو
DM	Demonstrative pronouns	تلك , هؤلاء , هذان , هذا
REL	Relative pronouns	اللواتي , اللذان , التي ,الذي
SMAJ	Singular masculine adjective	سليم , قوي , جميل
SFAJ	Singular feminine adjective	سليمة , قوية , جميلة
DMAJ	Dual masculine adjective	جميلي , جميلا , جميلين , جميلان
DFAJ	Dual feminine adjective	سليمتان , قويتان , جميلتين
PMAJ	Plural masculine adjective	جميلي ,جميلو , جميليون , جميلين
PFAJ	Plural feminine adjective	سليمات , قويات , جميلات
PIAJ	Plural irregular adjective	حمر , أقوياء , أصحاء
AJCMP	Comparative adjectives	أسلم , أقوى , أجمل
AJNM	Ordinal adjectives	ثالث , ثاني ,أول
PRSV	Present verb (active)	يحمل , يسأل , يقول
PSTV	Past verb (active)	حمل , سأل ,قال
PPRSV	Present verb (passive)	يحمل , يسأل , يقال
PPSTV	Past verb (passive)	حمل , سئل , أقيل
IMPV	Imperative verb	احمل , اسأل , قل
T	Temporal adverb	بعد , أحيانا , صباحا
LC	Location adverb	بعد , تحت , فوق
AV	Adverb	تماما , خاصة , يوميا

Table 5 Preposition and particle tags

Tag	Explanation	Example
D	Definite article	ال
C	Conjunctions	ف , أو ,و
P	Prepositions	ل , الى ,من
Q	Interrogative particles	كيف ,هل ,ماذا
COND	Conditional particles	إنْ ,إذا ,لو
NEG	Negation particles	لن ,لا ,لم
ACC	Accusative particles	لعل , لكن ,إن
SUB	Subordinate particles	كي ,أنْ
FUT	Future particles	سوف ,سـ
VOC	Vocative particles	ـم ,يا
ANS	Answer particles	كلالا ,نعم
EXL	Explanation particles	أما ,أي
EXP	Exceptive particles	عدا ,سوى
EXC	Exclamation particles	يا ,ما
RES	Restriction particle	إلا
CERT	Certainty particle	قد
SUR	Surprise particle	إذن ,إذا
EMPH	Emphatic particle	لـ
PRP	Purpose particle	لـ
RET	Retraction particle	بل
REM	Resumption particles	و ,ف
INTG	Interrogative particle	أ
PREV	Preventive particle	ما
INC	Inceptive particle	ألا
IMPP	Imperative particle	ل
PR	Prohibition particle	لا
ABB	Abbreviation	د (دكتور)
PX	Punctuation	., :

4.2 Annotation of Word Segments

We represent the morphology of words through complex tags that correspond to their internal structure. As shown above, the structure of a complex tag is

$$[\text{PROCLITIC}+]^{*}\ \text{BASETAG}\ [+\text{ENCLITIC}]^{*}$$

where BASETAG is one of the base POS tags; ENCLITIC, when present, stands for one or two clitic tags at the end of the word; and PROCLITIC, when present, is the combination of one to three tags out of a set of the proclitic tags at the beginning of the word.

In our corpus, the number of distinct individual tags (including both simple and complex tags) is 358, as shown in Table 3.

4.3 Annotation of Named Entities

The named entity tags output by the ALP preprocessor are shown below:

```
<NERTAG>   ::= <POSITION> "-" <CLASS>
<POSITION> ::= "B" | "I"
<CLASS>    ::= "PER" | "LOC" | "ORG" | "EVENT"| "MISC"
```

Following the conventions of CONLL-2003,[5] the NER tags provide both the class of the entity and its boundaries through indicating the positions of the tokens composing it. B- stands for *beginning*, i.e., the first token of the entity, while I- stands for *internal*, marking subsequent tokens of the same entity.

Our corpus currently distinguishes the most common types of named entities: persons, locations, organizations, events, and others. We did not yet classify entity classes such as date, time, currency, or measurement nor subclasses of organizations (e.g., we do not differentiate between a football team and a university).

Thus, the total number of NER tags is 10 as shown in Table 6; however, as shown earlier, NER tags can be further combined with clitic tags.

Table 6 Named entity tags

Tag	Explanation	Example
B-PER, *I-PER*	Person	نجيب_B-PER محفوظ_I-PER
B-LOC, *I-LOC*	Location	البحر_B-LOC الأبيض_I-LOC المتوسط_I-LOC
B-ORG, *I-ORG*	Organization	حزب_B-ORG الحرية_I-ORG والعدالة_I-ORG
B-EVENT, *I-EVENT*	Event	الحرب_B-EVENTالعالمية_I-EVENT الثانية_I-EVENT
B-MISC, *I-MISC*	Misc	درب_B-MISC التبانة_I-MISC

[5] http://www.cnts.ua.ac.be/conll2003/ner/annotation.txt.

4.4 Annotation of Lemmas

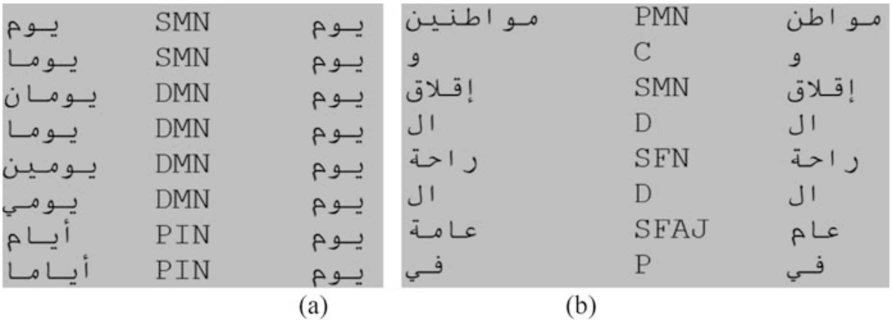
The lemmatization dictionary is a text file with each tab-separated row containing a word form, its POS tag, and the corresponding lemma and each column separated by a tab character. In case of ambiguous word forms (i.e., a word form-POS tag pair that has several lemmas), the corresponding lemmas are separated by "#" character: for example, the lemmas of the word form تزور are زور#زار. An example of the dictionary is shown in Fig. 4a.

The format of the annotated corpus for the learning-based lemmatizer is similar to the dictionary. The only difference is that the entries are ordered according to their original position in the sentence in the segmented corpus, in order to retain context. An empty line indicates the end of a sentence. An example of the training corpus is shown in Fig. 4b.

4.5 Annotation of Base Chunks

For example, the phrase الرجل الطويل/*the tall man* in Fig. 5 is a base chunk, while the phrase الرجل الطويل فوق المبنى/the tall man on the building is not.

We use the following BIO annotation schema to annotate the base chunks:



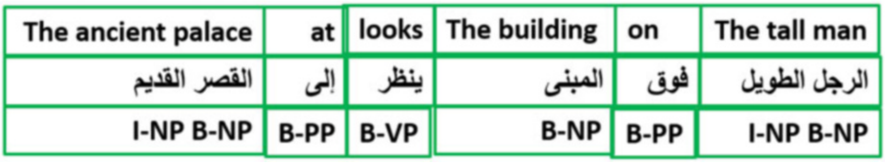
يوم	SMN	يوم
يوما	SMN	يوم
يومان	DMN	يوم
يوما	DMN	يوم
يومين	DMN	يوم
يومي	DMN	يوم
أيام	PIN	يوم
أياما	PIN	يوم

(a)

مواطنين	PMN	مواطن
و	C	و
إقلاق	SMN	إقلاق
ال	D	ال
راحة	SFN	راحة
ال	D	ال
عامة	SFAJ	عام
في	P	في

(b)

Fig. 4 Examples of the contents of (**a**) the lemmatization dictionary and (**b**) the training corpus of the classifier-based lemmatizer



The ancient palace	at	looks	The building	on	The tall man
القصر القديم	إلى	ينظر	المبنى	فوق	الرجل الطويل
I-NP B-NP	B-PP	B-VP	B-NP	B-PP	I-NP B-NP

Fig. 5 An example output of the base chunker

```
<PHRASE>    ::= <B> "-" <CLASS> (<I> "-" <CLASS)*
<B>         ::= "B"
<I>         ::= "I"
<CLASS>     ::= "NP" | "VP" | "PP" | "ADJP"| "ADVP"
```

In the following, we describe the base chunks and their annotations:

- Nominal phrases: The annotation schema for basic nominal phrases is B-NP (I-NP)*. This can be one of the following basic noun phrase categories:
  1. Pronouns: This category refers to separate pronouns such as هو/*he* or attached object pronouns like ه/*him*. Attached possessive pronouns like ه/*his* are genitive constructions as decribed below. Pronouns are annotated using the tag B-NP.
  2. Nouns: These refer to single-word nouns. They can be definite such كتاب/*book* and indefinite like الكتاب/*the book*. We use the tag B-NP if the noun is indefinite and the tag B-NP I-NP otherwise.
  3. Nouns+possessive pronouns: This category refers to nouns followed by attached possessive pronouns such as كتابي/*my book*. Such phrase is annotated as B-NP I-NP.
  4. Nouns+adjective modifiers: This category refers to nouns modified by one or more adjectives. They can be definite nouns such as الكتاب الجديد/*the new book* and كتابي الجديد/*my new book* or indefinite nouns like كتاب جديد/*a new book*. The possible tag sequence here is (B-NP|(B-NP I-NP)) (I-NP)$^+$.
  5. Genitive nouns: Noun+possessive pronouns are special case of this category. It includes also the other forms of genitive constructions. They can be definite nouns such كتاب المدرسة/*The school book* or indefinite nouns like كتاب مدرسة/*school book*. The length of the noun sequence can be of course more than two tokens such as كتاب معلم المدرسة/*the school teacher book*. We use the tag sequence B-NP (I-NP)$^+$ here.
  6. Genitive nouns+adjective modifiers: This category contains possessive constructions modified by one or more adjectives. The phrases may be definite such as كتاب المدرسة الجديد/*the new school book* or indefinite like كتاب مدرسة جديد/*a new school book*. The used tag sequence here is B-NP (I-NP) (I-NP)$^+$.
  7. Proper nouns: This category contains proper nouns such as محمد/*Mohammed* or محمد بن عبد الله/*Mohammed bin Abdullah*. The length of the sequence can be in some cases very long.
  8. Nouns+proper nouns+adjective modifiers B-NP (I-NP) (I-NP)$^+$: This category contains phrases such as خطاب ترامب الهجومي/*Trump's offensive speech*.

- Prepositional phrases: Prepositional phrases begin with a preposition followed by one of the basic nominal phrases. The only used tag here is B-PP.
- Verbal phrases: Verbal phrases consist of two parts—a verbal part that may be followed by one of the basic nominal phrases. The sequence in the verbal part may contain one token which the main verb such as تعلم/*learned*, an auxiliary verb followed by the phrase main verb such as كان يتعلم/*was learning*, negation particle followed by the main verb such as لم يتعلم/*didn't learn*, or a negation particle followed by an auxiliary verb and the main verb such as لم يكن يتعلم/*was not learning*. The possible tag sequences are B-VP (NULL—I-VP—(I-VP I-VP)).
- Predicative adjective phrases: Predicative constructions may be a predicative adjective such as جميل/*beautiful* or a negation particle followed by a predicative adjective like غير جميل/*not beautiful*. The possible tag sequences are B-ADJ (NULL|I-ADJ | (I-ADJ I-ADJ)).
- Adverbial phrases: Adverbial constructions contain an adverb that modifies a verbal or adjectival phrase. Examples for this category are جدا/*very*, أسبوعيا/*weekly*, or شخصيا/*personally*. The used tag for adverbial phrase is B-ADV.

5 Corpus Annotation

This section presents the methods we used to annotate the more than two-million-token corpus that is the fundament of ALP. We have chosen to develop an entirely new corpus instead of relying on existing resources such as the Penn Arabic Treebank [26] in order to be free both from licensing restrictions and from past modeling choices. The main challenge of the annotation process, besides the corpus size, was to cover and address as many cases of ambiguity (discussed in Sect. 2) as possible.

The corpus was assembled from documents and text segments from a varied set of online resources in Modern Standard Arabic, such as medical consultancy web pages, Wikipedia, news portals, online novels, and social media, as shown in Table 7. The diversity of sources serves the purpose of increasing the robustness of the model with respect to changes in domain, genre, style, and orthography. For consistency within the corpus and with the type of texts targeted by our annotator, we removed all short vowels from the input corpora.

The current corpus consists of more than 130k annotated sentences with more than 2 millions Arabic words and 200k punctuation marks (Table 8).

Table 7 Resources used in the training corpus

Resource	Proportion
Aljazeera online	30%
Arabic Wikipedia	20%
Novels	15%
Al-quds Al-Arabi newspaper	10%
Altibbi medical corpus	10%
IslamWeb	5%
Social networks (Facebook, Twitter)	5%
Other resources	5%

Table 8 Domains covered in the training corpus

No. of documents	No. of tokens	Domain	Description
Archaeology	61	19745	Archaeological and fossil science
Articles	285	261264	Articles from news portals
Business	73	49937	(Online) Business and trading
Economy	159	61,954	Regional and international economy
Encyclopedia locations	140	129553	Cities, villages, and countries
Encyclopedia persons	131	90601	Famous persons
Encyclopedia politics	53	52080	Political organizations and events
Environment	110	41506	Environment-related documents
Food recipes	11	4511	Food and cooking recipes
Literature	130	264354	Novels, short stories, proverbs
Medical	339	159,452	Human health-related documents
Medical answers	1	297608	Medical question answers from medical portals
Miscellaneous	24	2190	Uncategorized documents
Miscellaneous news	985	276323	None political or military nature news
Nature	62	25044	Nature-related documents
News	1013	373884	Political and war news
Quran	1	2915	Verses from Quran
Science	196	73650	Science-related documents
Space	84	41629	Space-related documents
Sport	30	9398	Sport-related documents
Technology	77	24588	Technology-related documents
Theology	10	19576	Islamic theology articles

5.1 POS and Name Annotation Method

The annotation was performed semi-automatically in two major steps:

1. Annotation of a corpus large enough to train an initial machine learning model.

2. Iterative extension of the corpus. We add new sets of sentences tagged by the current model after manual correction. Then, we retrain the model after each iteration.

Step 1 was an iterative process. It was bootstrapped using a 200-sentence gold standard *seed corpus* that was fully hand-annotated. The goal of each iteration was to add a new set of 100 new sentences to the seed corpus, until about 15k sentences were tagged. Each iteration consisted of the following steps:

1.a For each word in the untagged corpus that was already tagged in the seed corpus, simply copy the tag from the tagged word (this of course can lead to wrong tagging as the process does not take the context into account; we fix such mistakes later).
1.b Find the 100 sentences with the highest number of tags obtained through replacement in the previous step.
1.c Manually verify, correct, and complete the annotation of the sentences extracted in step 1.b.
1.d Add the annotated and verified sentences to the seed corpus and repeat.

At the end of step 1, many rounds of manual verification were performed on the annotated corpus.

In step 2, we extended the corpus in an iterative manner:

2.a Train an initial machine learning model from the annotated corpus.
2.b Use this model to label a new set of 100 sentences.
2.c Verify and correct the new annotations obtained.
2.d Add the newly annotated sentences to the annotated corpus and repeat.

For training the machine learning model, we used the POS tagger component of the widely known OpenNLP tool with default features and parameters. The annotation work was accomplished by two linguists, the annotator and a consultant who was beside the design of the tag set active in corrections and consultations especially in the first phase.

5.2 Lemma Annotation Method

In the following, we describe the method we used to build the lemmatization dictionary and the annotations for the learning-based lemmatizer. Our starting point was the POS corpus that we extended with lemma annotations.

5.2.1 Dictionary Lemmatizer

The dictionary was generated through the following steps:

Table 9 Plural classes

Class	Possible word forms	Example
SMN_PMN	SMN,SFN,DMN, DFN, PFN, PMN	مؤمنون,مؤمنات,مؤمنتان,مؤمنان,مؤمنة ,مؤمن:مؤمن
SMN_PFN	SMN,DMN,PFN	إجراءات,إجراءان,إجراء.,:إجراء
SFN_PFN	SFN,DFN,PFN	كتابات,كتابتين,كتابة.,:كتابة
SMN_PIN	SMN,DMN,PIN	أعلام,علمان,علم .,:علم
SFN_PIN	SFN,DFN,PIN	أسر,أسرتان,أسرة.,:أسرة
FWN_PFN	FWN,DMN,PFN	تلفزيونات,تلفزونان,تلفزيون.,:تلفزيون

1. **Segmentation.** The corpus was segmented as explained in the previous section. The result of this step was generating a segmented corpus that contains more than 3.1 million segmented tokens.
2. **POS tag-based classification.** In this step, we classified the word forms according to their POS tag.
3. **Inherent feminine and adjectival feminine classification.** In this step, we classified the feminine nouns into inherent feminine and adjectival feminine nouns. For example, the noun أسرة_SFN/*family* is inherent feminine, while the noun أسيرة_SFN/*prisoner* is adjectival. This differentiation is important because the lemma of adjectival nouns is the masculine singular form of the noun أسير, while there is no masculine singular lemma for أسيرة.
4. **Plural type classification.** In this step, we classified the singular and dual nouns (after extracting their singular forms) according to their plural type into six classes as shown in Table 9. This classification enables us to build the possible *word number-gender* forms of a given lemma automatically. For example, the class SMN_PMN has six different possible *number-gender* forms. On the other hand, using the feminine classification lists in the previous step enabled us to differentiate between the SMN_PFN and SFN_PFN. In the class SMN_PFN, the lemma of a singular feminine noun (SFN) is the singular masculine noun (SMN). In the class SFN_PFN, on the other hand, the lemma is the singular feminine noun itself. The adjectives were classified into three classes. The first class is similar to the class SMN_PMN which allows six different word forms. The second class contains a seventh possible form which is the broken plural adjective form. The third class contains PIAJ as a single possible plural form. For example, ماهر belongs to this class since it has two possible plural forms ماهرون and مهرة.
5. **Lemma extraction.** This step is semi-automatic as follows:
    (a) **Manual.** Assigning the lemmas to broken plural nouns and adjectives was performed manually.

   (b) **Automatic.** Based on the morphological features in the tags, it was possible to extract lemmas for singular, dual, masculine plurals, feminine plurals adjectives, and nouns. We also used rules to extract the verb lemmas such as removing the affixes وا.

6. **Lemma enrichment.** Using the lemmas from the previous step, we have enriched the corpus with new verbs, adjectives, and nouns. For example, if the lemma of a plural noun or adjective was missing, we added it to the noun and adjective lemma lists.
7. **Dictionary generation.** The files produced so far are as follows:

   - **Noun files.** Three files for masculine, feminine, and foreign nouns. The lemmas in these files were classified according to Table 9. There is a fourth file that contains quantifiers, pronouns, adverbs, etc.
   - **Adjectives.** Three files for adjectives, comparatives, and ordinal adjectives. The lemmas in the adjective file are classified according to Table 9.
   - **Verbs.** One file that contains all extracted verb lemmas.

   Using these files, the dictionary was generated as follows:

   (a) **Noun and adjective generation.** According to the plural class, the noun and adjective forms were generated. The ا case ending and changing ة to ت were also considered in this step.
   (b) **Verb generation.** For each verb in the verb lemma list, we automatically generated the verb conjugations in present, past, and imperative cases. We considered also accusative (فعل منصوب) and asserted verbs (فعل مجزوم).
   (c) **Dictionary building.** Using the results from the previous step, we built the dictionary as shown in Fig. 4b, where the lemmas of ambiguous surface forms were concatenated into a single string using the # separator.

5.2.2 Machine Learning Lemmatizer

We used the segmented corpus from the previous section to build the lemmatization corpus in the two steps below:

1. **Lemma assignation.** We used a dictionary lemmatizer to assign the word forms to their corresponding lemmas. In case of prepositions, particles, and numbers, the lemma of the word form was obtained through simple normalization. The lemmas of named entities were the named entities themselves. If a word form was ambiguous, all its possible lemmas were assigned.
2. **Validation.** We disambiguated the lemmas of the ambiguous word forms manually.

The size of the generated corpus is 3,229,403 lines. The unique word forms after discarding the digits are 59,049 as shown in Table 10.

Table 10 Distribution of lemmas and unique word forms by part of speech in the corpus of the learning-based lemmatizer

POS	No. of lemmas	No. of word forms
Noun	18,165	26,337
Adjective	6369	13,703
Verb	4258	19,009
Named entity	20,407	20,407
Particle	605	649

In a final step, we added all generated word forms and their corresponding lemmas from the dictionary described in the previous section to the corpus. This increased the size of the corpus to 3,890,737 lines.

5.3 Base Chunking Annotation Method

We used the segmented corpus from the previous section to build the chunking corpus by using the BIO tags semi-automatically. The manual part of this process was identifying the POS tag sequences that constitute a phrase chunk. The total number of the identified sequences was 4298. Most of these sequences were nominal phrase sequences, where the number of this group was 4250 sequence. In Table 11, we give examples for the identified noun sequences. The complete identified POS tag sequences can be found on the project page on ResearchGate.[6] In Table 12, we show the identified verb sequences. Table 13 contains predicative adjective sequences. The prepositional and the adverbial groups contained one sequence only which is the preposition or the adverb.

Using the identified sequences, we have built the corpus automatically. The size of the current chunking corpus is more than three million tokens. Of course, we do not claim that the identified sequences are exhaustive. There may be other non-detected sequences in our corpus or other sequences that our corpus does not cover.

6 Evaluation

In this section, we present evaluation results, both on individual NLP tasks and overall results pertaining to the ALP pipeline as a whole (the latter included within the lemmatizer results).

[6] https://www.researchgate.net/project/ALP-Arabic-Linguistic-Tool.

Table 11 Noun phrases Sequences

Length	No. of sequences	Sequences sample	Sample phrase
1	21	D+SMN	البيت
2	351	D+SFN D+SFAJ	الراحة الجسدية
3	1119	SMN PFN D+DMN	تعزيز علاقات البلدين
4	1534	AJCMP PFN SMN D+PFN	أهم مقومات نجاح الشركات
5	861	PFN SMN PFN D+PIN D+PIAJ	احتمالات ارتفاع مستويات الديون المتعثرة
6	263	SMN PIN SFN D+SMN D+SFAJ D+SFAJ	تنفيذ بنود خطة العمل الشاملة المشتركة
7	68	SFN SMN PFN SMN D+PFN D+PFAJ D+PFAJ	ورشة عمل مهارات إعداد الموازنات التشغيلية المالية
8	14	SMN NQ PIN SMN SFN SMN D+PIN D+PIAJ	اتباع بعض طرق تسهيل عملية بلع الأقراص الدوائية

Table 12 Verbal phrase sequences

Length	No. of sequences	Sequence sample	Sample phrase
1	5	PSTV	أرجع
2	14	PRSV PSTV	تكون أأارت
3	7	NEG PSTV PPRSV	ما زالت ترتكب

Table 13 Predicative adjective phrases Sequences

Length	No. of sequences	Sequence sample	Sample phrase
1	9	DFAJ	صامتين
2	9	NEG SFAJ	غير ضارة
3	7	SMAJ SMAJ SMAJ	بني رمادي مشقق

6.1 Evaluation of POS Tagging

To evaluate the performance of the proposed solution, we trained a machine learning model on the annotated corpus using the *OpenNLP maximum entropy POS tagger* with default features and *cutoff* = 3. We did not apply any preliminary normalization to the evaluation corpus. The evaluation corpus was taken from two sources: the *Aljazeera* news portal and the *Altibbi* medical consultancy web portal. The text contained 9990 tokens (9075 words and 915 punctuations). Manual validation of the evaluation results was performed. The per-task accuracy figures are shown in Table 14.

Table 14 Evaluation results on the POS tagging and word segmentation tasks

Error type	Number of errors	Accuracy
Segmentation	25	99.7%
Coarse-grained POS	131	98.7%
Fine-grained POS	206	97.9%

The *segmentation* error type refers to words that were not segmented correctly. The *coarse-grained POS* error type refers to words that were correctly segmented but the base POS tag was wrong. This also includes incorrect named entity POS tags. Finally, the *fine-grained POS* error type means that the word segmentation and the coarse-grained POS tag were correct but the fine-grained information within the tag was incorrect in one of the following ways:

- For nouns and adjectives: number/gender error
- For verbs: tense error or passive/active voice error

In some cases, the tag included more than one type of error. For example, the ومضرة_SFN tag (instead of C+SFAJ) includes both segmentation and POS tagging errors and therefore was counted twice.

The evaluation data and the process to replicate the evaluation tests are available online.[7]

6.2 Evaluation of NER

We evaluated the named entity recognition component separately. Our evaluation corpus contained 674 named entity tags that denote 297 named entities (e.g., روبرت_B-PER واتسون_I-PER is one named entity that contains two named entity tags). The total number of true positives (correctly detected and classified named entities) was 265 (89.2% precision). The number of false negatives (assigning a non-named entity tag, partial tagging, named entity boundary error, or a wrong named entity class applied) was 32 and the number of the false positives 15 (94.6% recall). F1-Measure = 91.8%. In Table 15, we provide some examples of these errors.

6.3 Evaluation of Lemmatization and Base Chunking

For evaluating the lemmatizers, we used a corpus of a 46,018-token text, retrieved and assembled from several news portals (such as Aljazeera news portal[8] and *Al*-

[7] http://www.arabicnlp.pro/alp/eval.zip.

[8] http://www.aljazeera.net/.

Table 15 Examples of NER mistakes

Error type	Example
Non-NER tag	وففوريدا_C+SMN instead of C+B-LOC
Partially tagged	وفيك_SMN سيجبيسما_I-PER instead of وفيك_C+B-PER سيجبيسما_I-PER
Boundary error	شركة_SFN سيبريان_B-ORG االروسية_I-ORG instead of شركة_SFN سيبريان_B-ORG االروسية_D+SFAJ
Wrong classification	غرين_B-LOC (in ودعا غرين) instead of غرين_B-PER
False positive	نظم_P+B-ORG ااآيكوووجية_I-ORG instead of نظم_P+D+PIN ااآيكوووجية_D+PIAJ

quds Al-Arabi newspaper[9]). We excluded from the evaluation the categories of tokens that cannot be lemmatized: 5853 punctuation tokens, 3829 tokens tagged as named entities, 482 digit tokens, and 10 malformed tokens (i.e., containing typos, such as أوروبيةبالموافق instead of أوروبية بالموافق). Thus, the number of tokens considered was 35,844. In order to have a clear idea of the efficiency of the lemmatization pipeline, we evaluated it in a fine-grained manner, manually classifying the mistakes according to the component involved. This allowed us to compute a comprehensive accuracy for the entire pipeline as well as evaluate individual components. The evaluation data files are available online.[10]

The fine-grained evaluation is summed up in Table 16.[11] *Nonexistent lemma* stands for cases where the POS tag and the segmentation were correct, yet the classifier gave a wrong, non-linguistic result. *Wrong disambiguation* means that the lemmatizer chose an existing but incorrect lemma for an ambiguous word form.

The accuracy measures reported in Table 17 were computed based on the results in Table 16. On these, we make the following observations. The performance of preprocessing (98.6%) represents an upper bound for the entire lemmatization pipeline. In this perspective, the near-perfect results of the classifier (99.5% when evaluated in isolation, 98.1% on the entire pipeline) are remarkable. We cross-checked these results using the built-in cross-validation feature of OpenNLP and obtained similar results (99.7%). The dictionary-based lemmatizer reached a somewhat lower yet still very decent result (96.6% in isolation, 95.2% on the entire pipeline), due to the 1207 OOV word forms. The fusion of the two lemmatizers, finally, improved slightly on the classifier: of the 170 mistakes made by the classifier, 120 could be correctly lemmatized using the dictionary. Thus, the fusion method

[9] http://www.alquds.co.uk/.

[10] http://www.arabicnlp.pro/alp/lemmatizationEval.zip.

[11] While, after tagging and segmentation, the number of (segmented) tokens rose to 62,694, we computed our evaluation results based on the number of unsegmented tokens.

Table 16 Types of mistakes committed by the learning-based lemmatizer and their proportions

Type of mistake	Occurrences	Example
POS tag (coarse-grained) mistakes	199	تبرج_SMN instead of تبرج_PRSV
Morphological tag (fine-grained) mistakes	201	كروم_SMN instead of كروم_PIN
Segmentation tag mistakes	103	أخذتنا_PSTV instead of أخذت_PSTV and نا_PRO
Classifier mistakes: nonexistent lemma	158	جدا instead of وجد for يجدوا_PRSV
Classifier mistakes: wrong disambiguation	12	كان instead of كون for يكونوا_PRSV
Dictionary mistakes: missing word form	1207	كاشطة,دواعم,قرفصاء
Fusion mistakes	50	نتن, عواام, دواعم

Table 17 Accuracy values computed for various components of the lemmatization pipeline

Component	Evaluation method	Accuracy
Preprocessing	All mistakes (POS, morphological, segmentation)	98.6%
Classifier-based lemmatizer	In isolation	99.5%
Classifier-based lemmatizer	In isolation, built-in OpenNLP cross-validation	99.7%
Classifier-based lemmatizer	Entire pipeline	98.1%
Dictionary-based lemmatizer	In isolation	96.6%
Dictionary-based lemmatizer	Entire pipeline	95.2%
Fusion lemmatizer	Entire pipeline	98.4%

reached a full-pipeline result of 98.4%, only a tiny bit worse than the performance of preprocessing itself. We have used cross-validation method to evaluate the chunking corpus. The evaluation result was 98.6%.

7 Conclusion and Future Work

This paper presented ALP, an Arabic linguistic pipeline that solves low-level Arabic NLP tasks: POS tagging, word segmentation, named entity recognition, and lemmatization. All of these tasks were performed using tools and resources derived from a new 2.2-million-word corpus hand-annotated by the authors. Due to the size of the corpus but also the annotation schemas and the overall pipeline design, ALP manages to disambiguate a large proportion of cases of lexical ambiguity and perform the tasks above with high accuracy. This increases the potential of downstream language understanding tasks, some of which we are planning to include in ALP in the future.

In particular, we are working on new Arabic components such as a vocalizer, a phrase chunker, a dependency parser, or a multiword expression detector.

The trained models and corresponding tools are free for research purposes upon request. We are also planning to release the annotated corpus itself in the near future.

We are also planning further improvements on the existing components, as detailed below:

Fine Tuning While the tool reached very good results with default OpenNLP features, we believe that they can still be improved by customizing the classifier and the features or using another machine or deep learning algorithm such as CRF and BiLSTM.

Noun Classification In the current tag set, we do not differentiate between gerunds (المصدر) and other noun classes. For example, the noun قلب/*heart* is tagged the same as the gerund قلب/*overthrow*.

Named Entity Classification The classification of named entities in our corpus is still incomplete and coarse-grained. For example, العباسيين, الشعب الألماني, and اللغة العربية are not classified as named entities. We plan to introduce new classes such as dates and currencies, as well as a finer-grained classification of organizations.

Chunker Coverage Extension We are planning to extend the chunker to detect sentence phrases without restriction. This will include detecting coordinated nominal phrases, verbal phrases, and relative clauses. We plan also to do more research on the relation between adverbs and other phrases and find a way to connect the modifying adverb to its modified phrase. On the other hand, we will also work on extending the verb POS tags to differentiate between transitive and intransitive verbs and study the effect of this new verb classification on proper name boundary disambiguation.

Other Tools and Corpora We plan to use the same corpus and tag set to produce annotations for other NLP tasks such as co-reference resolution and parsing.

References

1. Balakrishnan, V., Ethel, L.: Stemming and lemmatization: a comparison of retrieval performances. Lect. Notes Soft. Eng. **2**, 262–267 (2014)
2. Navigli, R.: Word sense disambiguation: a survey. ACM Comput. Surv. **41**, 10:1–10:69 (2009). http://doi.acm.org/10.1145/1459352.1459355
3. Bella, G., Zamboni, A., Giunchiglia, F.: Domain-based sense disambiguation in multilingual structured data. In: The Diversity Workshop at the European Conference on Artificial Intelligence (ECAI) (2016)
4. Freihat, A., Qwaider, M., Giunchiglia, F.: Using grice maxims in ranking community question answers. In: Proceedings of the Tenth International Conference on Information, Process, and Knowledge Management, EKNOW 2018, Rome, March 25–29, pp. 38–43 (2018)

5. Giunchiglia, F., Kharkevich, U., Zaihrayeu, I.: Concept search. In: The Semantic Web: Research and Applications, pp. 429–444 (2009)
6. Darwish, K., Mubarak, H., Abdelali, A., Eldesouki, M.: Arabic POS tagging: Don't abandon feature engineering just yet. In: Proceedings of the Third Arabic Natural Language Processing Workshop, pp. 130–137 (2017)
7. Diab, M.: Second generation AMIRA tools for Arabic processing: Fast and robust tokenization, POS tagging, and base phrase chunking. In: 2nd International Conference on Arabic Language Resources and Tools, vol. 110 pp. 285–288 (2009)
8. Khoja, S.: APT: Arabic part-of-speech tagger. In: Proceedings of the Student Workshop at NAACL, pp. 20–25 (2001)
9. Aldarmaki, H., Diab, M.: Robust part-of-speech tagging of Arabic text. In: Proceedings of the Second Workshop on Arabic Natural Language Processing, pp. 173–182 (2015)
10. Habash, N., Rambow, O.: Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 573–580 (2005)
11. Sawalha, M., Atwell, E.: Fine-grain morphological analyzer and part-of-speech tagger for Arabic text. In: Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10), pp. 1258–1265 (2010)
12. Mohamed, E., Kübler, S.: Is Arabic part of speech tagging feasible without word segmentation? In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 705–708 (2010)
13. Shaalan, K., Raza, H.: Arabic named entity recognition from diverse text types. In: Proceedings of the 6th International Conference on Advances in Natural Language Processing, pp. 440–451 (2008)
14. Althobaiti, M., Kruschwitz, U., Poesio, M.: A semi-supervised learning approach to Arabic named entity recognition. In: Recent Advances in Natural Language Processing, RANLP 2013, 9–11 September, Hissar, Bulgaria, pp. 32–40 (2013). http://aclweb.org/anthology/R/R13/R13-1005.pdf
15. Darwish, K.: Named entity recognition using cross-lingual resources: Arabic as an example. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1 pp. 1558–1567 (2013)
16. Abdallah, S., Shaalan, K., Shoaib, M.: Integrating rule-based system with classification for Arabic named entity recognition. In: Computational Linguistics and Intelligent Text Processing - 13th International Conference, CICLing 2012, New Delhi, March 11–17, 2012, Proceedings, Part I, pp. 311–322 (2012)
17. AlGahtani, S.: Arabic Named Entity Recognition: A Corpus-Based Study, Ph.D. Thesis. University of Manchester (2011)
18. Boudchiche, M., Mazroui, A., Ould Abdallahi Ould Bebah, M., Lakhouaja, A., Boudlal, A.: AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer. J. King Saud Univ. Comput. Inf. Sci.. **29**, 141–146 (2017). https://doi.org/10.1016/j.jksuci.2016.05.002
19. Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R.: MADAMIRA: a fast, comprehensive tool for morphological analysis and disambiguation of Arabic. LREC. **14**, 1094–1101 (2014)
20. Abdelali, A., Darwish, K., Durrani, N., Mubarak, H.: Farasa: A fast and furious segmenter for arabic. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 11–16 (2016)
21. Attia, M., Zirikly, A., Diab, M.: The power of language music: Arabic lemmatization through patterns. In: Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon, CogALex@COLING 2016, Osaka, December 12, 2016, pp. 40–50 (2016). https://aclanthology.info/papers/W16-5306/w16-5306
22. Al-Shammari, E., Lin, J.: A novel Arabic lemmatization algorithm. In: Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data, pp. 113–118 (2008). http://doi.acm.org/10.1145/1390749.1390767

23. El-Shishtawy, T., El-Ghannam, F.: An accurate Arabic root-based lemmatizer for information retrieval purposes. CoRR **abs/1203.3584** (2012). http://arxiv.org/abs/1203.3584
24. Diab, M.: Improved Arabic base phrase chunking with a new enriched POS tag set. In: Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, pp. 89–96 (2007). https://www.aclweb.org/anthology/W07-0812
25. Darwish, K., Mubarak, H.: Farasa: A new fast and accurate Arabic word segmenter. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (2016)
26. Maamouri, M., Bies, A., Buckwalter, T., Mekki, W.: The penn arabic treebank: Building a large-scale annotated Arabic corpus. In: NEMLAR Conference on Arabic Language Resources and Tools, vol. 27, pp. 466–467 (2004)
27. El-Haj, M., Koulali, R.: KALIMAT a multipurpose Arabic corpus. In: Second Workshop on Arabic Corpus Linguistics (WACL-2), pp. 22–25 (2013)
28. Freihat, A., Bella, G., Mubarak, H., Giunchiglia, F.: A single-model approach for Arabic segmentation, POS tagging, and named entity recognition. In: 2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP), pp. 1–8 (2018)
29. Freihat, A., Abbas, M., Bella, G., Giunchiglia, F.: Towards an optimal solution to lemmatization in Arabic. In: Proceedings of the 4th International Conference on Arabic Computational Linguistics (ACLing 2018), pp. 1–9 (2018)
30. Shaalan, K.: A survey of Arabic named entity recognition and classification. Comput. Linguist. **40**, 469–510 (2014)
31. Dukes, K., Habash, N.: Morphological annotation of quranic Arabic. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10) (2010)
32. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, pp. 173–180 (2003). https://doi.org/10.3115/1073445.1073478

Arabic Anaphora Resolution System Using New Features: Pronominal and Verbal Cases

Abdelhalim Hafedh Dahou, Mohamed Abdelmoazz, and Mohamed Amine Cheragui

Abstract Anaphoric resolution is one of the major problems in natural language processing (NLP), since this linguistic phenomenon is very answered in natural languages, especially in Arabic; therefore, a resolution mechanism is needed in several applications of NLP. The resolution process consists in establishing the link between anaphoric entities and their referents in the text. This research has two main objectives—the first one is to implement a resolution system dedicated to pronominal and verbal anaphoras in the Arabic language (called A^3T) based on a rule-based approach combining rules and statistical features to identify the referent. The second objective is to create a specialized corpus for Arabic anaphora resolution which we've named A^3C, in order to fill the gap of lack of resources. Using the AnATAr as a test corpus, our A^3T system obtains an accuracy of 83.19% for pronominal anaphora and 57.23% for verbal anaphora.

1 Introduction

Most of the obstacles faced by tasks in natural language processing are related to the presence of linguistic phenomena specific to the language that make this process difficult. One of these phenomena is anaphora.

One of the standard definitions of anaphora is given by [1] based on the notion of cohesion: "anaphora is a cohesion (presupposition) which points back to some previous item." In a discourse, whether spoken or textual, we often refer to the same object, fact, action, or event in a repetitive way. But we do not always refer to it in the same way. This avoids unnecessary repetition of information and ensures coherence in our discourse.

If anaphora is considered by linguists as an elegant way to avoid repetition [2], it constitutes a real challenge issue in NLP due to the difficulty of identifying the

A. H. Dahou (✉) · M. Abdelmoazz · M. A. Cheragui
Department of Mathematics and Computer Science, Ahmed Draia University, Adrar, Algeria

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_5

referent. Hence, setting up an anaphoric resolution approach has become essential for several languages, in particular Arabic, which has morphological and syntactic characteristics that make identification and resolution a real challenge, but also for a multitude of applications, such as sentiment analysis [3], question-answer systems [4], machine translation [5], text summarization [6], information extraction [7], and language generation and dialog systems [8], to improve their performance.

In our work, we aim to achieve—first, the development of an anaphora resolution system (A^3T) specific to pronominal and verbal anaphoras in the Arabic language, where our contribution is to propose a rule-based approach, and second, to build an annotated corpus (A^3C) to overcome the scarcity of linguistic resources that Arabic faces.

The present paper is organized as follows: in Sect. 2, we present varieties of anaphora in Arabic text. In Sect. 3, we review related work about Arabic anaphora resolution systems and corpus. In Sect. 4, we outline some Arabic anaphora resolution challenge. We describe in Sect. 5 the full system of anaphora resolution and text annotation using a friendly interface. In Sect. 6, we evaluate the proposed approach. Section 7 presents a discussion, and finally, Sect. 8 concludes the paper and gives some future directions.

2 Varieties of Anaphora in Arabic Text

What makes the anaphora resolution mechanism complex in natural language processing in general and in Arabic, in particular, is the fact that it can manifest in different forms (linguistic categories: lexical and grammatical) but also requires knowledge at different levels, as well as an “understanding” of the context. There are many varieties of anaphora in the Arabic text, and we will only mention the most frequent ones.

2.1 Verb Anaphora

Verbal anaphora is used to describe or represent various movements or actions by using the verb (did—فعل—) and the different conjugation variants to minimize writing and avoid repetition (Fig. 1) [9, 10].

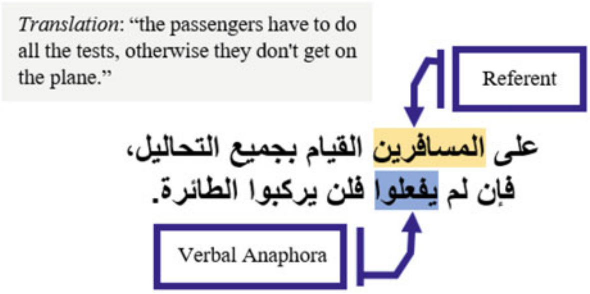

Translation: "the passengers have to do all the tests, otherwise they don't get on the plane."
Referent
على المسافرين القيام بجميع التحاليل،
فإن لم يفعلوا فلن يركبوا الطائرة.
Verbal Anaphora

Fig. 1 Example of verbal anaphora

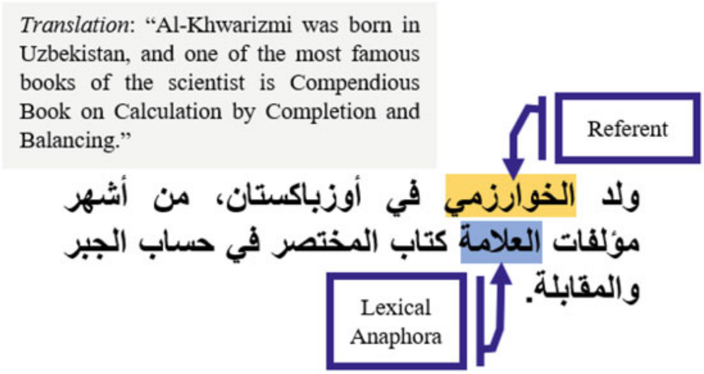

Translation: "Al-Khwarizmi was born in Uzbekistan, and one of the most famous books of the scientist is Compendious Book on Calculation by Completion and Balancing."
Referent
ولد الخوارزمي في أوزباكستان، من أشهر
مؤلفات العلامة كتاب المختصر في حساب الجبر
والمقابلة.
Lexical Anaphora

Fig. 2 Example of lexical anaphora

2.2 Lexical Anaphora

Lexical anaphora occurs when the referent is designated by definite descriptions or proper names representing the same concept (the anaphora) or concepts that are semantically close [11]. Usually, this form of anaphora adds more information to the sentence and increases cohesion and can take several forms (synonym, generalization/hypernymy, or specialization/hyponymy) (Fig. 2) [12].

2.3 Pronominal Anaphora

Based on statistical studies done by [11], it shows that the pronominal anaphora is the most frequent variant in Arabic texts. Pronouns form a special class of anaphora because of their empty semantic structures, and they have a meaning independent of its referents and usually refer to names or noun phrases [13]. However, not all pronouns are anaphoric.

Pronominal anaphoras can be divided into three categories, and each category can be subdivided into subcategories according to several parameters, such as gender, number, etc.

2.3.1 Third-Person Personal Pronouns

In the Arabic, not all personal pronouns are anaphoric, so the first-person (انا و نحن) and second-person (أنت ، انتما ....) pronouns are not (they specify the communication partners and their meaning goes back to their specific uses), except the third-person pronouns which have this characteristic (Figs. 3 and 4). These pronouns can be subdivided into two categories (Tables 1 and 2) [4]:

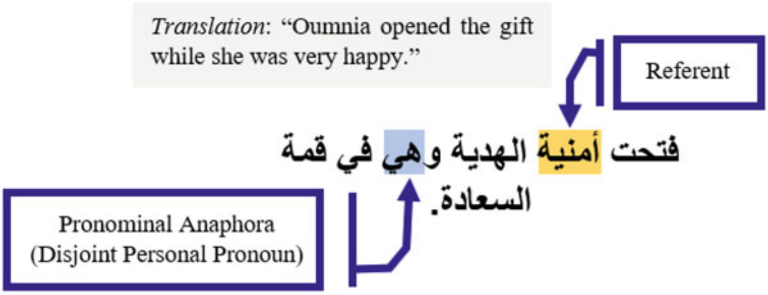
In some cases, the pronouns ـه and ـها are not anaphoric since they are not interpreted as related to an expression (referent). In this case, we will call them pleonastic pronouns (Fig. 5).

2.3.2 Relative Pronouns

Relative pronouns in Arabic have the characteristic of being always anaphoric; in addition, they have only one possible referent [14] and refer to the immediate nominal phrase mentioned before [15] which they agree in gender and number (Fig. 6; Table 3).

Table 1 Disjoint Personal Pronouns

		Nominative (محل رفع)			Accusative (محل نصب)		
3rd Person Pronoun		Singular	Dual	Plural	Singular	Dual	Plural
Disjoint Personal Pronouns (الضمائر المنفصلة)	Masculin	هو	هما	هم	إياه	إياهما	إياهم
	Feminine	هي		هن	إياها		إياهن



Translation: "Oumnia opened the gift while she was very happy."

فتحت أمنية الهدية وهي في قمة السعادة.

Fig. 3 Example of anaphora (case: third-person personal pronouns (disjoint))

Table 2 Joint Personal Pronouns

3rd Person Pronoun		Nominative (محل رفع)			Accusative (محل نصب)		
		Singular	Dual	Plural	Singular	Dual	Plural
Joint Personal Pronouns (الضمائر المتصلة)	Masculin	ا، و، ن			ـه	ـهما	ـهم
	Feminine				ـها		ـهن

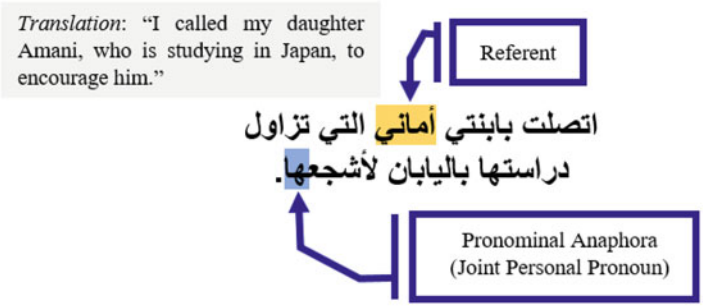

Translation: "I called my daughter Amani, who is studying in Japan, to encourage him."
Referent
اتصلت بابنتي أماني التي تزاول
دراستها باليابان لأشجعها.
Pronominal Anaphora
(Joint Personal Pronoun)

Fig. 4 Example of anaphora (case: third-person personal pronouns (joint))

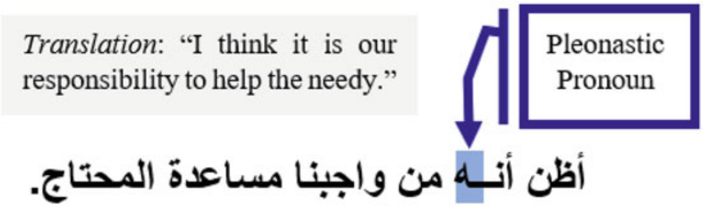

Translation: "I think it is our responsibility to help the needy."
Pleonastic
Pronoun
أظن أنــه من واجبنا مساعدة المحتاج.

Fig. 5 Example of pleonastic pronouns

Table 3 Relative pronoun

	Singular	Plural	Dual
Masculine	الذي	اـاذيــن، اااأى، اااَؤون، ااأأء	الــذَان، اااذين
Feminine	اـاــتي، اـاـت، اـاــتت	اـاــواتــي، ااااَتي، اـاــوَى	اـاـــتان، ااتين

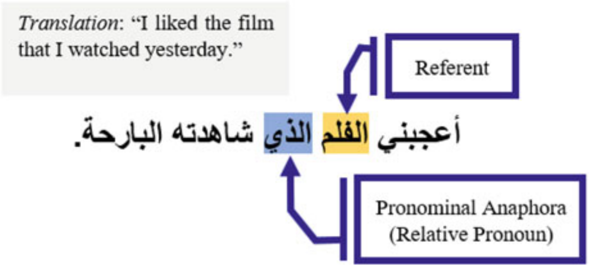

Translation: "I liked the film that I watched yesterday."
Referent
أعجبني الفلم الذي شاهدته البارحة.
Pronominal Anaphora (Relative Pronoun)

Fig. 6 Example of anaphora (case: relative pronouns)

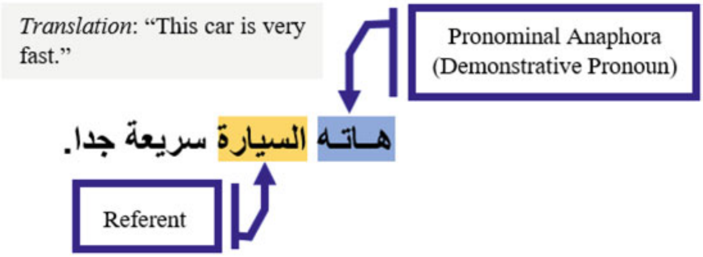

Translation: "This car is very fast."
Pronominal Anaphora (Demonstrative Pronoun)
هـاتـه السيارة سريعة جدا.
Referent

Fig. 7 Example of anaphora (case: demonstrative pronouns)

The use of relative pronouns is possible if the referent denotes a process or situation, and here the anaphora denotes some of these lexical meanings. They refer to persons, places, or things that are close or distant, and the table below illustrates this type of pronouns.

2.3.3 Demonstrative Pronouns

They are linguistic elements that accompany a designation gesture in order to coordinate the attention of the interlocutors when they are speaking [16]. Generally, demonstrative pronouns are cataphoric, and in some cases, they can be anaphoric and even deixis [15]. Demonstratives agree in person, gender (masculine/feminine), and number (singular/dual/plural) with their referent (Tables 4 and 5). In addition, there are pronouns which are considered demonstratives and which designate time and place (Fig. 7).

2.4 Comparative Anaphora

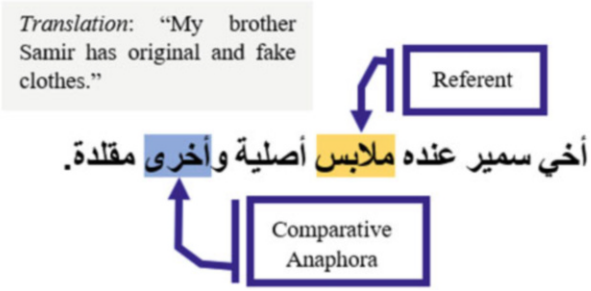
This type of anaphora is manifested by the introduction of lexical modifiers (e.g., اخر ، اخرى/other, وحدة/one) or comparative adjectives (أكبر من/greater than,

Table 4 Demonstrative pronoun

	Pronom					
	Singular		Dual		Plural	
Refer	M	F	M	F	M	F
Near person or thing	ذا هـذا،	تـا، تـي، ذي، ذِهِ، ذِهه، تــهْه ، تــه، هــتـا، هـذهِ، هــاتـه	هذين هذان،	هاتين هاتان ،	هــؤؤاء ووا ء،	
Far/distant person or thing	ذاك، هــذاك اك، ذذِكِ،	هــاتــيــك تتك، تــيــك،	ذانــك	تــانــك	أووالـك أووائــك،	

Table 5 Demonstrative pronoun (place)

Refer	Pronom
Near place	هنا
Far/distant place	هناك ، هنالك



Translation: "My brother Samir has original and fake clothes."

أخي سمير عنده ملابس أصلية وأخرى مقلدة.

Fig. 8 Example of comparative anaphora

أحسن من/better than) [11]. This variety of anaphora indicates a relation such as set-complement, similarity, and comparison between the anaphora and the referent (Fig. 8) [12].

3 Related Work

According to a review of the literature, the majority of studies on anaphora resolution (AR) in Arabic focus on the pronominal, which is the most common type of anaphora in Arabic texts (90%) [17]. We can divide the work into three categories in terms of resolution approaches: rule-based (symbolic and knowledge-rich approaches), machine learning (knowledge-poor approaches), and hybrid approach.

The rule-based approach is mainly based on setting up linguists rules using a set of morphological, syntactic, semantic, and even pragmatic features in order to determine the link between the anaphora and its referent. Mitkov et al. [18] start the contribution in the anaphora resolution and particularly with pronominal type by using some indicators such as number-gender, referential distance, definiteness, and others to calculate a score that preferred candidate than others to determine the best antecedent. Abolohom and Omar [19] tackled the pronominal type by using different linguistic rules depending on the morphological, lexical, heuristic, syntactic, and positional constraints. The experiments are conducted using the Quran corpus, and as a result, the model achieves an accuracy rate of 84,43%. Nabil and Saad [4] presented a resolution strategy to four Arabic anaphora types which are nominative disjoint personal pronouns, accusative disjoint personal pronouns, genitive joint personal pronouns which are attached to nouns and particles not verbs, and relative pronouns. The authors use syntactic rules for the selection of antecedents. The proposed method allows to obtain a success rate of 86% using the LDC Arabic Treebank Part 3.

Secondly, the objective of the machine learning approach is to identify anaphora-antecedent pairs by means of simple co-occurrence rules and a previously annotated corpus to train the resolution model. Abolohom et al. [20] presented a solution based on the extraction of a new set of computational and linguistic features and then using one of the multiple classifiers (the maximum entropy, decision tree, k-nearest neighbor, and stacking methods). The experiments were done using the Quran corpus, and they obtained the best result by combining the four classifiers with F1 score equal to 93.50%. Trabelsi et al. [9] used Markov decision process (MDP) and reinforcement learning for the pronominal type, and they got 83.33% accuracy rate. Bouzid and Zribi [17] combine a reinforcement learning method based on Q-learning algorithm and a word embedding model, to deal with pronominal anaphora and zero anaphora, and they achieved a precision rate of 79.37% case of pronominal anaphora and 70.82% case of zero anaphora. Elghamry et al. [21] proposed an algorithm dedicated to pronominal type based on collocational evidence, recency, and bands as AR-related features. As a result, the algorithm achieves an F-measured rate of 87.6%. Hammami et al. [22] also contributed in pronominal type using machine learning approach and a set of new features specific for Arabic language. They got an F-measure of 86.2% for the genre of technical manuals, 84.5%for newspaper articles, and 72.1% for the literary texts.

Finally, the hybrid approach combines the advantages of the two approaches mentioned above, which will give a better coverage of the anaphoric phenomenon, thus increasing the performance of the resolution systems. Abolohom and Omar [21] proposed a resolution approach for the three types of pronominal anaphora, namely, third-person pronouns, reflexive pronouns, and possessive pronouns. The authors combined the rule-based approach (using morphological and syntactic filter) and the machine learning approach (using k-nearest neighbor approach). The resolution approach was performed using the Quranic annotated corpus and obtained 74.80% F-score.

4 Arabic Anaphoric Resolution Challenges

Arabic is a semitic language with a very specific structure and characteristics (morphological richness and syntactic flexibility) that make the process of anaphoric resolution difficult. The aim of this section is to present the main factors which affect anaphoric resolution process.

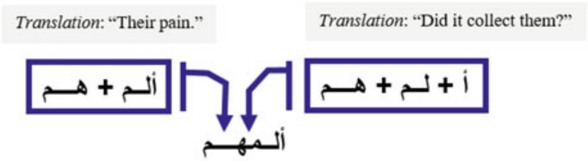

Translation: "Their pain."
Translation: "Did it collect them?"
ألـم + هـم
أ + لـم + هـم
ألـمهـم

Fig. 9 Example of incorrect segmentation

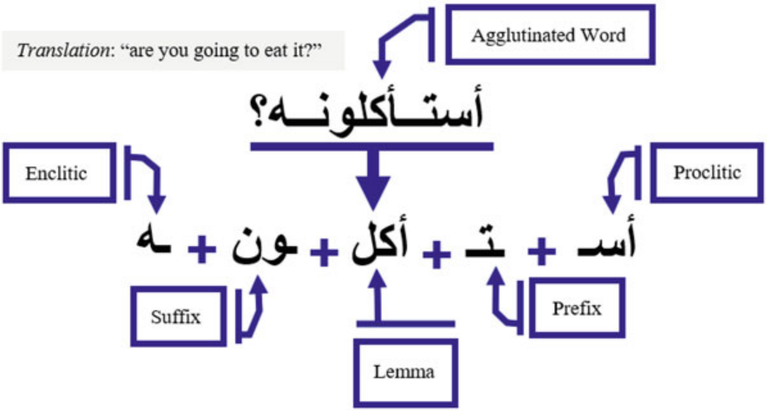


Fig. 10 Example of agglutination

4.1 Lack of Diacritical Marks

Without diacritical marks, an Arabic text is extremely unclear (morphologically and grammatically). According to [23], 74% of Arabic words might potentially take several lexical diacritization, making it difficult to determine if the anaphoric phenomenon or referent is the case (Fig. 9).

4.2 Agglutination Phenomenon

The Arabic writing structure is characterized by the agglutination phenomena which is explained by the fact of combining numbers of words in just one (Fig. 10). Compared to French or English, an Arabic word can sometimes correspond to a full sentence [17].

This characteristic generates morphological ambiguities during the analysis. For that, we have to break up (tokenize) the phrase and start the process of resolution to solve it.

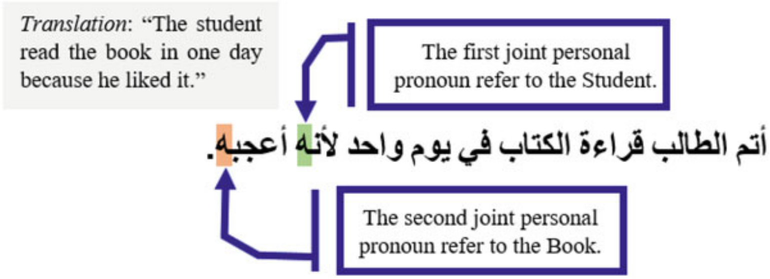

Translation: "The student read the book in one day because he liked it."
The first joint personal pronoun refer to the Student.
أتم الطالب قراءة الكتاب في يوم واحد لأنه أعجبه.
The second joint personal pronoun refer to the Book.

Fig. 11 Example of ambiguity of the referent

4.3 *Syntactic Flexibility (Words Free Order)*

Arabic is a nearly free-order language. This order causes artificial syntactic ambiguities, since the grammar should provide all the possible combination rules for reversing the order of words in the sentence (Table 6). For anaphora resolution, this type of flexibility is a problem for referent localization [13, 24].

4.4 *Ambiguity of the Referent*

This difficulty occurs when the referent is ambiguous (due to the presence of two or more referents for the same anaphora). In this case, external knowledge of the context is necessary to identify the correct referent (Fig. 11) [25].

4.5 *Hidden Referent*

This case occurs when the anaphora refers to something which is not present in the sentence or text. The Quranic text is an example where this phenomenon persists [26], so in the example below, the pronominal anaphora (هو/he) refers to (الله/Allah) which is not present in the "Aya." The human through his knowledge and reasoning system can easily make the connection between the pronominal anaphora (هو/he) and (الله/Allah). However, for anaphoric resolution systems, the task is complicated (Fig. 12).

Table 6 Example of words free order in Arabic sentences

Example of Arabic sentences	English translation	Order	Observation
كتب أسامة الرسالة	Oussama wrote the letter	VSO	/
أسامة كتب الرسالة	Oussama wrote the letter	SVO	/
الرسالة أسامة كتبها	The letter Oussama wrote it	OSV	Joint personal pronoun “ ها ” are anaphoric
كتبها الرسالة أسامة	The letter was written by Mohamed	VOS	Joint personal pronoun “ ها ” are cataphoric

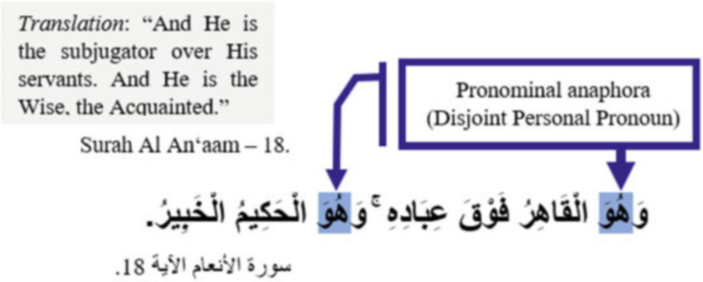

Translation: "And He is the subjugator over His servants. And He is the Wise, the Acquainted."
Surah Al An'aam – 18.
Pronominal anaphora (Disjoint Personal Pronoun)
وَهُوَ الْقَاهِرُ فَوْقَ عِبَادِهِ ۚ وَهُوَ الْحَكِيمُ الْخَبِيرُ.
سورة الأنعام الآية 18.

Fig. 12 Example of hidden referent

4.6 Lack of Annotated Corpora with Anaphoric Links

The lack of annotated Arabic corpus is due to the significant effort, time, and complexity required by a human annotator to annotate a large-volume corpus. This lack implied a limited amount of progress in Arabic natural language processing [27].

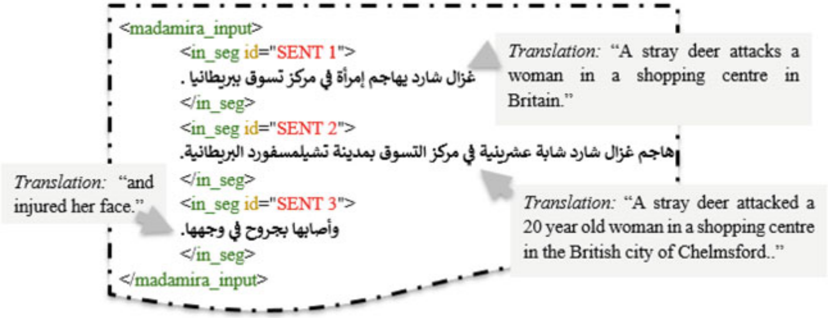
5 The A^3T Architecture

The purpose of this research is to solve the anaphora problem by addressing two types of anaphora: pronominal and verbal anaphora. Furthermore, the community will have access to a new annotated resource that may be used in the automatic Arabic anaphora resolving mechanism. We created a system based on a rule-based approach and a user-friendly interface that allows computer scientists and linguists to easily track the process and intervene during the annotation phase.

We'll go over the system's modules (A^3T) in this part and explain each step and its outcome. We considered dividing the process of creating our system environment into four (04) stages: preprocessing, anaphora and candidate identification, anaphora resolution, and finally corpus annotation. Each module is made up of crucial steps that must be completed in order for the module's purpose to be achieved.

5.1 Preprocessing

Sentence tokenization is used first to split the text file into sentences for which POS and morphological analysis will be applied. The following step is to determine the grammatical category and other morphological aspects of a given word (Fig. 14), such as gender, number, condition, and voice. Due to its precision and high-quality word-level disambiguation of Arabic text and also based on some experiments, the



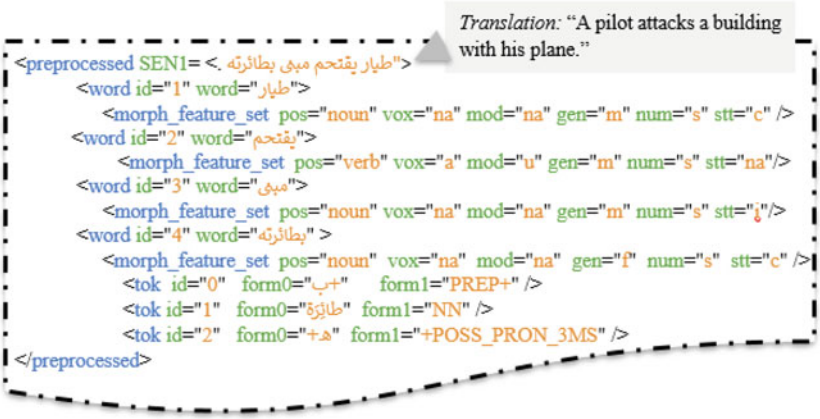
```
<madamira_input>
        <in_seg id="SENT 1">
        غزال شارد يهاجم إمرأة في مركز تسوق ببريطانيا .
        </in_seg>
        <in_seg id="SENT 2">
        هاجم غزال شارد شابة عشرينية في مركز التسوق بمدينة تشيلمسفورد البريطانية.
        </in_seg>
        <in_seg id="SENT 3">
        وأصابها بجروح في وجهها.
        </in_seg>
</madamira_input>
```

Translation: "A stray deer attacks a woman in a shopping centre in Britain."

Translation: "A stray deer attacked a 20 year old woman in a shopping centre in the British city of Chelmsford.."

Translation: "and injured her face."

Fig. 13 Data preparation



Translation: "A pilot attacks a building with his plane."

```
<preprocessed SEN1= <. طيار يقتحم مبنى بطائرته">
      <word id="1" word="طيار">
        <morph_feature_set pos="noun" vox="na" mod="na" gen="m" num="s" stt="c" />
      <word id="2" word="يقتحم">
          <morph_feature_set pos="verb" vox="a" mod="u" gen="m" num="s" stt="na"/>
      <word id="3" word="مبنى">
        <morph_feature_set pos="noun" vox="na" mod="na" gen="m" num="s" stt="i"/>
      <word id="4" word="بطائرته" >
        <morph_feature_set pos="noun" vox="na" mod="na" gen="f" num="s" stt="c" />
          <tok id="0" form0="+ب" form1="PREP+" />
          <tok id="1" form0="طائِرَة" form1="NN" />
          <tok id="2" form0="+ه" form1="+POSS_PRON_3MS" />
</preprocessed>
```

Fig. 14 POS output (MADAMIRA)

MADAMIDA tool [28] was chosen for our purposes. The word-level disambiguation feature will assist us in determining the attached pronouns.

5.2 *Anaphora and Candidate Identification*

Anaphoras are identified using the MADAMIRA tag set's grammatical code. The result is a list of all anaphoras in the text, along with information such as ID, name, gender, number, and sentence number. For the pronominal case, the process differs from one type to another, for example, the POS tagging for the attached pronominal anaphora does not contain a tag for gender, number, or person, and we should utilize a split mechanism to place each of them in their proper tag, as shown in Fig. 15.

On the other hand, for verbal anaphora identification, we take all of the features used for pronominal anaphora, such as gender, number, and so on, as shown in

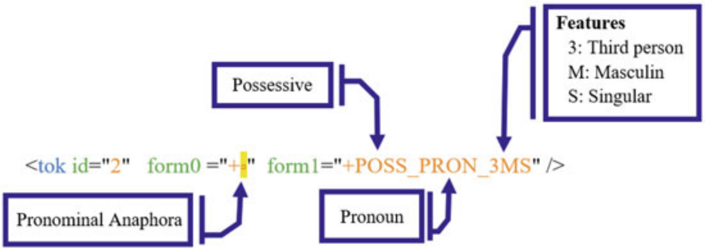


Fig. 15 Pronominal anaphora identification

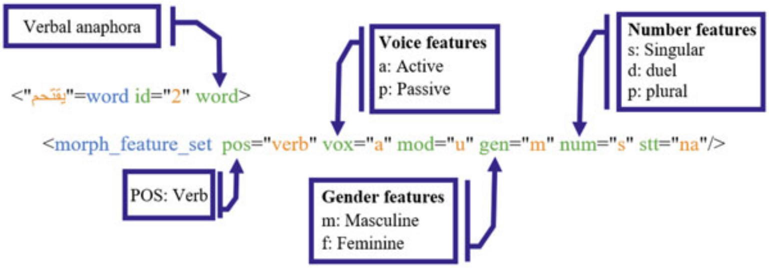


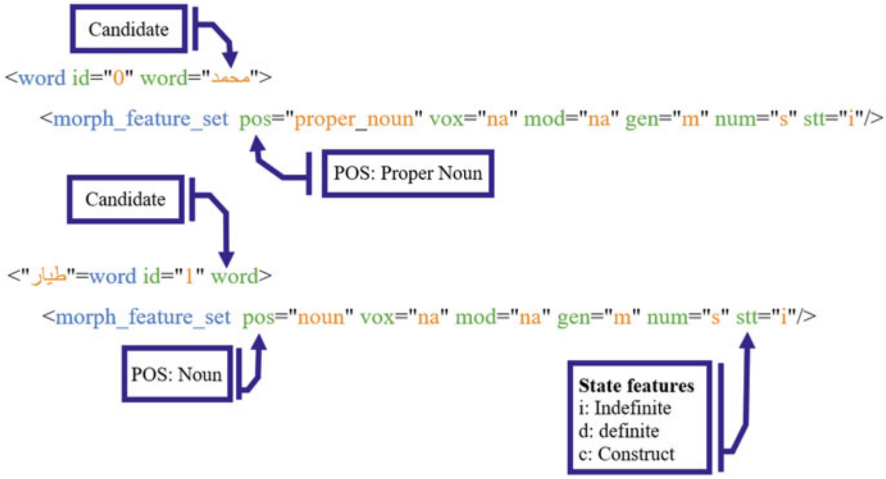
Fig. 16 Verbal anaphora identification

Fig. 16, and we add another feature, the voice feature (active or passive form), which will aid in resolution.

Candidates are chosen based on their POS (nouns, NPs, and proper noun) (Fig. 17) and a specific search scope adjusted based on some tests and previous research [29]. In the case of anaphora, the search scope is still unknown; however, based on analysis, a large number of references appear in the two preceding phrases. In our situation, we took two sentences before and the same number after as a special case for demonstrative type. In the case of verbal anaphora, we took two sentences following the verb in the active form and two sentences before the verb in the passive or unknown form. All of candidate's characteristics are taken into account, including gender, number, voice, definiteness, and sentence number.

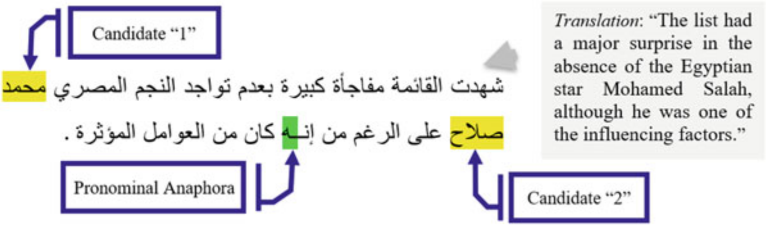
5.3 *Anaphora Resolving*

This is the most crucial module in the system because it will have a direct impact on the system's accuracy. The purpose of this module is to select the most appropriate referent for each anaphora from among the most likely candidates. To determine the proper referent, we applied a set of favorable features, as indicated in Table 7, that can favor certain candidates over others.



```
<word id="0" word="محمد">
    <morph_feature_set pos="proper_noun" vox="na" mod="na" gen="m" num="s" stt="i"/>
<"طيار"=word id="1" word>
    <morph_feature_set pos="noun" vox="na" mod="na" gen="m" num="s" stt="i"/>
```

Fig. 17 Identification of noun and proper noun candidates



شهدت القائمة مفاجأة كبيرة بعدم تواجد النجم المصري محمد
صلاح على الرغم من إنـه كان من العوامل المؤثرة .

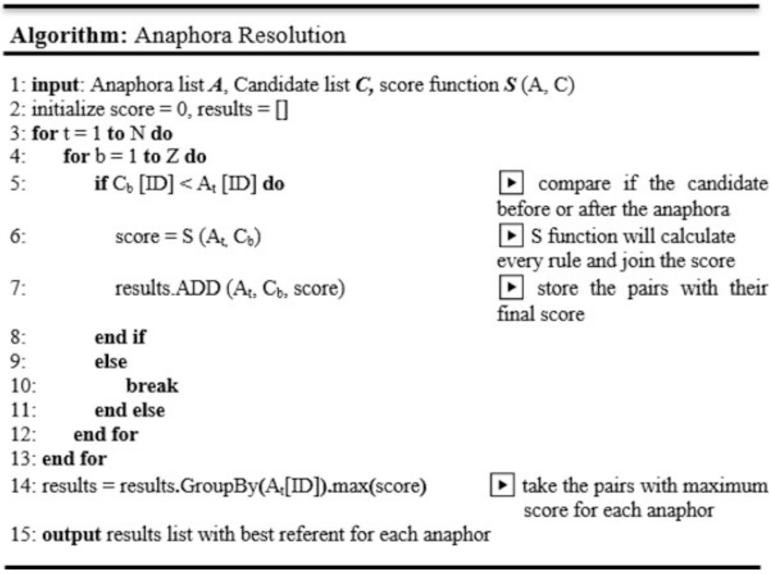
Translation: "The list had a major surprise in the absence of the Egyptian star Mohamed Salah, although he was one of the influencing factors."

Fig. 18 Score similarity (example in pronominal case)

For each rule, we assigned a score value (Fig. 18). Each score was determined by a series of trials that took into account past research [19]. For all candidates, we calculated the scores for each rule and joined them to each candidate's previous score. As shown in Fig. 19, all of the candidates were given a ranking, and the one with the highest total score was recommended as a good referent. As seen in Fig. 18, we chose the one that came closest to overcoming the score similarity.

5.4 *Automatic Text Annotation*

As previously stated, the system provides to the linguistic experts a user-friendly interface that allows them to check and, if necessary, modify the links between anaphora and its referent, resulting in a reliable corpus that can be used in future studies.



Algorithm: Anaphora Resolution

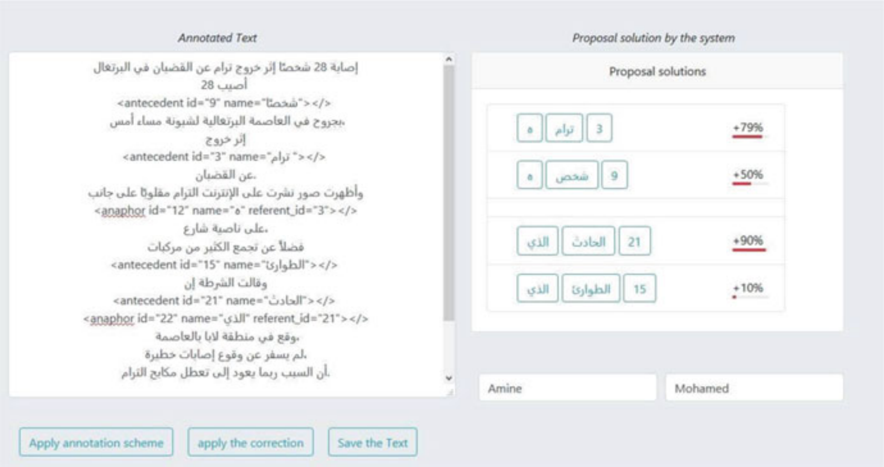
```
input: Anaphora list A, Candidate list C, score function S (A, C)
initialize score = 0, results = []
for t = 1 to N do
    for b = 1 to Z do
        if $C_b$ [ID] < $A_t$ [ID] do                ▶ compare if the candidate before or after the anaphora
            score = S ($A_t$, $C_b$)                 ▶ S function will calculate every rule and join the score
            results.ADD ($A_t$, $C_b$, score)        ▶ store the pairs with their final score
        end if
        else
            break
        end else
    end for
end for
results = results.GroupBy($A_t$[ID]).max(score)     ▶ take the pairs with maximum score for each anaphor
output results list with best referent for each anaphor
```

Fig. 19 Anaphora resolution algorithm

Table 7 The linguistic preferences and their respective scores

Linguistic rules	Description
Definiteness	A score of 1 is given if an NP is definite and of 0 if not
Recency	A score of 1 is assigned to the recency NP to the anaphora and 0 if not
Referential distance	A score of 2 is assigned to NPs in the previous sentence or two sentences, and further than those are given 0
NPs in the title	A score of 1 is issued to the existing NP in title and 0 if not
Grammatical function	Scores of 1 are given to an NP that has the same syntactic structure as the anaphora and 0 if not
NP frequency	A score of 2 is assigned to the most frequent NP in text and 0 if not
First noun phrases	A score of 1 is issued to the first NP of each sentence and 0 if not

More specifically, the interface (Fig. 20) displays the automatic annotated text in the center, while all of the couples anaphora/referent are shown on the right, with the system's chosen couple. In this case, all the expert has to do is to check whether the anaphora tag's number of referent matches the correct one, and if not, he may adjust the number of referent to the correct one from the other suggested couples or create a new one if the system doesn't find out the correct antecedent. Finally, the A^3T will generate an XML file that contains the text with anaphoric relationship tags as shown in Fig. 21.



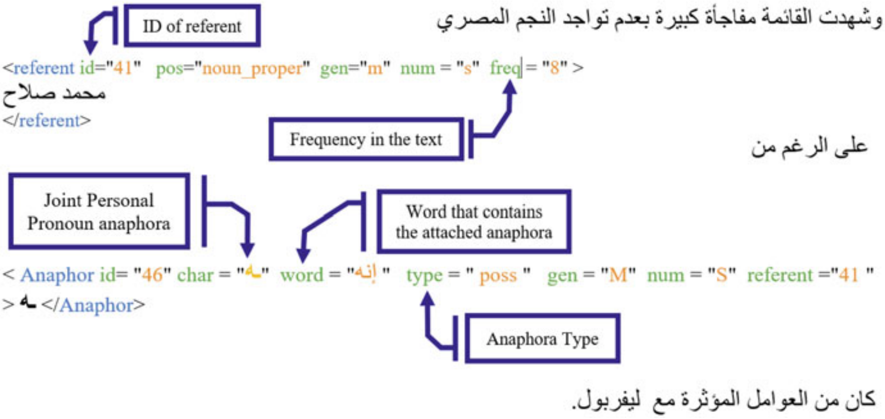
```
إصابة 28 شخصًا إثر خروج ترام عن القضبان في البرتغال
أصيب 28
<antecedent id="9" name="شخصًا"></>
.بجروح في العاصمة البرتغالية لشبونة مساء أمس
إثر خروج
<antecedent id="3" name="ترام "></>
.عن القضبان
وأظهرت صور نشرت على الإنترنت الترام مقلوبًا على جانب
<anaphor id="12" name="ه" referent_id="3"></>
.على ناصية شارع
فضلاً عن تجمع الكثير من مركبات
<antecedent id="15" name="الطوارئ"></>
وقالت الشرطة إن
<antecedent id="21" name="الحادث"></>
<anaphor id="22" name="الذي" referent_id="21"></>
.وقع في منطقة لابا بالعاصمة
.لم يسفر عن وقوع إصابات خطيرة
.أن السبب ربما يعود إلى تعطل مكابح الترام
```

ه	ترام	3	+79%
ه	شخص	9	+50%
الذي	الحادث	21	+90%
الذي	الطوارئ	15	+10%

Amine

Mohamed

Fig. 20 The friendly interface for the annotator


ID of referent
Frequency in the text
Joint Personal Pronoun anaphora
Word that contains the attached anaphora
Anaphora Type

```
وشهدت القائمة مفاجأة كبيرة بعدم تواجد النجم المصري
<referent id="41"  pos="noun_proper"  gen="m"  num = "s"  freq = "8" >
محمد صلاح
</referent>
على الرغم من
< Anaphor id= "46" char = "ـه"  word = "إنه "  type = " poss "  gen = "M"  num = "S"  referent ="41 "
> ـه </Anaphor>
كان من العوامل المؤثرة مع ليفربول.
```

Fig. 21 Example from the annotated text for the pronominal case

6 Experiments and Results

There are a lot of works in the field of anaphora resolution, notably with the pronominal type, as noted in the related work section; however, there is fewer corpus to use in the evaluation. We found Holy Qur'an corpus [29] and QurAna Corpus QAC [3] for the Quran script and AnATAr corpus [11] which is based on Tunisian education books and addresses the pronominal type and Hadder [4] with textbooks and newspapers, which tackles the zero anaphora. In our research, we used the "AnATAr" corpora for the evaluation of both anaphora phenomena. We used the standard accuracy metric to calculate the efficiency of the A^3T, and Table 8 presents

Table 8 The result of anaphora resolution system A^3T on AnATAr corpora

Anaphora type	Corpus	Accuracy achieved
Pronominal	AnATAr	83.19
Verbal	AnATAr	57.23

the results obtained. We were unaware of any prior works on verbal anaphora, so we tested our work by taking the help of a linguistic specialist and utilizing the same corpus.

7 Discussion

After analyzing the output of our system, particularly for the verbal anaphora, we found some factors that have influenced our findings. The first factor is the limitation of the word disambiguation tool that fails to extract the right meaning of words that have more than one, for example: the word "سموه" can be understood as a verb (name, designate) or a noun (Highness, grace). The second factor is the search scope, which could also lead to the best referent being excluded from the list of referents due to being out of scope. In the automatic resolution, the tool rid the references that span multiple sentences, but we correct this issue in the expert verification part. The third factor is that the MADAMIRA tool can't recognize composed words like "جمهورية مصر العربية" (Arab Republic of Egypt) or even compound proper names that always occur together like "محمد صلاح" (Mohamed Salah). Finally, in some situations, the voice feature causes a faulty judgment when deciding if the better referent occurs before or after the verb anaphora. As previously stated, we discovered that our system produced satisfactory results with the pronominal type but struggled with the resolution of the verbal type. We decided to use this system to annotate a collected text corpus with anaphora links under the supervision of a linguistic expert in order to produce a trustworthy corpus for the community. The following research paper [30], which explains the procedure from collection through annotation, gives more information about the size of the corpus as well as the categorizations that it comprises.

8 Conclusion

Anaphora plays an important role in understanding text and making it coherent, and at the same time, it is still a challenging task in Arabic language due to the complexity of language and the lack of tools and linguistic resources. By proposing a resolution method that uses several aspects to locate the relationships between pronominal and verbal anaphora and their referent in Arabic text, our current work will make a contribution to the field of anaphora resolution in Arabic. In addition, we provide a contribution to linguistic resources by offering an annotated corpus that includes both types of anaphora. To save time and effort during the resolution and annotation phases, we developed A^3T, a system that employs linguistic concepts to resolve and annotate this phenomenon. We are confident that the corpus with the assistance of an expert will be highly valuable in developing intelligence tools to address the Arabic anaphora problem. For the future, our focus will be on improving the verbal resolution mechanism using cutting-edge computational linguistic techniques and methodologies while also expanding the size of the corpus.

References

1. Halliday, M.A.K., Hasan, R.: Cohesion in english. Routledge (2014)
2. D'Souza, J., Ng, V.: Anaphora resolution in biomedical literature: a hybrid approach. In: Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine (pp. 113–122) (2012, October)
3. Cambria, E., Das, D., Bandyopadhyay, S., Feraco, A.: Affective computing and sentiment analysis. In: A Practical Guide to Sentiment Analysis (pp. 1–10). Springer, Cham (2017)
4. El-Said Nada, A.N.M., Saad, S., El-Magd Al-Ansary, A.: A syntactic based approach to anaphora resolution in Arabic. In: Proceeding of the Eighteenth Conference on Language Engineering (ESOLEC'18) (2018)
5. Phadke, M., Devane, S.: Pronoun resolution task for multilingual machine translation. In: 5th International Conference on Next Generation Computing Technologies (NGCT-2019) (2020)
6. Antunes, J., Lins, R.D., Lima, R., Oliveira, H., Riss, M., Simske, S.J.: Automatic cohesive summarization with pronominal anaphora resolution. Comput. Speech Lang. **52**, 141–164 (2018)
7. Matysiak, I.: Information extraction systems and nominal anaphora analysis needs. In: Proceedings of the International Multiconference on Computer Science and Information Technology (pp. 183–192) (2007)
8. Annam, V., Koditala, N., Mamidi, R.: Anaphora resolution in dialogue systems for south asian languages. Preprint (2019). arXiv:1911.09994
9. Trabelsi, F.B.F., Zribi, B.O., C., Mathlouthi, S.: Arabic anaphora resolution using Markov decision process. In: Gelbukh, A. (ed.) Computational Linguistics and Intelligent Text Processing. CICLing 2016 Lecture Notes in Computer Science, vol. 9623, pp. 520–532. Springer, Cham (2016)
10. Hamouda, W.: Anaphora resolution for Arabic machine translation: a case study of nafs (Doctoral dissertation, Newcastle University) (2014)
11. Hammami, S., Belguith, L., Ben Hamadou, A.: Arabic anaphora resolution: corpora annotation with coreferential links. Int. Arab J. Inf. Tech. (IAJIT) 6(5), 480–488 (2009)
12. Seddik, K.M., Farghaly, A.: Anaphora resolution. Natural Language Processing of Semitic Languages (pp. 247–277). Springer, Berlin, Heidelberg (2014)
13. Beseiso, M., Al-Alwani, A.: A coreference resolution approach using morphological features in arabic. Int. J. Adv. Comput. Sci. Appl. **7**(10), 107–113 (2016)
14. Mathlouthi, S., Ben, F., Trabelsi, F., Zribi, C.B.O.: A novel approach based on reinforcement learning for anaphora resolution. In: 28th IBIMA Conference (2016, November)
15. Bouzid, S.M., Trabelsi, F.B.F., Zribi, C.B.O.: How to combine salience factors for arabic pronoun anaphora resolution. In: 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA) (pp. 929–936). IEEE (2017, October)

16. Jarbou, S.O.: Time frame as a determinant of accessibility of anaphoric demonstratives in classical arabic. Top. Linguist. **19**(2), 57–71 (2018)
17. Bouzid, S.M., Zribi, C.B.O.: A generic approach for pronominal anaphora and zero anaphora resolution in arabic language. Procedia Comput. Sci. **176**, 642–652 (2020)
18. Mitkov, R., Belguith, L.H., Stys, M.: Multilingual robust anaphora resolution. In: Proceedings of the Third Conference on Empirical Methods for Natural Language Processing (pp. 7–16) (1998, June)
19. Abolohom, A., Omar, N.: A computational model for resolving arabic anaphora using linguistic criteria. Indian J. Sci. Tech. **10**(3), 1–6 (2017)
20. Abolohom, A., Omar, N., Pais, S., Cordeiro, J.: A comparative study of linguistic and computational features based on a machine learning for arabic anaphora resolution. Procedia Comput. Sci. **189**, 37–47 (2021)
21. Elghamry, K., Al-Sabbagh, R., El-Zeiny, N.: Arabic anaphora resolution using web as corpus. In: Proceedings of the Seventh Conference on Language Engineering, Cairo, Egypt (2007, December)
22. Hammami, S.M., Belguith, L.H.: Arabic pronominal anaphora resolution based on new set of features. In: International Conference on Intelligent Text Processing and Computational Linguistics (pp. 533–544). Springer, Cham (2016, April)
23. Debili, F., Achour, H.: Voyellation automatique de l'arabe. In: Computational Approaches to Semitic Languages (1998)
24. Fotiadou, G., Muñoz, A.I.P., Tsimpli, I.: Anaphora resolution and word-order across adulthood: Ageing effects on online listening comprehension (2020)
25. Mitamura, T., Nyberg, E., Torrejon, E., Svoboda, D., Brunner, A., Baker, K.: Pronominal anaphora resolution in the KANTOO multilingual machine translation system. In: Proceedings of the 9th Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages (2002)
26. Seddik, K.M., Farghaly, A.: Arabic anaphora resolution using Holy Qur'an text as corpus. In: Proceeding of Arabic Language Technology International Conference (ALTIC) (2011)
27. Seddik, K.M., Farghaly, A.: Anaphora resolution. In: Natural Language Processing of Semitic Languages (pp. 247–277). Springer, Berlin, Heidelberg (2014)
28. Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., ..., Roth, R.: Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14) (pp. 1094–1101) (2014, May)
29. Mitkov, R.: Anaphora Resolution: The State of the Art (pp. 1–34). School of Languages and European Studies, University of Wolverhampton (1999)
30. Dahou, A.H., Abdelmoazz, M., Cheragui, M.A.: A3C: Arabic anaphora annotated corpus. ICNLSP **2021**, 117 (2021)

A Commonsense-Enhanced Document-Grounded Conversational Agent: A Case Study on Task-Based Dialogue

Carl Strathearn and Dimitra Gkatzia

Abstract This paper argues that future dialogue systems must retrieve relevant information from multiple structured and unstructured data sources in order to generate natural and informative responses as well as exhibit commonsense capabilities and flexibility in dialogue management. To this end, we firstly review recent methods in document-grounded dialogue systems (DGDS) and commonsense-enhanced dialogue systems and then demonstrate how these techniques can be combined in a unified, commonsense-enhanced document-grounded dialogue system (CDGDS). As a case study, we use the `Task2Dial` dataset,[1] a newly collected dataset which contains instructional conversations between an information giver (IG) and information follower (IF) in the cooking domain. We then propose a novel architecture for commonsense-enhanced document-grounded conversational agents, demonstrating how to incorporate various sources to synergistically achieve new capabilities in dialogue systems. Finally, we discuss the implications of our work for future research in this area.

1 Introduction

Much of the work in dialogue systems has focused on developing task- and goal-oriented conversational agents that are capable of completing tasks, such as making restaurant reservations, ordering transport services and booking travel [1]. Traditionally, dialogue systems utilise domain-specific database schemas [2] and focus on slot-filling response generation. However, encoding all available information can be prohibitive in most domains, as the majority of domain knowledge exists in

[1] https://huggingface.co/datasets/cstrathe435/Task2Dial/tree/main.

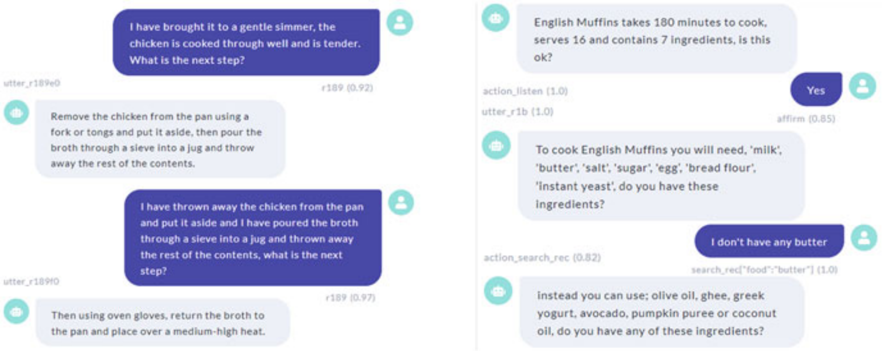
C. Strathearn (✉) · D. Gkatzia
Edinburgh Napier University, Edinburgh, UK
e-mail: c.strathearn@napier.ac.uk; d.gkatzia@napier.ac.uk

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_6

some unstructured format, such as documents [3]. DGDS can provide opportunities for dialogue systems that were not possible before, such as answering questions based on the information provided in documents and imitating the human capacity to possess background knowledge. Recent work on DGDS has focused on question-answering (Q&A) and machine reading comprehension. For instance, CoQA [4], a Q&A task between two interlocutors who have access to the same passage, requires the receiver to comprehend the passage in order to ask questions. Other tasks have focus on commonsense reasoning. For instance, QuAC [5] follows a similar setting as CoQA; however, only the receiver has access to the passage, and the questioner asks questions based on the title of the passage alone.

Here, we focus on `Task2Dial` [6], a new task for CDGDS, which aims at generating instructions grounded in a document so that the receiver of the instructions can complete a task. Task2Dial is similar to QuAC in that the information giver (IG) has access to the underlying document. However, Task2Dial differs from QuAC, because the information follower (IF) can ask questions for answers which are not grounded to a specific document, requiring commonsense capabilities by both IG and IF. Task2Dial requires following steps in a pre-specified order, invoking every day communication characteristics, such as asking for clarification, questions or advice, which may require the use of commonsense knowledge to answer. The proposed task differs from existing document-grounded tasks, as answers may require commonsense knowledge generated from the underlying information that may not be present in the document. Inspired by previous work on document-grounded dialogue [3, 7, 8], commonsense-enhanced natural language generation (NLG) [9, 10] and Q&A [4, 5], neural referring expression generation [11], concept acquisition [12], and task-based/instructional dialogue [13], we aim to capture two different types of knowledge: (1) document-level procedural context, i.e. what is the next step, and (2) commonsense, i.e. answering questions that are not available in the document, as demonstrated in Fig. 1. Our task is designed as an instruction-following scenario with an information giver (IG) and an information follower (IF), inspired partly by the GIVE challenge [13]. The IG has access to the recipe and gives instructions to the IF. The IG might choose to omit irrelevant information, simplify the content or provide it as is. The IF will either 'follow' the task by confirming that they have understood an instruction or ask for further information. For this, the IG may have to rely on information outside the given document; in other words, the IG will rely on their commonsense to enhance understanding and success of the task. To explore this, we propose a novel conversational agent, `ChefBot`, to structure and control the flow and type of information provided to the user from the documents. This requires a cumulative approach to formatting additional data from documents, i.e. structuring and extracting metadata to create additional knowledge databases that contain information such as the utility of objects and alternative ingredients, while retaining the underlying sequential structure of the instructional document [14].

The `Task2Dial` dataset introduces new challenges for dialogue systems: (1) generating instructions for task completion requires a flexible dialogue manager, as following specific steps in the form of a checklist might invoke discourse



I have brought it to a gentle simmer, the chicken is cooked through well and is tender. What is the next step?

r189 (0.92)

utter_r189e0

Remove the chicken from the pan using a fork or tongs and put it aside, then pour the broth through a sieve into a jug and throw away the rest of the contents.

I have thrown away the chicken from the pan and put it aside and I have poured the broth through a sieve into a jug and thrown away the rest of the contents, what is the next step?

r189 (0.97)

utter_r189f0

Then using oven gloves, return the broth to the pan and place over a medium-high heat.

English Muffins takes 180 minutes to cook, serves 16 and contains 7 ingredients, is this ok?

action_listen (1.0)

Yes

affirm (0.85)

utter_r1b (1.0)

To cook English Muffins you will need, 'milk', 'butter', 'salt', 'sugar', 'egg', 'bread flour', 'instant yeast', do you have these ingredients?

I don't have any butter

search_rec["food":"butter"] (1.0)

action_search_rec (0.82)

instead you can use; olive oil, ghee, greek yogurt, avocado, pumpkin puree or coconut oil, do you have any of these ingredients?

Fig. 1 Excerpt from dialogue showing the commonsense handling of hot objects using the Task2Dial dataset and ChefBot. Left: commonsense handling of objects. Right: swapping ingredients for appropriate alternatives using custom actions

phenomena not present in other dialogue styles, such as paraphrasing, as instructional responses may have been modified from the underlying document, and interlocutors may ask for clarification or alternative steps; (2) hence, this task requires commonsense knowledge, since questions may not necessarily be grounded in the document; (3) generating requires planning based on context, as task steps need to be provided in order; and finally (4) Task2Dial's human reference texts show more lexical richness and variation than other document-grounded dialogue datasets. The Task2Dial dataset contains dialogues with an average 18.15 number of turns and 19.79 tokens per turn, as compared to 12.94 and 12, respectively, in existing datasets. Therefore, developing a conversational agent based on this new task requires flexible dialogue management with global and domain-specific intents to enhance natural communication, custom actions to swap ingredients and explain unknown objects and rule-based state tracking for sequential and non-sequential information giving. For instance, it is not enough for the agent to just 'read' the next recipe instruction—the conversation might briefly diverge from the current plan to provide information about an ingredient replacement, and then it will have to correctly resume the previous conversation.

To this end, our contributions to the field can be summarised as follows:

- We propose a new task, Task2Dial, for commonsense-enhanced document-grounded dialogue.
- We present a novel dataset for commonsense-enhanced document-grounded dialogue.
- We propose a novel conversational agent architecture which considers how elements of the documents are represented within the dialogue manager, i.e. intents, utterances, entities and actions, and how the data is labelled to enable the system to follow the sequential logic of a given recipe while remaining flexible in terms of topic switch.

In the next section (Sect. 2), we refer to the related work. The proceeding sections cover the task formulation and data curation methodology (Sect. 6) and present an analysis of the Task2Dial dataset and a comparison to related datasets (Sect. 4). Finally, Sect. 5 proposes a novel conversational agent architecture for addressing the task of CDGDS, and finally, in Sect. 6, we discuss the implications and challenges for the development of instruction-giving dialogue systems for real-world tasks.

2 Related Work

The work presented in this paper focuses on the development of a CDGDS conversational agent for instruction-giving task-based dialogue, which is relevant to several areas of research on task- and goal-oriented dialogue, state tracking, document-grounded dialogue, commonsense reasoning and dialogue management. Next, we review each of these areas.

2.1 Task- and Goal-Oriented Dialogue

In dialogue management, task-oriented approaches focus on the successful completion of the individual stages of a task, towards achieving an end goal [15]. Comparatively, goal-oriented approaches focus on comparing the outcome or overall performance against a gold standard [16]. Task- and goal-oriented dialogue systems are common in domains such as booking and reservation systems for businesses [17]. However, business models are typically goal-oriented as the instructions are minimal and the focus is on the outcome [18]. Instead, the Task2Dial task is formulated as a task-oriented dialogue paradigm to imitate real-world practical scenarios that can vary in complexity and require adaptability, additional information, clarification and natural conversation in order to enhance understanding and success.

2.2 Dialogue State Tracking and Planning

Task-based dialogue systems require the user and artificial agent to work synergistically by following and reciting instructions to achieve a goal. Human-bot conversational models are defined as follows [19]:

- **Single intent and single turn policy:** relies solely on question and answer pairs assuming that the user provides all slot values in a single utterance. This type of task does not require dialogue state tracking.

- **Single intent and multi-turn policy:** extends the previous conversational model; however, this model can include multiple turns, to fill in missing information. Historic information is then extracted from all turns and used to structure data.
- **Multi-intent and multi-turn policy:** the intents can change depending on the context.

Instruction-giving scenarios follow the *multi-intent multi-turn* conversational framework, since they must accommodate knowledge and variability outside of a linear deterministic model as practical tasks can vary in complexity and the conversation can vary based on the interlocutors' prior knowledge and experience. In addition, there is no restriction on the amount of variability introduced into a task, such as introducing alternate methods, commonsense knowledge and concepts that change the structure and information within the dialogue. Variability is often reduced in human-machine scenarios as systems are limited in knowledge and their ability to respond to questions not seen in training [20], which can result in shortened responses and fewer questions asked on aspects of the task [21]. This reduces the system's ability to ensure that the IF has understood the IG's directions, which may produce irregular outcomes or result in an incomplete task. Therefore, capturing and emulating natural variability within the dialogue is crucial for creating robust and reliable conversational systems for instruction-giving scenarios.

Existing datasets such as the Multi-Domain Wizard-of-Oz (MultiWOZ) [22], Taskmaster-1 [21], Doc2dial [3] and Action-Based Conversations Dataset (ABCD) [23] strictly follow the sequential logic of an instructional document. However, in addition to grounded information in documents, Task2Dial aims to accommodate questions and clarification on different aspects of a task that might not be grounded in the document. In previous work, the user is limited to the path of the subroutine; however, in Task2Dial, the IF can ask the IG questions at any stage of the task, regardless of the position within a given sequence, and then return to that position after the question is fulfilled. For example, in a cooking scenario, the IF may ask the IG how to use a certain kitchen utensil. The IG would need to answer this question and then return to the correct stage in the recipe in order to continue the sequence. This introduces additional challenges for state tracking. The conversational agent must not only generate instructions sequentially, based on the schema of a document, but also request confirmation to ensure that the user has understood the task and answer questions outside its predefined script. Using document-grounded subroutines to capture intents that change the direction of a task broadens the interaction between the IG and IF [23] and introduces new challenges for dialogue state tracking.

2.3 Document-Grounded Dialogue

DGDS classify unstructured, semi-structured and structured information in documents to aid in understanding human knowledge and interactions, creating greater naturalistic human-computer interactions (HCI) [24]. The aim of DGDS is to formulate a mode of conversation from the information (utterances, turns, context, clarification) provided in a document(s) [25]. DGDS are particularly useful in task-oriented and goal-oriented scenarios as they emulate the natural dialogue flow between the IG and IF. A recent example of DGDS and closest to our work is Doc2Dial, a multi-domain DGDS dataset for goal-oriented dialogue modelled on hypothetical dialogue scenes (dialogue act, a role such as user or agent and a piece of grounding content from a document) and dialogue flows (a sequence of dialogue scenes) to simulate realistic interactions between a user and machine agent in information seeking settings [3]. DoQA [26] contains domain-specific Q&A dialogues in three domains including cooking, where users can ask for recommendations/instructions regarding a specific task, although the task does not involve providing steps for completing it. Other document-grounded tasks have been proposed such as MultiWOZ [22], Taskmaster-1 [21] and ABCD [23] which demonstrate how DGDS can be configured in end-to-end pipelines for task-driven dialogue in virtual applications such as online booking systems. Here, we follow a similar setup as Doc2Dial; however, in our proposed task, we allow users to ask clarification questions, the answers to which are not necessarily grounded in the document. This consideration is vital in the development of instruction-giving conversational agents as it has implications for the dialogue pipeline.

2.4 Commonsense-Enhanced Dialogue

Commonsense reasoning is a general understanding of our surroundings, situations and objects, which is essential for many AI applications [27]. Simulating these perceptual processes in task- and goal-oriented DGDS generates greater context and grounding for more human-like comprehension. An example of commonsense dialogue in a practical task-based scenario is understanding the common storage locations of objects or the safe handling and use of objects from their common attributes, i.e. a handle, knob or grip. Commonsense dialogue is highly contextual: in Question Answering in Context (QuAC) [5], dialogues are constructed from Wikipedia articles interpreted by a teacher. A student is given the title of the article and asks the teacher questions on the subject from prior knowledge, and the teacher responds to the students' questions using the information in the document. This mode of question answering (Q&A) development is more naturalistic and grounded than previous methods as the challenges of understanding the information are ingrained in the dialogue from the underlying context. Similarly, the Conversational Question Answering Challenge (CoQA) dataset [4] is formulated on a rationale, scenario and conversation topic, and the Q&A pairs are extracted from this data. This methodology is used in the Task2Dial dataset as it provides greater co-reference and pragmatic reasoning within the dialogue for enhanced comprehension as shown in Fig. 1.

In human-human IG/IF tasks, the IG may have prior knowledge of appropriate alternative methods, components and tools that can be used in a task that are not mentioned in the instructions. This information is vital if the IF has missing components or requires clarification on aspects of the task that are not clearly represented in the document. Variability is problematic to capture in DGDS alone as hypothetical scenarios in documents cannot account for all the potential issues in practice [28]. Thus, the ability to ask questions that are not available in the document is crucial when conducting real-world tasks due to the changeable conditions, complexity of the task and availability of components. This is particularly important in cooking tasks (as well as other instruction-giving tasks) as the user may not have all the ingredients stated in a recipe but may have access to alternative items that can be used instead. This approach can also be used in other domains such as maintenance or construction tasks if the user does not have a specific tool but has access to a suitable alternative tool without knowing it. This inevitably introduces new challenges for dialogue systems as commonsense-related intents and actions need to be introduced in the dialogue system. Task2Dial moves away from the closed knowledge base(s) in DGDS into incorporating multiple sources of information to broaden the adaptability and application of DGDS. This is achieved by developing additional resources that list alternative ingredients to those mentioned in the metadata from the original recipes, as well as instructions on how to use cookery tools. Appropriate alternative ingredients were collected and verified using certified online cooking resources that provide food alternatives.

2.5 Dialogue Management

Dialogue managers are used to structure data and control the flow of a conversation and the way in which information is delivered to the user [29]. There are numerous DM tools for DGDS; however, it is important to consider the structure of the dataset and the complexity of the task [30]. Due to the complexity of our cooking scenario, the DM must be able to read multiple documents, intents, state tracking, paths, entities, rules and actions to generate responses logically and coherently [14]. The ability to deploy a DM on different platforms, channels and servers is also an important consideration for accessibility, usability, data protection and security [31]. Open-source DM tools such as RASA X[2] are particularly useful for task-based dialogue as the natural language understanding and core dialogue manager libraries are highly configurable for different tasks [32]. This is an important consideration for handling structured and unstructured data; flexibility in dialogue management, i.e. customisation of features; configuring classifiers; interpreter pipelines for training; conversation history; and managing interaction. This cannot be achieved

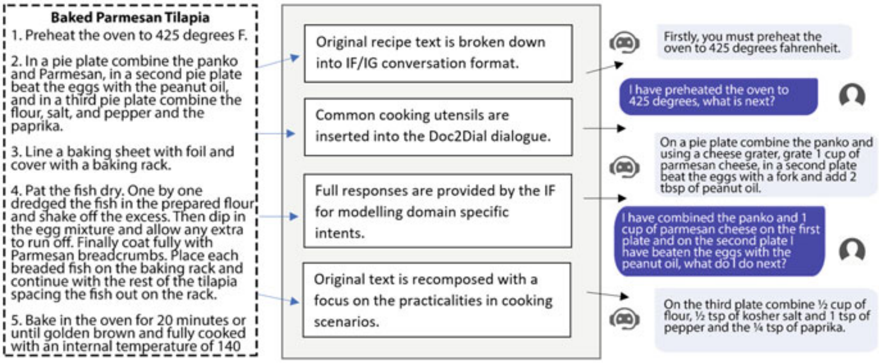
[2] rasa.com/docs/rasa-x/.

with DM tools such as Amazon Lex[3] and Google Dialogflow[4] due to system limitations and restricted user access [33, 34].

3 Task2Dial

The proposed task considers the recipe-following scenario with an information giver (IG) and an information follower (IF), where the IG has access to the recipe and gives instructions to the IF. The IG might choose to omit irrelevant information, simplify the content of a recipe or provide it as is. The IF will either follow the task or ask for further information. The IG might have to rely on information outside the given document (i.e. commonsense) to enhance understanding and success of the task. In addition, the IG decides on how to present the recipe steps, i.e. split them into sub-steps or merge them together, often diverting from the original number of recipe steps. The task is regarded successful when the IG has successfully followed/understood the recipe. Hence, other dialogue-focused metrics, such as the number of turns, are not appropriate here. Formally, *Task2Dial* can be defined as follows: given a recipe R_i from $R = R_1, R_2, R_3, \ldots, R_n$, an ontology or ontologies $O_i = O_1 1, O_2, \ldots, O_n$ of cooking-related concepts, a history of the conversation h, predict the response r of the IG.

The `Task2Dial` dataset includes (1) a set of recipe documents and (2) conversations between an IG and an IF, which are grounded in the associated recipe documents. Figure 2 presents sample utterances from a dialogue along with the associated recipe. It demonstrates some important features of the dataset, such as mentioning entities not present in the recipe document, re-composition of the



Baked Parmesan Tilapia

1. Preheat the oven to 425 degrees F.
2. In a pie plate combine the panko and Parmesan, in a second pie plate beat the eggs with the peanut oil, and in a third pie plate combine the flour, salt, and pepper and the paprika.
3. Line a baking sheet with foil and cover with a baking rack.
4. Pat the fish dry. One by one dredged the fish in the prepared flour and shake off the excess. Then dip in the egg mixture and allow any extra to run off. Finally coat fully with Parmesan breadcrumbs. Place each breaded fish on the baking rack and continue with the rest of the tilapia spacing the fish out on the rack.
5. Bake in the oven for 20 minutes or until golden brown and fully cooked with an internal temperature of 140

Original recipe text is broken down into IF/IG conversation format.

Common cooking utensils are inserted into the Doc2Dial dialogue.

Full responses are provided by the IF for modelling domain specific intents.

Original text is recomposed with a focus on the practicalities in cooking scenarios.

Firstly, you must preheat the oven to 425 degrees fahrenheit.

I have preheated the oven to 425 degrees, what is next?

On a pie plate combine the panko and using a cheese grater, grate 1 cup of parmesan cheese, in a second plate beat the eggs with a fork and add 2 tbsp of peanut oil.

I have combined the panko and 1 cup of parmesan cheese on the first plate and on the second plate I have beaten the eggs with the peanut oil, what do I do next?

On the third plate combine ½ cup of flour, ½ tsp of kosher salt and 1 tsp of pepper and the ¼ tsp of paprika.

Fig. 2 Original recipe text converted to Task2Dial dialogue

[3] https://aws.amazon.com/lex/.

[4] https://cloud.google.com/dialogflow.

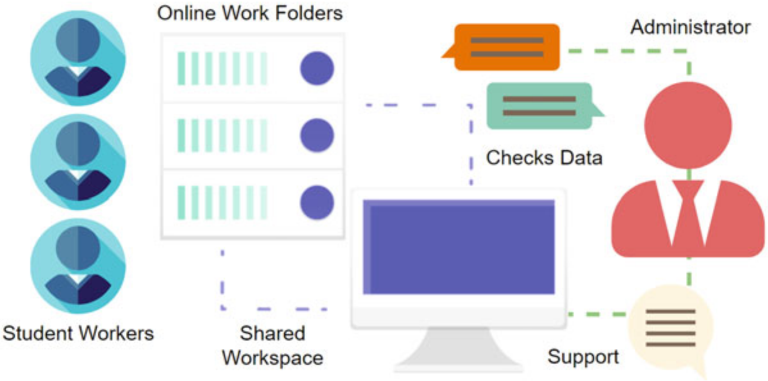


Fig. 3 Overview of the Task2Dial dataset collection

original text to focus on the important steps and the breakdown of the recipe into manageable and appropriate steps. Following recent efforts in the field to standardise NLG research [35], we have made the dataset freely available.[5]

3.1 Data Collection Methodology

The overall data collection methodology is shown in Fig. 3 and is described in detail below.

Pilot Data Collection Prior to data collection, we performed three pilot studies. In the first, two participants assumed the roles of IG and IF, respectively, where the IG had access to a recipe and provided recipe instructions to the IF (who did not have access to the recipe) over the phone, recording the session and then transcribing it. Next, we repeated the process with text-based dialogue through an online platform following a similar setup; however, the interaction was solely chat-based. The final study used *self-dialogue* [21], with one member of the team who wrote the entire dialogues assuming both the IF and IG roles. We found that self-dialogue results were proximal to the results of two-person studies. However, time and cost were higher for producing two-person dialogues, with additional time needed for transcribing and correction; thus, we opted to use self-dialogue.

Creation of a Recipe Dataset Three open-source and creative commons licensed cookery websites[6] were identified for data extraction, which permit any use or non-commercial use of data for research purposes [36, 37]. As content submission to the

[5] www.huggingface.co/datasets/cstrathe435/Task2Dial.

[6] (a) www.makebetterfood.com, (b) www.cookeatshare.com and (c) www.bbcgoodfood.com.

cooking websites was unrestricted, data appropriateness was ratified by the ratings and reviews given to each recipe by the public, and highly rated recipes with positive feedback were given preference over recipes with low scores and poor reviews [38]. From this, a list of 353 recipes was compiled and divided amongst the annotators for the data collection. As mentioned earlier, annotators were asked to take on the roles of both IF and IG, rather than a multi-turn WoZ approach, to allow flexibility in the utterances. This approach allowed the annotators additional time to formulate detailed and concise responses.

Participants Research assistants (RAs) from the School of Computing were employed on temporary contracts to construct and format the dataset. After an initial meeting to discuss the job role and determine suitability, the RAs were asked to complete a paid trial, and this was evaluated, and further advice was given on how to write dialogues and format the data to ensure high quality. After the successful completion of the trial, the RAs were permitted to continue with the remainder of the data collection. To ensure high quality of the dataset, samples of the dialogues were often reviewed, and further feedback was provided.

Instructions to Annotators Each annotator was provided with a detailed list of instructions, an example dialogue and an IF/IG template (see Appendix A). The annotators were asked to read both the example dialogue and the original recipe to understand the text, context, composition, translation and annotation. The instructions included information handling and storage of data, text formatting, metadata and examples of high-quality and poor dialogues. An administrator was on hand throughout the data collection to support and guide the annotators. This approach reduced the amount of low-quality dialogues associated with large crowdsourcing platforms that are often discarded post evaluation, as demonstrated in the data collection of the Doc2Dial dataset [3].

Time Scale The data collection was scheduled over 4 weeks. This was to permit additional time for the annotators to conduct work and study outside of the project. Unlike crowdsourcing methods, the annotators were given the option to work on the project flexibly in their spare time and not commit to a specific work pattern or time schedule.

Ethics An ethics request was submitted for review by the board of ethics at our university. No personal or other data that may be used to identify an individual was collected in this study.

Task2Dial Long-Form Description Unlike previous task- and goal-oriented DGDS, the Task2Dial corpus is unique as it is configured for practical IF/IG scenarios as demonstrated in Fig. 2. Following [39], we provide a long-form description of the Task2Dial cooking dataset here.

Curation Rationale Text selection was dependent on the quality of information provided in the existing recipes. Too little information and the transcription and interpretation of the text became diffused with missing or incorrect knowledge. Conversely, providing too much information in the text resulted in a lack of creativity and commonsense reasoning by the data curators. Thus, the goal of the curation was to identify text that contained all the relevant information to complete the cooking task (tools, ingredients, weights, timings, servings) but not in such detail that it subtracted from the creativity, commonsense and imagination of the annotators.

Language Variety The recipes selected for this dataset were either written in English or translated into English prior to data collection for ease of the annotators, language understanding and future training for language models. This made the dataset accessible to all contributors involved in the curation, support and administration framework.

Speaker Demographics The recipes are composed by people of different race/ethnicity, nationalities, socioeconomic status, abilities, age, gender and language with significant variation in pronunciations, structure, language and grammar. This provided the annotators with unique linguistic content for each recipe to interpret the data and configure the text into an IF/IG format. To help preserve sociolinguistic patterns in speech, the data curators retained the underlying language when paraphrasing, to intercede social and regional dialects with their own interpretation of the data to enhance lexical richness [40].

Annotator(s) Demographics Undergraduate RAs were recruited through email. The participants were paid an hourly rate based on a university pay scale which is above the living wage and corresponds to the real living wage, following ethical guidelines for responsible innovation [41]. The annotation team was composed of two male and one female data curators, under the age of 25 years of mixed ethnicity with experience in AI and computing. This minimised the gender bias that is frequently observed in crowdsourcing platforms [42].

Speech Situation The annotators were given equal workloads, although workloads were adjusted accordingly over time per annotator availability to maximise data collection. The linguistic modality of the dialogue is semi-structured, synchronous interactions as existing recipes were used to paraphrase the instructions for the IG. Following this, the IF responses were created spontaneously following the logical path of the recipe in the context of the task. The intended audience for the Task2Dial dataset is broad, catering for people of different ages and abilities. Thus, the dataset is written in plain English with no jargon or unnecessary commentary to maximise accessibility.

Text Characteristics The structural characteristics of the Task2Dial dataset are influenced by real-world cooking scenarios that provide genre, texture and structure to the dialogues. This provides two important classifications, utterances and intents, that are universal for all task-based datasets and domain-specific text that is only relevant for certain tasks. This data is used when training language models as non-domain-specific sample utterances such as ‘I have completed this step’ can be used to speed up the development of future task-based DGDS.

Recording Quality As mentioned previously, the dialogues in Task2Dial are text-based.

4 Dataset Analysis

This section presents an overall statistics of the Task2Dial dataset. We compare our dataset to the Doc2Dial dataset, although the latter focuses on a different domain. Employing research assistants to collect and annotate data rather than using crowdsourcing platforms meant that no dialogues were discounted from the dataset. However, a pre-evaluation check was performed on the dataset before statistical analysis to reduce spelling and grammatical errors that may affect the results of the lexical analysis.

Size Table 1 summarises the main descriptive statistics of Task2Dial and Doc2Dial. The dialogues in Task2Dial contain a significantly higher number of turns than Doc2Dial dialogues (18.15 as opposed to 12.94). In addition, Task2Dial utterances are significantly longer than in Doc2Dial, containing on average more than seven tokens.

Lexical Richness and Variation We further report on the lexical richness and variation [43], following [44] and [45]. We compute both type-token ratio (TTR), i.e. the ratio of the number of word types to the number of words in a text, and the mean segmental TTR (MSTTR), which is computed by dividing the corpus into successive segments of a given length and then calculating the average TTR of all segments to account for the fact the compared datasets are not of equal size.[7] All results are shown in Table 1. We further investigate the distribution of the top 25 most frequent bigrams and trigrams in our dataset as seen in Fig. 4. The majority of both trigrams (75%) and bigrams (59%) is only used once in the dataset, which creates a challenge to efficiently train on this data. For comparison, in Doc2Dial, 54% of bigrams and 70% of trigrams are used only once. Infrequent words and phrases pose a challenge for the development of data-driven dialogue systems as handling out-of-vocabulary words is a bottleneck.

Table 1 Size and lexical richness of the dataset

Dataset	#docs	#Turns	#Tkns/turn	TTR	MSTTR
TASK2DIAL	353	**18.15**	**19.79**	**0.025**	0.84
DOC2DIAL	487	12.94	12	0.011	0.86

[7] TTR and MSTTR have been computed using https://github.com/LSYS/LexicalRichness.

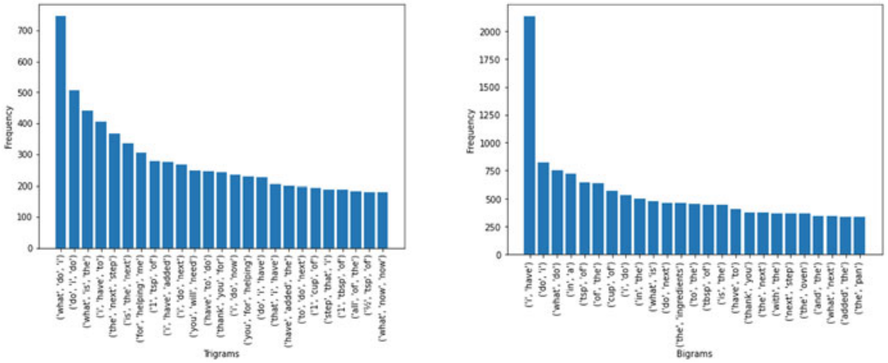


Fig. 4 Frequencies of trigrams and bigrams in the Task2Dial dataset

5 The ChefBot Conversational Agent

ChefBot was created using the RASA X dialogue manager[8] to control the dialogue flow and access external databases for swapping ingredients, object explanations, intents, utterances and entities modelled from the dialogues in the Task2Dial dataset, as shown in Fig. 5.

ChefBot System Architecture The system architecture for ChefBot is depicted in Fig. 6, and the technical details of the system are described in this section. The data folder contains the files that the ChefBot is trained on, and these include the IF dialogues and recipe sequences from the Task2Dial dataset. The rules file contains the directives for intents, paths and state tracking. The actions folder holds the entities files which are the external datasets and rules for alternative ingredients and object explanations. When a model is trained, it is stored in the models folder. Similarly, if a path is changed or corrected during a session, i.e. using RASA interactive, it is stored in the test folder. The domain file contains the IG dialogues from the Task2Dial dataset configured into utterances. This file also contains the classifications for the intents, entities and actions. The credential's file contains the parameters for deploying the system on channels and servers. Similarly, the endpoints file is the data for the custom actions server for entity extraction. The config file is the interpreter pipeline for the NLU model that includes the classifiers and policies for training the ChefBot. When a trained model is loaded into a terminal (such as Anaconda[9] or similar), it can be deployed using the RASA shell or RASA X commands to load the RASA user interface (UI) on a channel or server, allowing the user to interact with ChefBot.

[8] rasa.com/blog/dialogue-policies-rasa-2/.

[9] www.anaconda.com/.

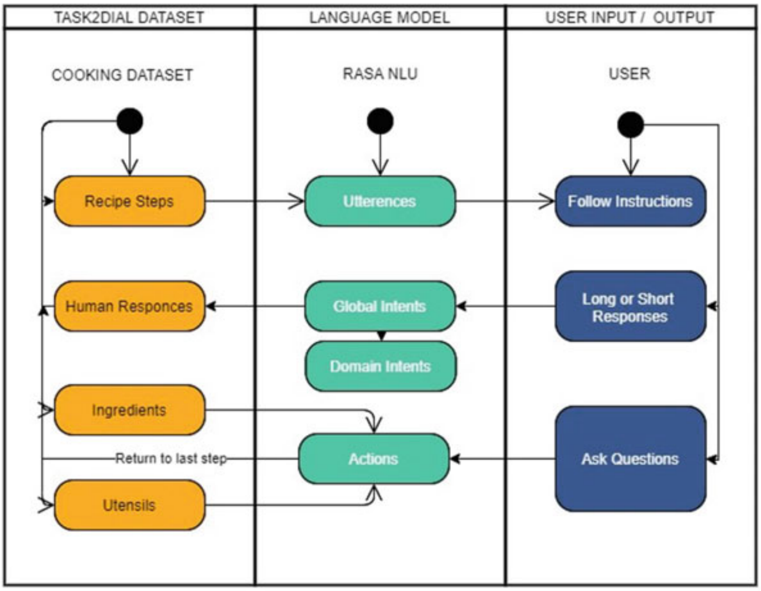


Fig. 5 The pipeline from Task2Dial to ChefBot and the user

Data Entry and Formatting Data entry was conducted over a 6-week period by project members. Due to the restructuring of data, manual entry was the most effective way to ensure data from the Task2Dial dataset was formatted and entered correctly in ChefBot. All 353 recipe documents, alternative food and object databases from the Task2Dial dataset were successfully uploaded into ChefBot within the designated 6-week period.

Modelling Intents and Utterances ChefBot uses non-domain-specific 'user' responses from the Task2Dial dataset to model *global intents* in the dialogue manager, such as 'I have done this', 'OK what's next' and 'What is the next step'. These global intents can be used in other task-based dialogue scenarios, such as cleaning and maintenance tasks. *Domain-specific intents* are modelled from the user responses which contain information that is only relevant to the cooking domain. For example, 'I have put the cake in the oven' or 'I have mixed the ingredients in a bowl'. This approach is important for enhancing natural communication between the IG and IF as it allows the IF to give both short and full responses to the IG, proximal to a genuine human conversation. Within the domain file, the instructions from the IG were turned into utterances and numerically labelled depending on the position of the instruction within the sequence of a recipe, i.e. r1a, r1b, r1c, etc., as shown in the example below. This approach creates a sequential order for each recipe which can be tracked in the DM. This data is used both for state tracking and creating a dialogue pathway for each recipe.

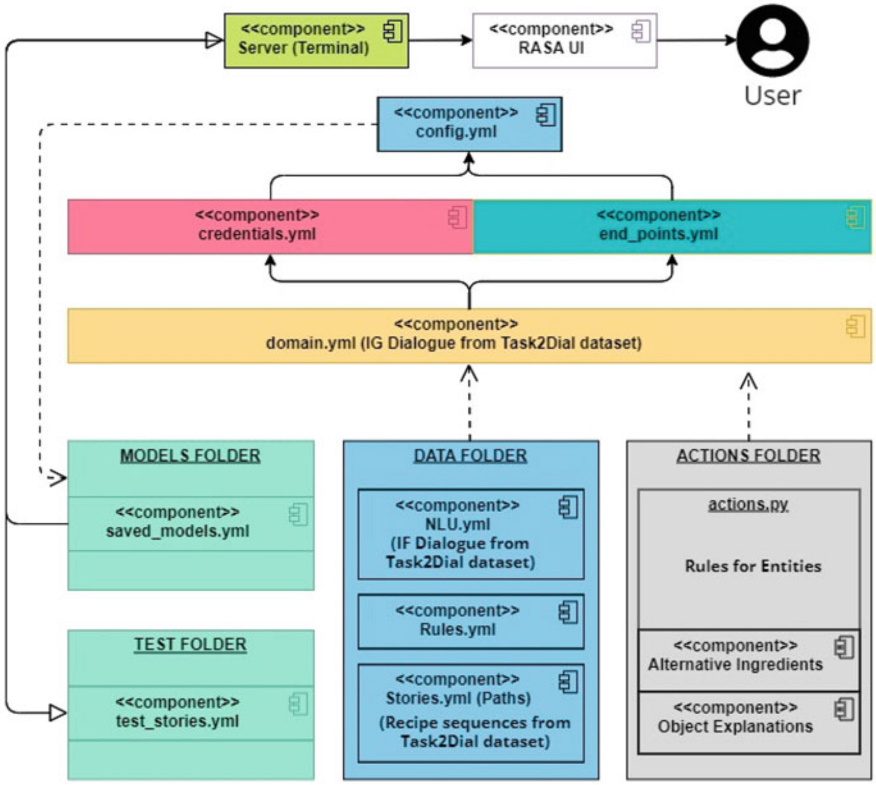

<<component>> Server (Terminal)
<<component>> RASA UI
User
<<component>> config.yml
<<component>> credentials.yml
<<component>> end_points.yml
<<component>> domain.yml (IG Dialogue from Task2Dial dataset)
MODELS FOLDER
<<component>> saved_models.yml
TEST FOLDER
<<component>> test_stories.yml
DATA FOLDER
<<component>> NLU.yml (IF Dialogue from Task2Dial dataset)
<<component>> Rules.yml
<<component>> Stories.yml (Paths) (Recipe sequences from Task2Dial dataset)
ACTIONS FOLDER
actions.py
Rules for Entities
<<component>> Alternative Ingredients
<<component>> Object Explanations

Fig. 6 ChefBot system architecture

Example of Modelling User Utterances in ChefBot
utter_r1a:—text: English Muffins takes 180 minutes to cook, serves 16 and contains 7 ingredients, is this ok?
utter_r1b:—text: To cook English Muffins you will need, ‘milk’, ‘butter’, ‘salt’, ‘sugar’, ‘egg’, ‘bread flour’, ‘instant yeast’, do you have these ingredients?
utter_r1c:—text: To start with combine 1 and three quarter cups of lukewarm milk, 3 tablespoons of soft butter, one and a half teaspoons of salt, 2 tablespoons of sugar, one egg, 5 cups of bread flour, and 2 teaspoons of yeast in a large mixing bowl of an electric stand mixer.

Modelling Dialogue Paths and Conversation History Within each pathway are the global and domain-specific intents for each recipe that are activated using the ‘or’ variable in multi-intent, multi-turn policy, as outlined in the literature review. This information is important for training the DM to determine the next logical step in a sequence from the history of the conversation and the path. Custom actions for alternative ingredients activate if the user answers ‘no’ to ‘do you have all the ingredients?’. This initiates the search_rec function and lists the alternative ingredients for each recipe. The paths are modelled using the IG and IF sequences from the Task2Dial dataset, as demonstrated below.

Example of Recipe Path in ChefBot

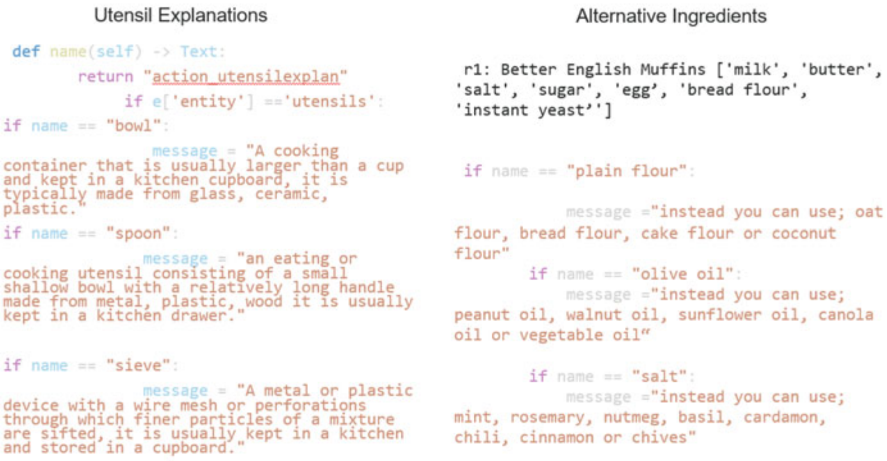
- story:
  strawberrypienopath //Name of path
  steps:
- intent: strawberrypie //Name of recipe from Task2Dial dataset
- action: utter_r2a //First line of the recipe sequence
- or:
- intent: globint //Global intents
- intent: r2 //Domain/recipe specific intents
- action: utter_r2b //Second line of the recipe sequence
- intent: nomode
- action: utter_ingredients_strawberrypie //Identify missing ingredient's
- intent: search_rec_r2a // Perform a search for alternative ingredients

Rule-Based Tracking and Entity Extraction The frequently asked questions (FAQ) rule in RASA allows the user to ask questions that may not be represented within a given form or path. The DM will then answer the question using specific 'FAQ' labelled intents and then return to the next or previous step in a sequence, as shown below. The FAQ labelling facility can also be used to create a list of intents for context-aware entity extraction, i.e. 'how do I use a' with entities [cheese knife] (utensil) within a given FAQ function. This method is less formulaic than using RASA forms, which requires specified slots to be filled at each stage of a sequence or sub-sequence, which is important in ChefBot as we aim to capture the natural flow of conversation between the IG and IF from the Task2Dial dataset, to enhance user understanding and accessibility.

Rule-Based Tracking Example

- rule: respond to IF questions
  steps:
- intent: utter_faq_questions
- action: search_utensils

External Databases for Alternative Ingredients and Object Descriptions In ChefBot, additional commonsense knowledge is modelled in two external databases. The first is the ability to swap ingredients for appropriate alternatives. It is important that the alternative ingredients do not alter the procedural context of the recipe. For example, swapping olive oil for sunflower oil will not change how a recipe is prepared or cooked. Conversely, changing chicken breast for beef fillet would require a significant change in the recipe instructions. This would have an impact on the cooking situation, including times, food preparation, servings, steps and utensils, and may require additional ingredients for cooking or preparation. Therefore, to avoid unnecessary complications, all alternative ingredients must not significantly affect the sequence and instructions within a given recipe.



Utensil Explanations

```
 def name(self) -> Text:
        return "action_utensilexplan"
              if e['entity'] =='utensils':
if name == "bowl":
                  message = "A cooking
container that is usually larger than a cup
and kept in a kitchen cupboard, it is
typically made from glass, ceramic,
plastic."
if name == "spoon":
                 message = "an eating or
cooking utensil consisting of a small
shallow bowl with a relatively long handle
made from metal, plastic, wood it is usually
kept in a kitchen drawer."

if name == "sieve":
                 message = "A metal or plastic
device with a wire mesh or perforations
through which finer particles of a mixture
are sifted, it is usually kept in a kitchen
and stored in a cupboard."
```

Alternative Ingredients

```
 r1: Better English Muffins ['milk', 'butter',
'salt', 'sugar', 'egg’, 'bread flour',
'instant yeast’']


 if name == "plain flour":

           message ="instead you can use; oat
flour, bread flour, cake flour or coconut
flour"
       if name == "olive oil":
           message ="instead you can use;
peanut oil, walnut oil, sunflower oil, canola
oil or vegetable oil“

       if name == "salt":
           message ="instead you can use;
mint, rosemary, nutmeg, basil, cardamon,
chili, cinnamon or chives"
```

Fig. 7 Examples of how the additional datasets were handled as custom actions in ChefBot. Left: utensil explanations. Right: alternative ingredients

Metadata containing information on the ingredients and utensils used in each recipe from the Task2Dial dataset was extracted. The first dataset was created using the list of ingredients from each recipe. A Google search using cooking and food health websites was performed to find appropriate alternatives for each ingredient. Similarly, a list of cooking utensils and kitchen devices was constructed using the same approach. However, the second dataset also contains object descriptions, object comparisons, alternative names for objects, appropriate handling methods and common storage locations. This data is important as it may not be grounded in the original documents, but vital for enhancing user understanding. This approach allows the IG to simplify the content or provide additional information depending on the needs of the IF. The two datasets are transformed into custom actions in the dialogue manager as shown in Fig. 7.

Using these databases as custom actions allows the user to trigger an action at any stage of the task from keyword recognition. For instance, in Fig. 6, the keyword or entity extraction is the names of the ingredients and objects. In the intent list, these entities are given context, for example, ‘how do I use a (fish slice) [object-name]’ or ‘what does a (lemon zester) [object-name] look like’. This is important as the user’s response may consist of more than one named entity. For instance, ‘I do not know where my (fish slice) is kept or what a (lemon zester) looks like’. Here context awareness is important for relaying information back to the user in a meaningful way. This was achieved by using the multi-intent function in rasa to handle more than one intent per turn.

ChefBot Demo and Repository Training ChefBot takes approximately 2–3 hours, so a trained model is supplied in a GitHub repository within the system files for ease of demonstration.[10] A description of the libraries and system requirements needed to run ChefBot are located in the 'requirements.txt' file. The provided video demonstrates how the ChefBot generates dialogue, swaps ingredients, uses global and domain-specific intents and explains the utility of objects and state tracking, using a random recipe selected from the Task2Dial dataset.[11]

6 Conclusions and Future Work

This paper demonstrates how commonsense-enhanced document-grounded dialogue can be modelled for task-based dialogue. As a case study, we used the Task2Dial, a task-based document-grounded conversation dataset, modelled as an interaction between an IG and an IF during a cooking task. In this domain, commonsense is the ability to provide alternative ingredients and provide recommendations on object utility, both of which are not present in the cooking instruction dialogues and require additional knowledge in the form of a database or domain ontology. We then presented a novel conversational agent architecture, ChefBot, which is able to flexibly adapt to the changes in dialogue flow. With this research, we extend previous work in DGDS in order to emulate the unpredictability of human-human conversations in instruction-giving tasks that do not necessarily follow a tight schema as the sequential structure of instructional documents. Instead, other discourse and dialogue phenomena might take place such as clarification questions and explanations. We further considered the aforementioned challenges of modelling dialogue for instruction-giving tasks with a focus on state tracking, task planning and commonsense reasoning and proposed a new task, model and associated dataset. With this, we demonstrate a more robust approach for DGDS called CDGDS to more effectively handle real-world task-based scenarios and open the door to tasks outside the cooking domain, such as general maintenance and furniture assembly.

6.1 Future Work and Open Questions

Our proposed task aims to motivate research for modern dialogue systems that address the following challenges. Firstly, modern dialogue systems should be flexible and allow for 'off-script' scenarios in order to emulate real-world phenomena, such as the ones present in human-human communication. This will require new ways of encoding user intents and new approaches to dialogue management in general. Secondly, as dialogue systems find different domain applications, the

[10] github.com/carlstrath/ChefBot.

[11] https://youtu.be/XoTXraGs5rA.

complexity of the dialogues might increase as well as the reliance of domain knowledge that can be encoded in structured or unstructured ways, such as documents, databases, etc. Many applications might require access to different domain knowledge sources in a course of a dialogue, and as such, context selection might prove beneficial in choosing 'what to say' [46]. Finally, as we design more complex dialogue systems, commonsense will play an essential part, with models required to perform reasoning with background commonsense knowledge, and generalise to tackle unseen concepts, similar to [9]. In the future, we aim to benchmark and evaluate a dialogue system based on the Task2Dial dataset and the ChefBot [14] and extend this approach to a human-robot interaction (HRI) scenario. Other interesting directions can include the exploration of pre-trained models as part of a conversational agent architecture to eliminate the need to encode knowledge or design domain ontologies [47].

Acknowledgments The research is supported under the EPSRC projects CiViL (EP/T014598/1) and NLG for low-resource domains (EP/T024917/1).

References

1. Chen, H., Liu, X., Yin, D., Tang, J.: A survey on dialogue systems: Recent advances and new frontiers. SIGKDD Explor. Newsl. **19**(2), 25–35 (2017). https://doi.org/10.1145/3166054.3166058
2. Shah, P., Hakkani-Tür, D., Liu, B., Tür, G.: Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers), pp. 41–51. Association for Computational Linguistics, New Orleans - Louisiana (2018). https://doi.org/10.18653/v1/N18-3006. https://www.aclweb.org/anthology/N18-3006
3. Feng, S., Wan, H., Gunasekara, C., Patel, S., Joshi, S., Lastras, L.: doc2dial: A goal-oriented document-grounded dialogue dataset. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8118–8128. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.emnlp-main.652. https://www.aclweb.org/anthology/2020.emnlp-main.652
4. Reddy, S., Chen, D., Manning, C.D.: CoQA: A conversational question answering challenge. Trans. Assoc. Comput. Linguist. **7**, 249–266 (2019). https://doi.org/10.1162/tacl_a_00266. https://aclanthology.org/Q19-1016
5. Choi, E., He, H., Iyyer, M., Yatskar, M., tau Yih, W., Choi, Y., Liang, P., Zettlemoyer, L.: Quac: Question answering in context (2018)
6. Strathearn, C., Gkatzia, D.: The Task2Dial dataset: A novel dataset for commonsense-enhanced task-based dialogue grounded in documents. In: Proceedings of The Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021), pp. 242–251. Association for Computational Linguistics, Trento, Italy (2021). https://aclanthology.org/2021.icnlsp-1.28
7. Hu, Z., Dick, M., Chang, C.N., Bowden, K., Neff, M., Fox Tree, J., Walker, M.: A corpus of gesture-annotated dialogues for monologue-to-dialogue generation from personal narratives. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pp. 3447–3454. European Language Resources Association (ELRA), Portorož, Slovenia (2016). https://aclanthology.org/L16-1550

8. Stoyanchev, S., Piwek, P.: Constructing the CODA corpus: A parallel corpus of monologues and expository dialogues. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), Valletta, Malta (2010). http://www.lrec-conf.org/proceedings/lrec2010/pdf/127_Paper.pdf
9. Lin, B.Y., Zhou, W., Shen, M., Zhou, P., Bhagavatula, C., Choi, Y., Ren, X.: CommonGen: A constrained text generation challenge for generative commonsense reasoning. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1823–1840. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.findings-emnlp.165. https://aclanthology.org/2020.findings-emnlp.165
10. Clinciu, M.A., Gkatzia, D., Mahamood, S.: It's commonsense, isn't it? demystifying human evaluations in commonsense-enhanced NLG systems. In: Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval), pp. 1–12. Association for Computational Linguistics, Online (2021). https://aclanthology.org/2021.humeval-1.1
11. Panagiaris, N., Hart, E., Gkatzia, D.: Generating unambiguous and diverse referring expressions. Comput. Speech Lang. **68**, 101184 (2021). https://doi.org/10.1016/j.csl.2020.101184. https://www.sciencedirect.com/science/article/pii/S0885230820301170
12. Gkatzia, D., Belvedere, F.: "what's this?" comparing active learning strategies for concept acquisition in hri. In: Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, HRI '21 Companion, p. 205–209. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3434074.3447160
13. Gargett, A., Garoufi, K., Koller, A., Striegnitz, K.: The GIVE-2 corpus of giving instructions in virtual environments. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), Valletta, Malta (2010). http://www.lrec-conf.org/proceedings/lrec2010/pdf/532_Paper.pdf
14. Strathearn, C., Gkatzia, D.: Chefbot: A novel framework for the generation of commonsense-enhanced responses for task-based dialogue systems. In: Proceedings of the 14th International Conference on Natural Language Generation, pp. 46–47. Association for Computational Linguistics, Aberdeen, Scotland, UK (2021). https://aclanthology.org/2021.inlg-1.5
15. Hosseini-Asl, E., McCann, B., Wu, C.S., Yavuz, S., Socher, R.: A simple language model for task-oriented dialogue. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 20179–20191. Curran Associates, Inc. (2020). https://proceedings.neurips.cc/paper/2020/file/e946209592563be0f01c844ab2170f0c-Paper.pdf
16. Ham, D., Lee, J.G., Jang, Y., Kim, K.E.: End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 583–592. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.54. https://aclanthology.org/2020.acl-main.54
17. Zhang, Z., Takanobu, R., Huang, M., Zhu, X.: Recent advances and challenges in task-oriented dialog system. CoRR **abs/2003.07490** (2020). https://arxiv.org/abs/2003.07490
18. Ilievski, V., Musat, C., Hossmann, A., Baeriswyl, M.: Goal-oriented chatbot dialog management bootstrapping with transfer learning. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, p. 4115–4121. AAAI Press (2018)
19. Zamanirad, S., Benatallah, B., Rodriguez, C., Yaghoubzadehfard, M., Bouguelia, S., Brabra, H.: State machine based human-bot conversation model and services. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) Advanced Information Systems Engineering, pp. 199–214. Springer International Publishing, Cham (2020)
20. Shum, H.Y., He, X., Li, D.: From eliza to xiaoice: Challenges and opportunities with social chatbots (2018)
21. Byrne, B., Krishnamoorthi, K., Sankar, C., Neelakantan, A., Duckworth, D., Yavuz, S., Goodrich, B., Dubey, A., Cedilnik, A., Kim, K.: Taskmaster-1: Toward a realistic and diverse dialog dataset. CoRR **abs/1909.05358** (2019). http://arxiv.org/abs/1909.05358

22. Budzianowski, P., Wen, T.H., Tseng, B.H., Casanueva, I., Ultes, S., Ramadan, O., Gašić, M.: MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 5016–5026. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/D18-1547. https://www.aclweb.org/anthology/D18-1547
23. Chen, D., Chen, H., Yang, Y., Lin, A., Yu, Z.: Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 3002–3017. Association for Computational Linguistics, Online (2021). https://www.aclweb.org/anthology/2021.naacl-main.239
24. Zhou, K., Prabhumoye, S., Black, A.W.: A dataset for document grounded conversations. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 708–713. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/D18-1076. https://aclanthology.org/D18-1076
25. Ma, L., Zhang, W., Li, M., Liu, T.: A survey of document grounded dialogue systems (DGDS). CoRR **abs/2004.13818** (2020). https://arxiv.org/abs/2004.13818
26. Campos, J.A., Otegi, A., Soroa, A., Deriu, J., Cieliebak, M., Agirre, E.: Doqa—accessing domain-specific faqs via conversational qa (2020)
27. Ilievski, F., Oltramari, A., Ma, K., Zhang, B., McGuinness, D.L., Szekely, P.: Dimensions of commonsense knowledge (2021)
28. Li, Z., Niu, C., Meng, F., Feng, Y., Li, Q., Zhou, J.: Incremental transformer with deliberation decoder for document grounded conversations. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 12–21. Association for Computational Linguistics, Florence, Italy (2019). https://doi.org/10.18653/v1/P19-1002. https://aclanthology.org/P19-1002
29. Galitsky, B., Ilvovsky, D.: Chatbot with a discourse structure-driven dialogue management. In: Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pp. 87–90 (2017)
30. Ma, L., Zhang, W.N., Li, M., Liu, T.: A survey of document grounded dialogue systems (dgds) (2020)
31. Hasal, M., Nowaková, J., Ahmed Saghair, K., Abdulla, H., Snášel, V., Ogiela, L.: Chatbots: Security, privacy, data protection, and social aspects. Concurr. Comput. Pract. Exp. **33**(19), e6426 (2021). https://doi.org/10.1002/cpe.6426. https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6426
32. Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A.: Rasa: Open source language understanding and dialogue management (2017)
33. Williams, S.: Hands-On Chatbot Development with Alexa Skills and Amazon Lex: Create Custom Conversational and Voice Interfaces for Your Amazon Echo Devices and Web Platforms. Packt Publishing Ltd. (2018)
34. Sabharwal, N., Agrawal, A.: Cognitive Virtual Assistants Using Google Dialogflow: Develop Complex Cognitive Bots Using the Google Dialogflow Platform. Apress (2020)
35. Gehrmann, S., Adewumi, T.P., Aggarwal, K., Ammanamanchi, P.S., Anuoluwapo, A., Bosselut, A., Chandu, K.R., Clinciu, M., Das, D., Dhole, K.D., Du, W., Durmus, E., Dusek, O., Emezue, C., Gangal, V., Garbacea, C., Hashimoto, T., McMillan-Major, A., Mille, S., van Miltenburg, E., Nadeem, M., Narayan, S., Nikolaev, V., Niyongabo, R.A.: The GEM benchmark: Natural language generation, its evaluation and metrics. CoRR **abs/2102.01672** (2021). https://arxiv.org/abs/2102.01672
36. Bień, M., Gilski, M., Maciejewska, M., Taisner, W., Wisniewski, D., Lawrynowicz, A.: RecipeNLG: A cooking recipes dataset for semi-structured text generation. In: Proceedings of the 13th International Conference on Natural Language Generation, pp. 22–28. Association for Computational Linguistics, Dublin, Ireland (2020). https://www.aclweb.org/anthology/2020.inlg-1.4

37. Marin, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I., Torralba, A.: Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. IEEE Trans. Pattern Anal. Mach. Intell. (2019). arXiv:1810.06553
38. Wang, Y., Kim, J.: Interconnectedness between online review valence, brand, and restaurant performance. J. Hosp. Tour. Manag. **48**, 138–145 (2021). https://doi.org/10.1016/j.jhtm.2021.05.016. https://www.sciencedirect.com/science/article/pii/S1447677021000851
39. Bender, E.M., Friedman, B.: Data statements for natural language processing: Toward mitigating system bias and enabling better science. Trans. Assoc. Comput. Linguist. **6**, 587–604 (2018). https://doi.org/10.1162/tacl_a_00041
40. Zampieri, M., Nakov, P., Scherrer, Y.: Natural language processing for similar languages, varieties, and dialects: A survey. Nat. Lang. Eng. **26**(6), 595–612 (2020). https://doi.org/10.1017/S1351324920000492
41. Silberman, M.S., Tomlinson, B., LaPlante, R., Ross, J., Irani, L., Zaldivar, A.: Responsible research with crowds: Pay crowdworkers at least minimum wage. Commun. ACM **61**(3), 39–41 (2018). https://doi.org/10.1145/3180492
42. Goodman, J.K., Cryder, C., Cheema, A.: Data collection in a flat world: Strengths and weaknesses of mechanical turk samples. J. Behav. Decis. Making (2012, Forthcoming)
43. Van Gijsel, S., Speelman, D., Geeraerts, D.: A variationist, corpus linguistic analysis of lexical richness, pp. 1–16 (2005)
44. Novikova, J., Dušek, O., Rieser, V.: The E2E dataset: New challenges for end-to-end generation. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, pp. 201–206. Association for Computational Linguistics, Saarbrücken, Germany (2017). https://doi.org/10.18653/v1/W17-5525. https://aclanthology.org/W17-5525
45. Perez-Beltrachini, L., Gardent, C.: Analysing data-to-text generation benchmarks. In: Proceedings of the 10th International Conference on Natural Language Generation, pp. 238–242. Association for Computational Linguistics, Santiago de Compostela, Spain (2017). https://doi.org/10.18653/v1/W17-3537. https://aclanthology.org/W17-3537
46. Gkatzia, D.: Content selection in data-to-text systems: A survey. CoRR **abs/1610.08375** (2016). http://arxiv.org/abs/1610.08375
47. Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.: Language models as knowledge bases? In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2463–2473. Association for Computational Linguistics, Hong Kong, China (2019). https://doi.org/10.18653/v1/D19-1250. https://aclanthology.org/D19-1250

BloomQDE: Leveraging Bloom's Taxonomy for Question Difficulty Estimation

Sabine Ullrich, Amon Soares de Souza, Josua Köhler, and Michaela Geierhos

Abstract Current question answering systems often focus on providing a simple entity or short sentence as an answer. By gaining confidence in information retrieval systems, users start to ask more complex questions that require sophisticated answers, such as reasoning chains. However, no research has been carried out yet to determine how exhaustive an answer should be.

We combine Bloom's learner's levels of understanding with question difficulty classification. Therefore, we heuristically determine the threshold within Bloom's taxonomy that separates question types into simple and complex. Moreover, we extract keywords from the taxonomy within question datasets and categorize questions accordingly. Then, we train a word n-gram multi-layer perceptron (MLP) and an LSTM model with syntactic features. The results are further improved by applying a genetic algorithm for parameter tuning.

1 Introduction

Question answering (QA) has become increasingly popular in information retrieval and chatbots and on smart home devices. The returned answer thereby heavily depends on the trained system: some systems (Alexa, Siri) return simple answers such as short entities or yes/no answers. Other systems are trained on reasoning datasets and provide a chain of reasoning steps. This can be useful when explanations are needed, but the steps are provided for any type of question complexity.

Question difficulty estimation (QDE) can help here, as it is already used for measuring difficulty in question forums or student essays, for example. A popular approach for student essays is Bloom's taxonomy, in which questions are estimated using question keywords. Such keywords range from simple ones such as "name" or "list" to more difficult questions including "explain" or "justify."

S. Ullrich · A. Soares de Souza · J. Köhler · M. Geierhos (✉)
Research Institute CODE, Bundeswehr University Munich, Neubiberg, Germany
e-mail: michaela.geierhos@unibw.de

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_7

These keywords also occur in QA. Therefore, extracting them and classifying the questions accordingly can help to determine the required answer type. Furthermore, the inclusion of syntactic features helps to classify unseen sentences that do not contain keywords from the taxonomy. This can help to provide a tailored answer to the question without being limited to simple answers or multi-hop answers.

2 Related Work

As relevant as determining the difficulty of questions in QA systems and reasoning might seem, not much research has been conducted on QDE [12]. To the best of our knowledge, there has been no work done on predictive QDE using Bloom's taxonomy.

However, some research has been done on the question if Bloom's taxonomy can be usefully applied to determine question difficulty. Padó [6] experiments on a small dataset consisting of test questions and student answers showed that variation in student responses can be an indicator of question difficulty as well as that the latter can be successfully approximated by Bloom's taxonomy. Question difficulty was estimated by applying the Rasch model, which is a joint model of student ability and question difficulty. The knowledge dimension according to Bloom's taxonomy was determined for each question by the annotators. In a final step, they analyzed whether the annotation according to Bloom's taxonomy actually reflected the difficulty determined by the Rasch model.

Some work has been done on predicting question difficulty in the context of community question services [5, 10]. This context benefits from the advantage that both textual features and additional meta-information about users and their interactions can be added to an estimation, so the problem addressed is only partly related to our problem.

One approach [5] uses a competition-based Bayesian model. This model determines question difficulty by reframing the process of question answering as a competition, where the questioner, the answerer, and the question are considered to be players u. Their expected performance is given by a normal distribution $\mathcal{N}(\mu, \sigma)$, where μ indicates the skill level. The parameters of the distribution are iteratively fitted by applying the model to a dataset.

A model by Huang et al. [3] for estimating the question difficulty in reading tasks uses sentence representations for questions and neural networks to process text and estimate difficulty. The network architecture is a convolutional neural network that uses an attention mechanism to measure the contribution of the reading context to question difficulty. For training and validation, they used a dataset of reading tasks for standard tests, where test logs gave an indicator of task difficulty for each reading task. They also introduced a test-dependent pairwise loss function in order to address the incomparability of question difficulty across tests.

Some research has been conducted to predict question difficulty in specific domains. DAN [7] uses an attention-based neural network with document expansion to determine question difficulty in multiple-choice medical exams. DAN retrieves relevant medical documents to enrich questions with additional textual information. To improve effectiveness, DAN breaks down the difficulty of questions into two distinct parts so that two attention layers can attend to different aspects of the question. Finally, DAN processes the output of the two attention layers to estimate the final difficulty score.

3 Approach

We combine Bloom's learner's levels of understanding with question difficulty classification. We argue that questions that are difficult for learners require similarly difficult answers in QA systems. Bloom separates the levels of understanding into the knowledge dimension (factual, conceptual, procedural, and meta-cognitive knowledge) and the cognitive process dimension (remember, understand, apply, analyze, and evaluate). The complexity of the questions increases from top to bottom (A–D) and from left to right (1–5). Formally, complexity can be defined as follows:

Theorem 1 *Let $Q = \{q_1, \ldots, q_n\}$ be a set of n questions, and find all lexical, structural, and semantic features F_{lex}, F_{struct}, $F_{sem} \in F$ that map Q to a binary set of complexity classes $C = \{0, 1\}$. If the mapping $F(q_m) \rightarrow C$ yields* 0*, a simple answer is sufficient, while* 1 *indicates that q_m requires a complex answer. Simple answers include words, entities, and short phrases or sentences, while complex answers indicate the necessity for including reasoning.*

Typical verbs can be categorized into the resulting matrix. The verbs serve as keywords for searching questions from a selection of QA datasets. Table 1 shows Bloom's matrix with the assigned keywords. Syntactic and lexical features are extracted from selected candidates. Since a classification into 20 categories is not necessary and would also lead to poor results due to the uneven distribution and small number of candidates, a binary classification is performed. This is also helpful for our use case, since the question classification should indicate two answer types, simple or complex. After separating into the two classes, resampling is done to balance the training data. The following sections describe the processing and training steps in detail.

3.1 Datasets

Most datasets focus on factoid questions where superficial cues alone are sufficient to find an answer. We consider two different sources: the AI2 Reasoning Challenge (ARC) [2] and the Stanford Question Answering Dataset (SQuAD) [9].

Table 1 The knowledge dimension matrix by Cannon and Feinstein [1] filled with key indicators for complex questions. The complexity ranges from low (top left) to very high (bottom right) [11]

	The cognitive process dimension				
The knowledge	1	2	3	4	5
dimension	Remember	Understand	Apply	Analyze	Evaluate
A	Name, list	Restate	State	Distinguish	Select
Factual	Define, label	Order	Determine	Classify	According to
B	Identify	Describe	Illustrate	Examine	Rank
Conceptual	Locate	Explain	Show	Analyze	Compare
C	Tell	Summarize	Solve	Deduct	Conclude
Procedural	Describe	Translate	Demonstrate	Diagram	Choose
D		Interpret	Find out	Infer	Justify
Meta-cognitive	–	Paraphrase	Use	Examine	Judge

3.1.1 ARC

The AI2 Reasoning Challenge (ARC) [2] contains science questions for elementary school students. It is the largest public-domain collection of its kind (7787 multiple-choice questions without diagrams). The questions were taken from 21 different sources, including questions from the California Standards Test and the Arkansas Comprehensive Testing, Assessment, and Accountability Program. It is divided into a set of 2590 complex questions (those that cannot be answered correctly by both a retrieval and a co-occurrence method) and a set of 5197 simple questions. The characteristics of the dataset already indicate the importance of syntactic features such as question length for question difficulty: the "complex dataset" contains longer questions with a min/avg/max of 2/22.3/128, whereas the "simple dataset" contains shorter questions with a min/avg/max of 3/19.4/118 [2].

3.1.2 SQuAD

The Stanford Question Answering Dataset (SQuAD) 1.0 is a reading comprehension dataset consisting of more than 100,000 questions from Wikipedia articles [9]. The dataset contains a wide range of question and answer types that require reasoning. The new version, SQuAD 2.0, also called SQuADRUn, combines the existing SQuAD data with over 50,000 unanswerable questions written by crowdworkers to resemble answerable questions [8]. Since we are interested in a large number and variety of questions, we consider all questions from SQuAD 2.0 that are suitable for our approach.

3.2 Data Preparation

The data preparation steps are (1) performing part-of-speech (PoS) tagging for the questions, (2) stemming all occurring verbs, (3) mapping these verbs to the stemmed verbs in Bloom's matrix, and (4) categorizing the questions accordingly and adding them to the training data. (5) The test dataset for evaluation is created by human annotation.

3.2.1 Keyword Mapping

To generate the training data, the questions from the described datasets are classified by occurring keywords from Bloom's matrix. This is done by stemming the words in each question and searching the stems in the matrix. This helps us to find morphologically matched words with different word endings, such as "describing" or "examines" which are both stemmed to "describ" and "examin," respectively. However, a simple search for stemmed words within the keywords is not sufficient. Some questions contain phrases like "in order to" or the noun phrase "United States." Stemming would place the first in category A2 and the second in category A3. To get around this problem, all words are tagged with their respective PoS tag, and then only verbs are searched in the matrix.

3.2.2 PoS Tagging

The syntactic features are PoS tags. Instead of using lexical tokens or tuples of tokens enriched with syntactic information, we do not train with lexical features. The reason is that we want to derive complexity from the structure of the questions and avoid learning domain-specific complexity. The only lexical features we use are question words, which are domain independent. This means PoS tags are not determined for question words, e.g., "what," "who," "how," "when," "where," "why," "which," "whom," and "whose." In fact, since question words affect the difficulty of a question and their omission would reduce classification accuracy, it is desirable to keep their lexical information. For example, questions containing the question words "how" or "why" require longer, more sophisticated answers than simple factual questions with "who" or "when."

3.2.3 Class Binarization

The classes are very unbalanced, and some categories even contain no samples after keyword assignment. Our goal is to decide whether a question is simple or complex, so it is not necessary to assign 20 different classes. We perform a heuristic split where we look at the training samples and determine whether the question requires

Table 2 Examples for each of the analyzed categories. Each class from Bloom's matrix (A1–D5) is assigned to a category. "Category 0" contains simple questions, while "Category 1" contains complex questions that may require reasoning

No.	Example	Class	Category
1	**Name** the British economist theorist and philosopher who is also an author and alumni	A1	0
2	What cannot be **restated** as decision problems?	A2	0
3	What unit is measured to **determine** circuit simplicity?	A3	0
4	Complexity theory **classifies** problems based on what primary attribute?	A4	0
5	Who **selects** and hires the best ideas and appropriate contractors?	A5	0
6	What river was Petrela **located** by?	B1	0
7	How do physical experiments **explain** fluid inclusion data?	B2	1
8	How does Victoria **rank** as to population density?	B3	1
9	What are the ties that best **described** what the "eight counties" are based on?	C1	1
10	What does "Huisgenoten" **translate** to in French?	C2	1
11	What cannot be **solved** by mechanical application of mathematical steps?	C3	1
12	In whose **diagram** is each matter particle represented as a curved line?	C4	1
13	What did Basset **analyze** before coming to his conclusions?	C5	1
14	What can be **interpreted** by individuals to determine if funding for course content is forbidden?	D2	1
15	What number is **used** in perpendicular computing?	D3	1
16	What is a science that **examines** the structure and function of the brain?	D4	1
17	What could **justify** restrictions on freedom of establishment?	D5	1

Table 3 Number of questions in the SQuAD and ARC datasets. The "Category 0" and "Category 1" columns show the number of questions assigned to the respective categories based on the keywords from Table 1

Dataset	Category 0	Category 1	Total
SQuAD	2517	6325	8842
ARC	242	919	1161

reasoning or not. Table 2 shows a selection of examples and their classification into simple and complex. We decide that category B2 is the first that requires reasoning, since "explain" clearly indicates the need for explanation. Since the complexity in the matrix ranges from low (top left) to high (bottom right), classes with a higher complexity than B2 are assigned to "Category 1" (i.e., complex), while classes in a lower class than B2 are assigned to "Category 0" (i.e., simple). Both classes are balanced; class 0 contains 9,714 training samples, and class 1 contains 10,232 samples. Therefore, no resampling is required.

Table 4 Number of questions classified by the different annotators per dataset

Annotator	SQuAD	ARC
# 1	354	300
# 2	588	366
# 3	354	263

Table 3 shows the distribution of simple and complex questions in the SQuAD and ARC datasets. In both cases, the complex ones (i.e., “Category 1”) dominate.

3.2.4 Test Data

Using the keyword-based approach to create the test data would be very biased, as questions that fall into a particular class of Bloom’s taxonomy can easily be classified without using the keywords provided in Table 1. For instance, if we consider the example “How does photosynthesis work?” and “Explain how photosynthesis works,” both prompt an explanation of the process of photosynthesis, and both would be classified as B2 in Bloom’s taxonomy, one with the keyword “explain” and one without.

For this reason, a total of 2225 annotations were performed by 3 independent annotators (cf. Table 4). Random questions from SQuAD and ARC were selected for annotation. The annotators were asked to classify these questions into what they considered to be the appropriate complexity classes (according to Bloom’s matrix in Table 1). If the annotators are unsure about their choice, they have the option of making a second choice. This allows us to characterize the inherent ambiguity of questions that are removed of their original context.

To generate the test set, majority voting was used for the different questions. Thus, for a question, the class that the majority (in this case, two) of the annotators agreed with was chosen. If all annotators disagreed on a question and no absolute majority was reached, the question was not included in the test set. In addition to the complexity classes, other datasets were created with binary classes. Here, the following mapping from Table 2 was used. This procedure resulted in datasets with the quantities shown in Table 5.

The inter-annotator agreement between the datasets (cf. Table 6) is significantly different. This is primarily due to the fact that there are many simple questions in the SQuAD dataset. Since the binary classes are only an abstraction of Bloom’s taxonomy, the results are better there. More detailed analysis of the data shows that annotators often agree that a question is complex, but not on the exact class.[1] Another part of the disagreement between annotators can be attributed to the different interpretations of some classes. Despite written annotation guidelines and a discussion beforehand, there were some classes here that were interpreted

[1] The annotation and evaluation results, as well as the scripts used for the evaluation, are available in the GitHub repository: https://github.com/amonsoes/bloom_qde.

Table 5 Number of remaining annotations after a majority vote. Annotators disagreed on some questions, so these were not included in the test set

Dataset	Including second choice	Classes	Number of questions in test set
ARC	No	Binary	162
		Bloom	58
	Yes	Binary	162
		Bloom	94
SQuAD	No	Binary	266
		Bloom	177
	Yes	Binary	266
		Bloom	213

Table 6 Inter-annotator agreement (Cohen's κ) using the binary classes and Bloom's taxonomy

	SQuAD		ARC	
Annotators	Binary classes	Bloom's taxonomy	Binary classes	Bloom's taxonomy
# 1 & # 2	0.50	0.27	0.33	0.15
# 1 & # 3	0.39	0.20	0.47	0.24
# 2 & # 3	0.45	0.30	0.37	0.22

differently. The last identified reason for disagreement between annotators is the framing of the questions. One annotator assumed that the questions should be evaluated from the perspective of a human, for example, in an exam situation, while the other annotators evaluated the complexity of creating an answer for an automated answering system.

4 Experiments

In the following, we present how model training and parameter optimization were performed before presenting and discussing the evaluation results.

4.1 Model Training

We perform experiments with two different neural networks. Firstly, we train a multi-layer perceptron (MLP) based on n-grams, where $n = \{1, 2, 3, 4\}$ to capture the sequential question structure. Secondly, we train a long short-term memory (LSTM) because it is a recurrent neural network that automatically captures the sequential structure. For each model, we try two different feature sets. One contains only syntactic features (PoS tags) and question words. Another contains lexical features along with the structural information as tuples. All experiments are trained

Table 7 Detailed results on test set for the MLP and LSTM models. If the model name contains SQuAD, it means that the model was tested on the manually created SQuAD annotations. Accordingly, ARC is tested only on ARC annotations, and ALL refers to the entire test data (for binary classification). Bold numbers indicate the best observed result for a metric over all tested model-data combinations

	Accuracy	Precision	Recall	F1 score
MLP_SQuAD	0.31	0.52	**0.58**	0.28
MLP_ARC	0.31	0.43	0.43	0.31
MLP_ALL	0.33	0.48	0.46	0.31
LSTM_SQuAD	0.42	**0.55**	0.51	0.35
LSTM_ARC	**0.44**	0.40	0.36	**0.37**
LSTM_ALL	0.43	0.47	0.43	0.36

to provide both a binary classification into simple and complex questions and a multi-class classification that categorizes all classes within Bloom's taxonomy.

4.2 Parameter Optimization

The hyperparameters of both models are optimized using a genetic algorithm to find the best combination of parameters. The algorithm iteratively improves the overall fitness of a population (Y) consisting of n chromosomes randomly initialized in the first iteration t [4]. In our case, the fitness v of a chromosome $x \in (Y)$ is determined by the evolutionary accuracy of the hyperparameter sequence of that chromosome. If the fitness of two chromosomes x_p and x_q is high enough, they can generate a new chromosome of the new population $(Y)_{t+1}$ by utilizing the crossover operation that designs a chromosome with a combination of the hyperparameter settings of the original chromosomes [4]. However, since there is an obvious discrepancy between the statistics of the test dataset and the training dataset, the hyperparameter set still needs to be manually adjusted afterwards.

4.3 Experimental Results

The n-gram MLP achieves the best results for $n = \{1, 4\}$, a learning rate of 0.003, 5 layers, 128 units, and a dropout rate of 0.2. The precision for "Category 0" (i.e., simple) is 0.89, while it remains at 0.14 for "Category 1" (i.e., complex). However, the recall for "Category 1" lies at 0.75, illustrating the unbalanced distribution in the test dataset. The overall results in Table 7 show the averaged values of precision and recall across both classes. Best results were obtained with combinations of ARC and SQuAD and with the evaluation of the ARC test set only. The highest precision value of 0.52 is achieved on SQuAD. The overall accuracy remains low at 0.33.

Similarly, the LSTM also favors overrepresented classes in the test set. In general, it is helpful to keep the number of neurons relatively low, firstly to avoid overfitting and secondly to prevent the model from selecting only the overrepresented class. Recall seems to be the biggest issue for the LSTM, and the overall F1 score for the test data is 0.36.

4.4 Room for Improvement

There is a variety of possibilities to further improve our results. A major drawback is the strict separation of machine-labeled training data and human-annotated test data. It would be desirable to annotate a greater number of instances and extend the training data to include human-annotated questions. Also, questions containing Bloom's keywords should be added to the test set to provide more representative data for both training and testing. Furthermore, the number of annotators is relatively small, which was acceptable for our proof of concept, but could definitely be adjusted for larger experiments. Finally, we consider only a small number of features, i.e., PoS tags and question words. Question complexity depends on more than just structural features, so results could be improved by adding lexical features such as word embeddings or language models to the feature set.

5 Conclusion and Future Work

We presented an approach that classifies keywords in QA systems into difficulty levels, as known from Bloom's learner's levels of understanding in pedagogy. We searched for Bloom's keywords in questions from QA systems and derived PoS tags from these questions to identify the structural difference between difficult and simple questions. For binary classification (simple and complex), we defined a threshold between the classes B1 and B2, where all classes up to B1 require words or short entities as answers, while classes B2 or higher require longer, complex answers such as (multi-hop) reasoning. Moreover, we created a new dataset divided into Bloom's taxonomy to provide an objective unbiased test set for evaluation. We trained an MLP and LSTM for question difficulty estimation (called BloomQDE) based on structural features and evaluated both on a human-annotated dataset.

For future work, we plan to consider a greater variety of features, including lexical, structural, and semantic features. In addition, the test set should be extended and labeled by more annotators so that bias can be reduced and inter-annotator agreement can be improved. In this way, human-annotated questions can also be integrated into the training set to obtain a representative classifier for datasets that are equally automatically and manually labeled. Finally, we aim for a more balanced distribution of classes in the new dataset.

References

1. Cannon, H.M., Feinstein, A.H.: Bloom beyond Bloom: Using the revised taxonomy to develop experiential learning strategies. In: Developments in Business Simulation and Experiential Learning: Proceedings of the Annual ABSEL Conference, vol. 32 (2005)
2. Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., Tafjord, O.: Think you have solved question answering? Try arc, the AI2 reasoning challenge. Preprint (2018). arXiv:1803.05457. https://arxiv.org/pdf/1803.05457.pdf
3. Huang, Z., Liu, Q., Chen, E., Zhao, H., Gao, M., Wei, S., Su, Y., Hu, G.: Question difficulty prediction for READING problems in standard tests. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA, pp. 1352–1359. AAAI Press (2017). http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14572
4. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multim. Tools Appl. **80**(5), 8091–8126 (2021). https://doi.org/10.1007/s11042-020-10139-6
5. Liu, J., Wang, Q., Lin, C., Hon, H.: Question difficulty estimation in community question answering services. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18–21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, pp. 85–90. ACL (2013). https://aclanthology.org/D13-1009/
6. Padó, U.: Question Difficulty – How to Estimate Without Norming, How to Use for Automated Grading. In: Tetreault, J.R., Burstein, J., Leacock, C., Yannakoudakis, H. (eds.) Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, BEA@EMNLP 2017, Copenhagen, Denmark, September 8, 2017, pp. 1–10. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/w17-5001
7. Qiu, Z., Wu, X., Fan, W.: Question difficulty prediction for multiple choice problems in medical exams. In: Zhu, W., Tao, D., Cheng, X., Cui, P., Rundensteiner, E.A., Carmel, D., He, Q., Yu, J.X. (eds.) Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019, pp. 139–148. ACM (2019). https://doi.org/10.1145/3357384.3358013
8. Rajpurkar, P., Jia, R., Liang, P.: Know What You Don't Know: Unanswerable Questions for SQuAD. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 784–789 (2018). https://aclanthology.org/P18-2124.pdf
9. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ Questions for Machine Comprehension of Text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2383–2392 (2016). https://aclanthology.org/D16-1264.pdf
10. Sun, J., Moosavi, S., Ramnath, R., Parthasarathy, S.: QDEE: Question Difficulty and Expertise Estimation in Community Question Answering Sites. In: Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM 2018, Stanford, California, USA, June 25–28, 2018, pp. 375–384. AAAI Press (2018). https://aaai.org/ocs/index.php/ICWSM/ICWSM18/paper/view/17854
11. Ullrich, S., Geierhos, M.: Using Bloom's Taxonomy to Classify Question Complexity. In: Proceedings of The Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021), pp. 285–289. Association for Computational Linguistics, Trento, Italy (2021). https://aclanthology.org/2021.icnlsp-1.34
12. Wang, Q., Liu, J., Wang, B., Guo, L.: A regularized competition model for question difficulty estimation in community question answering services. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pp. 1115–1126. ACL (2014). https://doi.org/10.3115/v1/d14-1118

A Comparative Study on Language Models for Dravidian Languages

Rahul Raman, Danish Mohammed Ebadulla, Hridhay Kiran Shetty, and Mamatha H.R.

Abstract We train embeddings for four Dravidian languages, a family of languages spoken by the people of South India. The embeddings are trained using the latest deep learning language models, to successfully encode semantic properties of words. We demonstrate the effect of vocabulary size on word similarity and model performance. We evaluate our models on the downstream task of text classification and small custom similarity tasks. Our best model attains accuracy on par with the current state of the art while being only a fraction of its size. Our models are released on the popular open-source platform HuggingFace. We hope that by publicly releasing our trained models, we will help in accelerating research and easing the effort involved in training embeddings for downstream tasks.

1 Introduction

Distributed representation is the foundation of natural language processing, as advances in language modelling serve as a stepping stone for many NLP tasks. Popular domains like text classification, text generation, translation, sentiment analysis, NER, etc. can be advanced with access to contextualized word embeddings. A rise in quality of embeddings is synonymous with an improvement in downstream NLP tasks.

India is a diverse and rapidly growing country. With advances in technology, electronic devices are making their way into the hands of every citizen of the country, giving them the ability to access information that was previously out of reach for them. But this also presents another problem. India has over 22 official languages and several thousand more languages and dialects. It is of paramount importance that we develop NLP tools that bridge this gap and help India progress faster.

R. Raman (✉) · D. M. Ebadulla · H. K. Shetty · Mamatha H.R.
PES University, Bangalore, India
e-mail: mamathahr@pes.edu

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_8

Indian languages are considered resource poor and have very little monolingual corpora that are publicly available for NLP tasks. Dravidian languages in particular are far behind Indo-Aryan languages. With access to such few resources, training a language model is very challenging, as it is very easy to overfit your model and lose its ability to generalize. Many corpora are also domain specific, making it difficult for the model to generalize context.

In this paper, we experiment with the latest language models on four Dravidian languages: Kannada, Tamil, Telugu and Malayalam. We first train word embedding models and run word similarity tests to evaluate the effect of vocabulary size on model performance and word choice. We then train contextual embedding models on all four languages and evaluate these models on the news article classification provided by indicNLP.[1] We show that lightweight transformer-based models such as RoBERTa [13], DeBERTa [6] and ELECTRA[3] outperform previously used mainstream models. We release these models on the popular transformer's open-source repository HuggingFace[2] where our fine-tuned models, capable of generating quality word embeddings, will significantly improve all Kannada language downstream tasks.

2 Related Work

One of the earliest papers to perform embedding generation on Kannada at scale was fastText by Meta [2]. They proposed an improvised approach for the skip-gram model, representing each word as a bag of character n-grams. This overcame the main drawback in Word2Vec [14], where words were considered as atomic units leading to subpar performance on morphologically rich languages such as Kannada. FastText's embeddings are used as a benchmark for comparison of results in several Indic language model papers.

Kunchukuttan et al. [12] released the indicNLP corpus in 2020, a monolingual corpora for ten Indian languages sourced from various domains and sites. Word embeddings trained on fastText using this corpus were also released. A news classification dataset to be used as a downstream evaluation task was also released. Their embeddings were compared against the original fastText embeddings and were found to outperform the latter in several languages.

Gaurav Arora [1] released the Natural Language Toolkit for Indic languages a few months later, which also released embeddings for 13 Indic languages that outperformed indicNLP and fastText. ULMFiT [7] and Transformer-XL [5] were used to train the embeddings, and the data sourced from Wikipedia was only a fraction of the indicNLP corpora's size. A two-step augmentation technique was used to improve the performance of their models. Kumar Saurav et al. [11] also

[1] https://github.com/AI4Bharat/indicnlp_corpus#indicnlp-news-article-classification-dataset.

[2] https://huggingface.co.

released word embeddings for 14 Indian languages in a single repository, although their results are not competitive with Anoop Kunchukuttam or Gaurav Arora. They trained their embeddings on several transformer architectures such as BERT [9] and ELMo [15] and tested them on several custom tasks.

Kakwani et al. [8] presented the IndicNLPSuite, a collection of large-scale, general-domain, sentence-level corpora of 8.9 billion words across 11 Indian languages, along with pre-trained models and NLU benchmarks available to the public. Yinhan Liu et al. [13] carefully studied the impact of various hyperparameters in pre-training BERT models and released and improved procedure – RoBERTa. Pengcheng He et al. [6] released a model called DeBERTa that improves on BERT and RoBERTa using a disentangled attention mechanism and an enhanced mask decoder. As an alternative to masked language models like BERT, Kevin Clark et al. [3] suggested a discriminative model leveraging replaced token detection called ELECTRA which was shown to be more efficient than BERT particularly for smaller models.

3 Methodology

The following section elaborates on the data pipeline, the preprocessing steps and the experimental setup of this work.

3.1 Dataset

Our pre-training data is sourced from the indicCORP [8], a collection of ten Indic languages. We use a small subset of the datasets available to prove that our models can perform in resource-constrained situations. We use the news classification released by indicNLP for the downstream task of text classification. We also build small custom datasets for word similarity and word analogy tests. To ensure fair comparison for downstream tasks across all models for a particular language, we train all our models on the same corpora.

3.2 Preprocessing

The corpora were cleaned to remove any foreign tokens and fix formatting errors. Shuffling and deduplication were applied after extracting the data subset. An md5 hash was applied to deduplicate the corpora, leaving us with roughly four to five million sentences per language after it was applied on the corpora. To make initial

Table 1 Dataset statistics. Pre-training data is the indicCORP subset used to pre-train our models. News classification data is the indicNLP news category classification dataset

Language	Pre-training	News classification data	
		Train	Test
Kannada	4.07M	24,000	2400
Telugu	4.82M	19,000	2400
Tamil	5.16M	5346	669
Malayalam	5.85M	5036	630

training easier, any sentences greater than 30 words or having English in more than 30% of the sentence were removed. The exact statistics of each language's dataset are given in Table 1.

3.3 Tokenization and Vocabulary

Our word embedding models are tokenized using SentencePiece [10] with varying vocabulary sizes. The RoBERTa and DeBERTa models use the ByteBPE tokenizer, and ELECTRA uses BertWordPiece. With the help of SentencePiece API,[3] tokens were generated by experimenting with the hyperparameters. Vocabulary size ranged from 8000 to 32,000 with incremental steps of 4000. BertWordPiece and ByteBPE were trained to generate a vocabulary size of 32,000 with words having a minimum frequency of 4.

Previous works claim that higher vocabulary sizes correspond to a lower chance of out-of-vocabulary words occurring, and this usually translates to better performance in downstream tasks. But without a morphologically motivated technique to segment subwords, increasing the vocabulary size might lead to an increased occurrence of different inflections of the same word. Hence, we decide to compare varying vocabulary sizes and their performance.

3.4 Experimental Setup

We evaluate our models on the downstream task of text classification using the indicNLP and iNLTK news classification dataset. All information relevant to the datasets is tabulated in Table 1. All models were trained using a single 12GB NVIDIA Tesla K80 GPU.

[3] https://github.com/google/sentencepiece.

4 Models and Evaluation

The following section covers the models we used starting from word embedding models and going up to contextual embeddings. It covers their architecture and the downstream task setup.

4.1 Word Embedding Models

Our word embedding models are trained using the fastText API.[4] The publicly released language model has an approximate vocabulary size of 1.7 million. With the API, we pre-trained a fastText model from scratch with both CBOW and skip-gram architecture. The fastText API takes its input directly and handles the tokenization. Due to very few hyperparameters provided by the fastText API for tuning the model, further experimentation was done with the gensim API. With the gensim API, first the input data was tokenized with SentencePiece. The API provides hyperparameters for tokenization, vocabulary frequency and the architecture which helped us fine-tune our model for better accuracy in the news classification dataset. The API's supervised module was used to perform text classification on the news dataset.

4.2 Contextual Embedding Models

We train a different language model for each language and use three BERT-based architectures: RoBERTa, DeBERTa and ELECTRA. The following subsections will cover these models in more detail.

4.2.1 RoBERTa

Since base BERT models require a large corpus and access to heavy computation resources, we trained embeddings on a RoBERTa model with distilBERT's [16] configuration. When compared to the BERT pre-training technique, one of the key aspects of the design feature in the RoBERTa model is the removal of the next sentence prediction objective from the pre-training phase and the addition of dynamic masking for the training data, which has shown a significant improvement in performance.

[4] https://github.com/facebookresearch/fastText.

Our model was trained using the HuggingFace API. Byte Pair Encoding [17] was used to tokenize the corpus after which the tokenizer weights were transferred to the RoBERTa tokenizer. The vocabulary size was set to 32,000, and the model's configuration was set to 6 hidden layers, 12 attention heads and 768 embedding size. The size of the model was 68 M parameters. After the pre-training phase, two linear layers were added to fine-tune the model on the classification task. We pre-trained the model for 300,000 steps and stopped the model when loss flattened out. Hyperparameters such as batch size, hidden layers, number of attention layers and the embedding size were tuned to accommodate the decreased model size.

4.2.2 DeBERTa

In transformers, the input word vector for the multi-head attention mechanism is a mathematical combination of the word embedding and positional encoding. In absolute positional encoding, used in language models like BERT and RoBERTa, each token will have its own positional encoding vector. But in relative positional embedding, each token will have 'n' (size of the tokenized sentence) positional vectors indicating the positional relation between the current token and other tokens, and these positional vectors are shared amongst all the tokens.

The DeBERTa architecture utilizes relative positional encoding and incorporates two techniques, disentangled attention mechanism and enhanced mask decoder. Disentangled attention mechanism computes the attention weights using disentangled matrices on the word and positional encodings instead of mathematically combining word and positional encodings. The enhanced mask decoder incorporates the absolute positional embedding at the last layer to address the disambiguation relation between the generated word and the context.

The DeBERTa model was also trained using the HuggingFace API with Byte Pair Encoding for tokenization. The vocabulary size was set to 32,000. We used the DeBERTa v2 model with 6 hidden layers, 12 attention heads and an embedding size of 768. The final model had 75M parameters. Pre-training steps of the model are given in Table 2. Two additional linear layers were added to the model during the classification tests.

Table 2 Number of pre-training steps for all our language models

Model	Kannada	Tamil	Telugu	Malayalam
RoBERTa	330K	360K	280K	210K
ELECTRA	200K	200K	200K	200K
DeBERTa	210K	200K	200K	200K

4.2.3 ELECTRA

We also trained embeddings using one of Google research's newer models, ELECTRA. Unlike the RoBERTa and DeBERTa models, which were pre-trained with masked language model task, the ELECTRA model was trained with replaced token detection task. The architecture setup for pre-training comprises of two components: a generator and a discriminator. During the pre-training task, the ELECTRA model predicts whether the sentence has been generated by the BERT model or if the sentence is from the dataset.

The efficiency gains by replacing the token detection approach are due to the loss being defined over all tokens rather than just the mask token (which is the case for MLM) and because there is no masked token discrepancy between pre-training and fine-tuning phases.

Since ELECTRA generates `tf.pretrain` records of the input corpora and stores them offline, it is not limited by memory and is capable of training on large datasets. The model uses the BertWordPiece tokenizer. The vocabulary size was set to 32,000. We used the 'small' version of the model which has 14 M parameters and trained it for 200,000 steps. Maximum sequence length was set to 512. After pre-training the model, it was fine-tuned and evaluated on a text classification task using the ktrain library on the news article dataset.

5 Results

5.1 Word Similarity

We found that our fastText models trained with a vocabulary size of 8,000 had more meaningful similar word predictions compared to the same models with a 32,000 vocabulary size. As a baseline, we also present results on a simple Word2Vec model trained on the same data. Our fastText model's accuracy was marginally lower than the original fastText model. Figure 1 shows some notable results from our experiments on word similarity. Word similarity results were largely comparable across all languages; hence, Fig. 1 only shows the results for the Kannada language. Word2Vec results were observed to be heavily influenced by the domain of the dataset and contained pronouns in word similarity results as it considers word as the atomic token value. In comparison, fastText produces significantly better results at it considers the n-gram characters' information as an atomic unit.

We can also observe that the lower vocabulary models produce words that are synonyms of the input word, while the large vocabulary models produce inflections of the same word. The official fastText model had very different words at the morpheme level, but these words were distinctly similar to the actual word.

Model	**Word Similarity**					
	ರಾಜ (king)			**ಮನುಷ್ಯ (man)**		
FastText_R - CBOW	ರಾಜನಾದ Rājanāda ‘The king’	ಪ್ರವಾದಿ Pravādi ‘Prophet’	997ರಲ್ಲಿ 997Ralli ‘In 997’	ಫರಿಸಾಯನು Pharisāyanu ‘The Pharisee’	ಪುತ್ರನೇ Putranē ‘Son’	ಮನುಷ್ಯನನ್ನು Manuṣyanannu ‘The man’
FastText_R - SG	ದಾವೀದ Dāvīda ‘David’	ಸೊಲೊಮೋನ Solomōna ‘Solomon’	ಅಮಚ್ಯನು Amājiyā ‘Amaziah’	ಬಿದ್ದುಹೋಗುವದು Bidduhōguvadu ‘To fall off’	ಮನುಷ್ಯನ Manuṣyana ‘Man's’	ಇಸ್ರಾಯೇಲನಿಗೆ Isrāyēlanige ‘For the Israelites’
FastText - inltk	ರಾಜನ Rājana ‘King's’	ರಾಜ್ Rāj ‘Raj’	ರಾಜಕೀಯ Rājakīya ‘Politics’	ಮನುಷ್ಯರ Manuṣyara ‘Human beings’	ಭಗವಂತ Bhagavanta ‘Lord’	ದೆವ್ವ Devva ‘Devil’
FastText_G - 8K	ಅರಸ Arasa ‘King’	ಅರಸನಾದ Arasanāda ‘The king’	ಕುಮಾರ Kumāra ‘Son’	ಪುರುಷನು Puruṣanu ‘The man’	ಪುರುಷ Puruṣa ‘Male’	ಮನುಷ್ಯನಿಂದ Manuṣyaninda ‘By man’
FastText_G - 16K	ರಾಮ Rāma ‘Rama’	ಪ್ರವಾದಿ Pravādi ‘Prophet’	ಅರಸ Arasa ‘King’	ಕುಮಾರನು Kumāranu ‘Son’	ಸ್ತ್ರೀಗೆ Strīge ‘Female’	ಹುಡುಗ Huḍuga ‘Boy’
FastText_G - 32K	ನಾಥ Nātha ‘Nath’	ನಾಟ Nāṭa ‘Nata’	ರಾಜನ Rājana ‘King's’	ಹುಡುಗಿಯ Huḍugiya ‘Girl’	ಮನುಷ್ಯನಿಂದ Manuṣyaninda ‘By man’	ಫರಿಸಾಯನು Pharisāyanu ‘The Pharisee’
FastText - pretrained	ಸಿಂಘನನು Siṅghananu ‘Lioness’	ರಾಣಿ Rāṇi ‘Queen’	ಎನುತಲಿ Enutali ‘Chattering’	ಪ್ರವಾದಿಯೂ Pravādiyū ‘Prophetic’	ಪುತ್ರನೇ Putranē ‘Son’	ಮನೆಯವರೇ Maneyavarē ‘Housekeeper’

Fig. 1 *Word similarity*. *FastText_R*: fastText’s Meta Research implementation with character-level embeddings. *FastText_G*: is the gensim implementation which takes in SentencePiece embeddings

Table 3 News article classification results. FT models are all fastText models trained by Kakwani et al. [8]. Our models are italicized

Model	Kannada	Tamil	Telugu	Malayalam
FT-W	95.93	95.99	98.67	89.02
FT-WC	96.53	95.90	98.08	89.18
IndicFT	97.43	97.26	99.17	92.83
RoBERTa	**98.30**	95.36	99.16	94.76
ELECTRA	97.43	91.47	97.29	89.36
DeBERTa	97.96	93.87	99.16	93.01
indicBERT base	97.87	96.60	99.67	93.33
indicBERT large	97.87	95.24	**99.67**	85.33
indicNLP	97.20	97.01	98.79	92.50
XLM-R	97.60	**97.28**	99.33	**96.00**
mBERT	97.87	94.56	98.67	81.33

The bold values indicate the best performing model

5.2 News Article Classification

Our models are compared against Meta Research’s fastText model trained on Wikipedia and Commmon Crawl, fastText models trained by Kakwani et al. [8], indicNLP, indicCORP and large BERT-based models like XLM-R [4] and mBERT on a text classification task using the indicNLP news classification dataset. The results are documented in Table 3.

All three of our contextual embedding models manage to outperform the fastText models. Of our three language models, RoBERTa performs the best, especially in Kannada where it manages to outperform the previous state-of-the-art models as well with a classification accuracy of 98.30%.

The ELECTRA model managed to keep up with the other models despite being considerably smaller than them. Our ELECTRA model is built using the 'small' version with 14 M parameters and was fine-tuned on the text classification task after pre-training for 200,000 steps. The accuracy it obtains is only marginally lower than the other models' accuracy on the same task, despite having a fraction of the parameters. All our models were trained on lesser data and with smaller parameters but still managed to deliver performance comparable to the state of the art. This proves that with the right tokenization and hyperparameter choices, we can overcome the morphological richness of Indic languages and build compact models that can deliver a high level of performance on downstream NLP tasks.

6 Conclusion

In this work, we present a detailed comparative study of language models and their performance on the agglutinative languages of South India. We start our comparison with basic word embedding models like Word2Vec and fastText and build our way up to the latest contextual embedding models like RoBERTa and ELECTRA. We explore the effect of vocabulary size on language models when we use subword segmentation techniques on our corpus. We show that larger vocabulary sizes correspond to the models choosing inflections of the original word in similarity tasks. We also train BERT-based lightweight models like RoBERTa, DeBERTa and ELECTRA and compare them against other state-of-the-art Indic language models. Our models perform on par with their much larger counterparts, and our RoBERTa model achieves the best performance on the news classification task, beating larger models like XLM-R and mBERT. All our models are released on HuggingFace[5] for the open research community to experiment with.

7 Future Work

Future work will involve training contextual embedding models on all Indic languages and uploading them on a popular site like HuggingFace. The models used in this work have proven to be a competitive and efficient choice to develop language models for Indic languages by achieving accuracy on par with much larger and compute-intensive BERT models. BERT-based models have proved to be superior

[5] https://huggingface.co/RahulRaman.

to the previously utilized mainstream models like Word2Vec and fastText. With more training and fine-tuning, lightweight BERT models might even be able to outperform their mainstream counterparts in low-resource settings.

We believe that the vocabulary size dilemma can be overcome by using a linguistically motivated subword segmentation technique like Morfessor.[6] This will help us identify frequently occurring suffixes and eliminate the occurrence of inflections in the vocabulary.

References

1. Arora, G.: inltk: Natural language toolkit for indic languages. In: Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS), pp. 66–71 (2020)
2. Bojanowski, P., Grave, É., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
3. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. In: International Conference on Learning Representations (2019)
4. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. In: ACL (2020)
5. Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2978–2988 (2019)
6. He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention. In: International Conference on Learning Representations (2020)
7. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 328–339. Association for Computational Linguistics, Melbourne, Australia (2018). https://doi.org/10.18653/v1/P18-1031. https://aclanthology.org/P18-1031
8. Kakwani, D., Kunchukuttan, A., Golla, S., Gokul, N., Bhattacharyya, A., Khapra, M.M., Kumar, P.: Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 4948–4961 (2020)
9. Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, pp. 4171–4186 (2019)
10. Kudo, T., Richardson, J.: Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 66–71 (2018)
11. Kumar, S., Kumar, S., Kanojia, D., Bhattacharyya, P.: "a passage to India": Pre-trained word embeddings for Indian languages. In: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), pp. 352–357. European Language Resources association, Marseille, France (2020). https://aclanthology.org/2020.sltu-1.49
12. Kunchukuttan, A., Kakwani, D., Golla, S., N.C., G., Bhattacharyya, A., Khapra, M.M., Kumar, P.: Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages.

[6] https://github.com/aalto-speech/morfessor.

Preprint (2020). arXiv:2005.00085

13. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. Preprint (2019). arXiv:1907.11692
14. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (2013)
15. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations (2018)
16. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. Preprint (2019). arXiv:1910.01108
17. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715–1725 (2016)

Arabic Named Entity Recognition with a CRF Model Based on Transformer Architecture

Muhammad Al-Qurishi, Riad Souissi, and Sarah Al-Qaseemi

Abstract Named Entity Recognition (NER) is one of the most important tasks of Natural Language Processing that has a very well-defined purpose. There are multiple methodologies suitable for identification of a named entity in a wider text, each with its own advantages. One methodology that has been demonstrated to be very effective involves local versions of deep learning algorithms based on BERT, where ARBERT/MARBERT and AraBERT represent some of the best known implementations in the Arabic language. In this chapter, we introduce a simple model capable of performing NER task with Arabic content. This model is composed of three separate layers—one based on Transformer architecture, one fully connected, and finally one based on the Conditional Random Field technique. This model performs quite well compared to other NER tools for Arabic, reaching F1-scores of nearly 90% on three relevant datasets (ANERCorp, AQMAR, and CANERCorp). We also evaluate our model on one of our customized datasets of Arabic legal textual content. The model achieved 85% which is the highest results of F1-macro score comparing to other related models.

1 Introduction

Named Entity Recognition (NER) is a group of techniques for recognition of several types of named entities (i.e., persons, organizations, locations, etc.) within a corpus of text. Since the linguistic rules guiding the construction of sentences and word sequencing are language-specific, NER tools are far more effective when they are developed for a certain language. Accurate performance of the NER task has a lot of practical values and sometimes represents a precondition for the completion of more complex linguistic operations. Unusual morphology and syntax of the Arabic language infuse additional difficulty into this problem, so different methods need

M. Al-Qurishi (✉) · R. Souissi · S. Al-Qaseemi
Research Department, Elm Company, Riyadh, Saudi Arabia
e-mail: mualqurishi@elm.sa; souissi@elm.sa; salqaseemi@elm.sa

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_9

to be developed to properly classify Arabic words and detect named entities. Broad global distribution of Arabic speakers and the existence of large number of regional variations emphasize the need to create versatile and customizable tools that could be equally effective regardless of the dialect [19, 38].

So far, only a few NER algorithms specialized for the Arabic language had been created, and those have been practically tested only in a limited capacity and their results have been uneven. There are several examples of NER tools based on machine learning principles, such as MADAMIRA [34], FARASA [1], and CAMeL [31]. Probably, the best way to overcome the limitations shared by those works is to establish a sound methodological framework that would facilitate more efficient processing of the large corpus of Arabic text samples that has already been collected. Such a framework would immensely help the researchers in multiple fields and potentially lead to more diverse application of NLP technologies for tasks such as semantic analysis of data in various media and identification of unique entries within the big data paradigm. While the approach relying on automation is not new in NLP and has already been well described in the literature, there are numerous recent studies that provide valuable insights into NER techniques, such as [19, 27]. Early wave of NER research was mostly focused on simple and straightforward learning mechanisms [10, 12, 32], followed by studies that leverage grammatical structure of the language to identify named entities [24]. In the recent years, a new generation of powerful linguistic tools has been developed, leading to general increase of accuracy with many tasks, NER included. Those tools are derived from the BERT algorithm developed by Devlin et al. in 2018 [16], taking advantage of its extreme flexibility and suitability for localization. Consequently, many BERT variations in local languages have been created, with ARBERT/MARBERT [3] and ARABERT [8] as the most successful Arabic language models to date.

This study is an extension to our previous work in [5]. We propose a model based on the Transformers architecture that can successfully tag Arabic language text and identify named entities within it. The model has three separate layers organized in the following way. Bidirectional encoder–decoder model represents the foundation and is generated by pretraining and fine-tuning of the Arabic BERT. The next layer consists of a fully connected layer and serves to optimize the outgoing values, thus setting up the initial weights for the next step in the process. Finally, the last layer includes a CRF (conditional random field) algorithm, which decides which tag to associate with a particular word based on neighboring tags. After processing the extracted features and their weights, this algorithm calculates the likelihood of the analyzed word being a named entity. In this manner, a highly complex semantic analysis problem can be simplified and its solution derived from statistical operations. At the same time, linear analysis of nearby words ensures that context of the sentence is preserved. This mixture of sophisticated analytic capacity and relatively simple structure makes CRF very popular for the purpose of NER tagging. Three well-known public datasets were used to train and evaluate the solution, ANERCorp [11], AQMAR [37], and CANERCorpus [36]. During the evaluation, the three-layered model was found to perform better than any currently available alternatives, and this result was repeated on three datasets. In addition, we evaluate our model on one of our customized datasets of Arabic legal textual content. The model achieved 85%, which is the highest results of F1-macro score comparing to other related models. The organization of this chapter is constructed as follows. Section 2 introduces a brief background about the used language models. Section 3 explains existing works related to this solution, Sect. 4 provides a detailed blueprint of the suggested model, Sect. 5 explains the setup of the experimental evaluation, Sect. 6 reports the main findings, while Sect. 7 draws some conclusions about the value of the proposed methodology.

2 Background

In this section, we will try to give a brief background about the transformer-based language models that we used in this chapter.

2.1 AraBERT

The idea behind AraBERT is to localize BERT architecture for linguistic analysis and use it for completing NLP tasks in the Arabic language [8]. By training it with Arabic content and adjusting for its specificities, the authors created their own model named AraBERT, harnessing the deductive power of machine learning and maximizing its effectiveness with limited availability of good datasets. In the first stage, data preprocessing was performed, with masked language modeling (MLM) where entire words were used as tokens as 80% of them were masked, 10 percent were replaced with a random token, and only 10% were left in their natural state.

This procedure allows the algorithms to derive conclusions based on entire words rather than just linguistic elements, which is better suited for the Arabic language. Next, Sentence Prediction task was used as well in order to teach the model about the relationships between different sentences. Due to relative scarcity of publicly available large-scale resources in Arabic, the training dataset had to be manually collected from several sources and included many regional variations as well as Latin characters in personal names. The final size of the training set after the removal of duplicates was approximately 70 million sentences. To account for the unique nature of Arabic prefixes, sub-word segmentation was performed to separate all tokens into prefixes, stems, and suffixes.

This resulted in a vocabulary of around 60K words, which was used to pretrain the model and create its regional AraBERT variation. Another version was created without segmentation in order to illustrate the benefits of this procedure. After this step, fine-tuning for specific tasks was performed, with downstream linguistic tasks including Named Entity Recognition, Sentiment Analysis, and Question Answering as the main requirements. For the NER task, IOB2 format was used to facilitate multi-class differentiation and recognize entities that consist of more than two words. Final version of AraBERT was evaluated on the same tasks that it was fine-tuned for and compared against existing baselines to check whether it creates any tangible improvement. It was found that pretraining of the model with a large dataset significantly improves the performance on all linguistic tasks over any alternative methods, including the multilingual BERT model that inspired AraBERT.

2.2 AraELECTRA

The success of linguistic models based on the transformer architecture has inspired numerous works on the local or regional level, with the objective to use a proven blueprint and customize it for a particular language or thematic field. BERT [16] and ELECTRA [14] are well-known examples of deep models capable of mastering demanding NLP tasks with adequate training, and there have been several implementations of similar designs aimed at Arabic semantic analysis.

The original ELECTRA model is pretrained based on the masked language modeling (MLM) task, while AraELECTRA [9] model (as the localization Arabic version of ELACTRA) was pretrained using the replaced token detection (RTD) technique. This was done to address an inherent weakness where the algorithm was able to keep track of tokens only during pretraining but not in the fine-tuning stage.

Two different neural models were trained with RTD, one acting as the generator (G) and the other as the discriminator (D). G model has fewer layers and attention heads than D model, which is nearly identical to BERT only with an additional linear classification layer on top of the existing architecture. The output of the G model is fed directly into the discriminator component, which attempts to interpret which tokens from the original input were replaced based on the previously seen examples. In the fine-tuning stage, three new tasks were introduced—named entity recognition (NER), questions answering (QA), and sentiment analysis (SA), validating the level of linguistic comprehension of the model and refining its ability to conduct specific predictions.

As opposed to some similar methods based on adversarial relations between two models, AraELECTRA deploys the principle of maximal likelihood and uses meaningful input with replaced tokens rather than randomly generated content. Pretraining was conducted using the same datasets that were originally used for the benchmark solution, AraBERT, allowing for head-to-head comparisons between them along with several other Arabic language linguistic models built with similar architecture, including ALBERT with localizations, Arabic BERT [35], and ARBERT [3].

In sum, 15% of all tokens were replaced during pretraining, which took a total of 24 days to complete. All hyperparameters were standardized, as it was deemed too computationally expensive to conduct an optimization procedure and discover the most suitable values for learning rate, batch size, or sequence length.

2.3 RoBERTa

Starting from the well-known BERT model that was proven to be very effective with lots of different automated linguistic tasks, the authors develop a specific version of the algorithm aimed at achieving maximum accuracy. They retain the basic Transformer architecture that includes a number of stacked bidirectional encoders and decoders with self-attention heads and hidden dimensions, but they optimize the training procedure to include larger and more diverse datasets. Training parameters were also modified to various extents to increase the efficiency of the process and acquire further gains. This implementation with robust optimization was named RoBERTa [28], and it was intended to be a refinement of the original BERT algorithm.

RoBERTa was trained on the same tasks that were used in the formulation of BERT, including Masked Language Modeling where tokens are randomly replaced by "masks" and the algorithm tries to predict their values. The second training task was namely Next Sentence Prediction, where the algorithm attempts to decide whether two sentences logically follow each other or not. However, after tracking the contributions of each task, the authors concluded that the model can be improved if the Next Sentence Prediction objective was eliminated from consideration, thereby simplifying training. Parameter optimization included control of multiple factors such as learning rate, weight decay, batch size, and dropout rate, with the idea to discover the most favorable combination that allows for maximizing predictive abilities of the model. Sequence length was limited to 512 tokens in this implementation, with up to 256 sequences per batch. In contrast to BERT, in this case, only full-length sequences were used without the insertion of shorter sentences early in the training cycle. Training was conducted using five diverse datasets compiled from different sources, including the one used in the original BERT study; all datasets contained English language material only. All of the datasets contained large volumes of documents, in line with the objective to maximize the performance of the algorithm regardless of the time constraints.

The impact of the modifications implemented by the authors was assessed by benchmarking RobERTa against basic BERT implementation on three different evaluation tools designed to measure linguistic aptitude—GLUE, RACE, and SQuAD. This comparison clearly demonstrated that with improved training Transformer architecture can be utilized more effectively and model's capacity for contextual analysis deepened. RoBERTa represents a product of this testing, as the finalized model uses optimized procedures and hyperparameters that were obtained by testing.

3 Related Works

While most existing automated NER systems are made for English language content, in the recent years, specialized systems for Arabic are increasingly receiving attention. Those systems differ in terms of complexity and basic methodology, as they employ a wide range of analytic procedures to detect named entities. In the following sections, the most important approaches used to extract named entities will be explained.

3.1 Rule-Based Approach

Rule-based methods for Named Entity Recognition in Arabic were among the first to be proposed, as they represent a simple extension of grammatical rules. These methods in general rely on specific expert knowledge about the language provided by gazetteers or human linguists and require handcrafted features to be developed specifically for the purpose of recognizing named entities. As such, they tend to be labor intensive and poorly suited for creation of unsupervised NER tools. Some of the best known systems of this kind include TAGARAB [29], NERA [39], as well as a solution based on BPC features proposed [10]. The authors in [17] analyzed the structure of the Arabic sentences and searched for indicators of named entities using rules based on heuristics. Another approach that was proven very successful was the use of the POS morpho-syntactic tag, which can be utilized to find boundaries of named entities within a sentence. This approach was demonstrated by [18], yielding solid results and motivating other authors to experiment with the same concept. Some of the most notable works that rely primarily on morphologic and syntactic rules include [4, 38, 42]. Some of those methods achieve levels of accuracy comparable to machine learning approaches, especially when used on a suitable dataset. They take full advantage of language-specific rules to detect situations where appearance of a named entity is highly likely, which is very important in a language as complex and unique as Arabic. However, the time needed to create highly predictive features and accurately tag input sequences is too great for such methods to be fully scalable. For this reason, rule-based methods are increasingly being combined with other approaches in hybrid arrangements.

3.2 Machine Learning Approach

Various machine learning techniques have been proven effective for linguistic tasks, and many authors developed Arabic NER tools based on them. Such methods typically rely on learning algorithms to capture the relationships between words and identify named entities based on statistical calculations. Many different machine learning approaches have been used for linguistic tasks that require accurate NER tagging, with Conditional Random Fields, Support Vector Machines, Hidden Markov Models, Maximum Entropy, and Decision Trees among the most popular. Some of the first NER tools that could be used with Arabic text were language independent [41]. In 2007, Benajiba et al. [6] formulated ANERsys, which is based on ME approach, while the same authors experimented with CRF and HMM algorithms in the later years with the goal to discover the most optimal settings. The authors in [30] used a three-stage procedure that included transliteration of Arabic text into Latin characters and the use of an Artificial Neural Network classifier. A very interesting study was completed by [2], combining two different machine learning methodologies to facilitate accurate recognition of 10 different entity classes. More recent examples of Arabic NER solutions are based on machine learning [20–23], with increasingly accurate performance. As opposed to methods with handcrafted features, tools from this group are well suited to work with an open-ended vocabulary and very large datasets, which are important practical advantages. While they eliminate the need for knowledge bases, those methods require high-quality datasets to be sufficiently trained, which can present a difficulty considering the limited availability of linguistic resources in Arabic and the differences between regional variants of this language. Under optimal conditions, many tools from this group perform very well and identify even those named entities that can be ambiguously interpreted.

3.3 Deep Learning Approach

Further sophistication of self-training algorithms led to the development of high-powered neural networks and multilayered learning modules that are able to track a huge number of latent connections within the text. Those tools are typically very complex and involve several stages such as feature engineering, model training, and classification. There are many different network architectures that can be used to process linguistic tasks, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and many other types. One model in particular was proven to be very effective with linguistic tasks including NER—BERT algorithm developed by Devlin et al. [16]. This model is built upon Transformer architecture and features a stack of layers that process words and sentences in both directions. One of the greatest advantages of BERT is the possibility to pretrain it on specific datasets, allowing for the creation of highly specialized versions of the model that achieve even better results in a certain language or with specific topics. Consequently, BERT has been widely customized to create more narrowly focused tools, including several implementations in the Arabic language [22]. Probably, the best known examples of this approach are AraBERT, proposed by [8] in 2020 and ARBERT/MARBERT that was created by [3] and takes into account multiple dialects of Arabic spoken in different countries. More recently, Arabic language tools specialized for thematic niches started to emerge, with AbioNER [13], which is completely dedicated to the biomedical domain as a great example. Deep learning tools generally outperform older methods by a steady margin and can recognize named entities based on contextual clues carried by distant words. On the negative side, they also tend to be far more computationally demanding, especially when working with sizable vocabularies and very large training samples. From this perspective, creation of language-specific tools is a more promising direction of research, as localized versions of deep learning algorithms can achieve high levels of accuracy with fewer training examples. However, one precondition for this development is improved availability of labeled training materials in Arabic that would be publically shared for benchmarking and cross-referencing. The appearance of better datasets created specifically for Arabic NER will enable more optimal pretraining of Transformer-based models, thus creating the possibility to work on efficient and highly accurate systems for named entity recognition in many forms of Arabic written and spoken text.

3.4 Hybrid Approach

To leverage the advantages of various methods for named entity recognition, some authors attempted to combine different approaches and create hybrid solutions. This usually involves using separate tools to handle different phases of the process, i.e., using one algorithm to extract features and another in the role of a classifier. It is possible to use multiple tools of the same type (i.e., two machine learning methods) or to combine approaches and use a simpler technique alongside a more complex one. For example, the authors in [1, 34] attempted to create hybrid frameworks by using rule-based methods along with machine learning methods. Another work from this group was conducted by [33], reaching high-accuracy figures on multiple categories of named entities with a hybrid algorithm. On the other side of the spectrum, the work by [7] demonstrated that two highly advanced methods such as BERT and bidirectional GRU can be effectively merged and optimized to analyze Arabic language content. Due to their effectiveness and practical benefits, hybrid methods are regarded as very promising, and it can be expected that more such models will be developed for Arabic NER in the near future.

4 Transformer-Based CRF Model

BERT and all of the derivative models feature an identical architecture which was directly inspired by Transformer [40]. In all cases, there are two-layer stacks—decoder stack and encoder stack, each of which must contain one attention layer at the bottom. Self-attention mechanism in the decoder stack encodes the semantic relationships from the input sequence as attention scores and passes their normalized

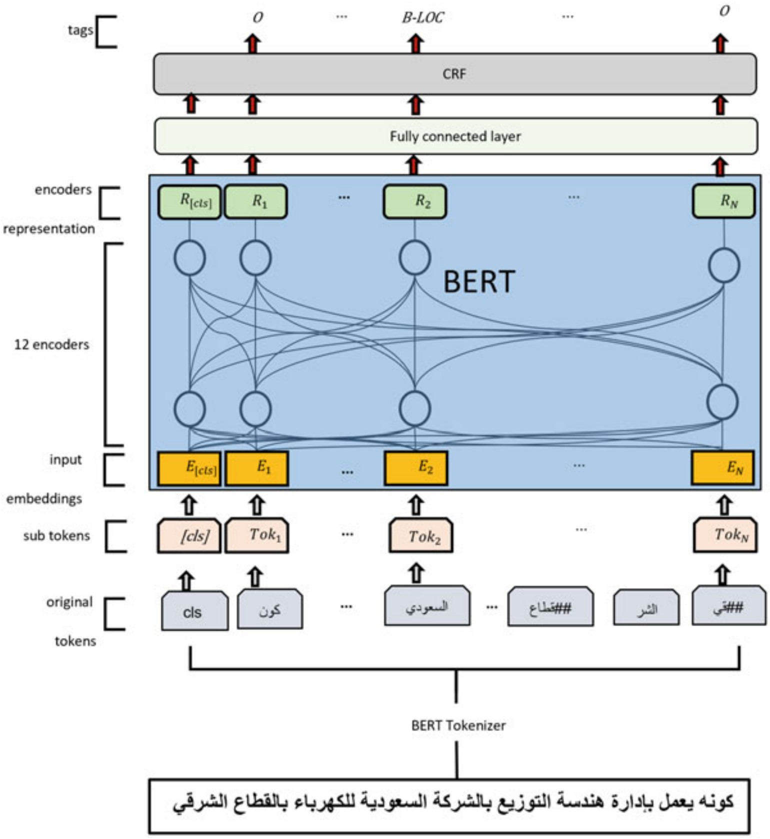


Fig. 1 Basic layout of the model. Words are sent into a BERT-CRF model. Tokens are used to contextualize the word, to build the final representation

values to a series of forward-propagating layers. Conversely, in the encoder stack, the representations are gradually refined with each new layer, until the correct output sequence can be produced. The number of layers and self-attention heads in the model can be variable, while the model is capable of processing semantic information in both directions, thus treating the entire sequence as a single connected unit. This simple setup is now broadly accepted as the basis for advanced linguistic tasks, but it can be greatly improved through pretraining on certain supervised benchmark tasks, as well as fine-tuning of relevant model hyperparameters. Our proposed architecture consists of three layers, as shown in Fig. 1. In the first layer, we fine-tuned three pretrained models, including AraBERT [8], AraELECTRA [9], and XLM-Roberta [15] for Arabic Named Entity Recognition task. The second layer is a fully connected linear layer that receives the output of the pretrained model and reshapes it to be an input for the third layer. The final layer is the conditional random field (CRF), which is the tagging algorithm. The CRF makes sure that the objective of the model training is to return the most accurate combination of outgoing tags. We applied a dropout procedure in order to keep the training procedure well balanced between entity classes.

4.1 Proposed Model Architecture

Figure 1 illustrates the proposed model architecture. If we have an input sentence x, it has to be tokenized before it is entered into the BERT model. In this process, the sentence x is padded to reach the maximum length of the sequence. Tokenized x with the corresponding attention mask is passed to the model, which outputs contextual embeddings of x. Transformer models including BERT use numerous separate attention mechanisms in each layer, in case of BERT base model, a total of 12 x 12 attention heads. In practice, this means that each token can be connected with 12 distinct features of any other token in the sequence. There are two major aspects of BERT output crucial for accurate classification—prediction scores and hidden states. The prediction scores are obtained as the output of the last layer of the model and thus represent the result of all attention heads in all layers and are relative to parameters such as batch size, hidden states size, and sequence length. Meanwhile, hidden states are the outputs of the individual layers of the model, and their total number is equal to the number of layers + 1 (12+1 for BERT). The output of one layer is immediately used as input for the next layer, which contextualizes its content further using its own attention heads. Thus, the prediction score basically represents a hidden state created by the final layer in the BERT model. This output of the model can be understood in different ways. Intuitively, it seems logical that the last layer should contain all the information gained through different stages of learning, so its output should be seen as relevant regardless of the way the input vectors changed as they passed through the layers. On the other hand, it is quite possible that some of the vector modifications accidentally eliminated bits of useful information that could have contributed to an accurate prediction. To compensate for it, the input vectors can be partially or completely concatenated, or their sum could be used. We noticed that this procedure provides significant gains in terms of accuracy improvement, which is why we used this technique in our experiments. The model also includes a linear layer that helps to reshape the output of BERT into 3 dimensions (batch size, number of tags, and sequence length) and then pass it to a CRF layer tasked with making a prediction regarding the probabilities for each of the tags.

4.2 Linear Layer

Linear layer of the model serves to apply a linear transformation on the data arriving from the BERT-based model. This transformation can be summarized as

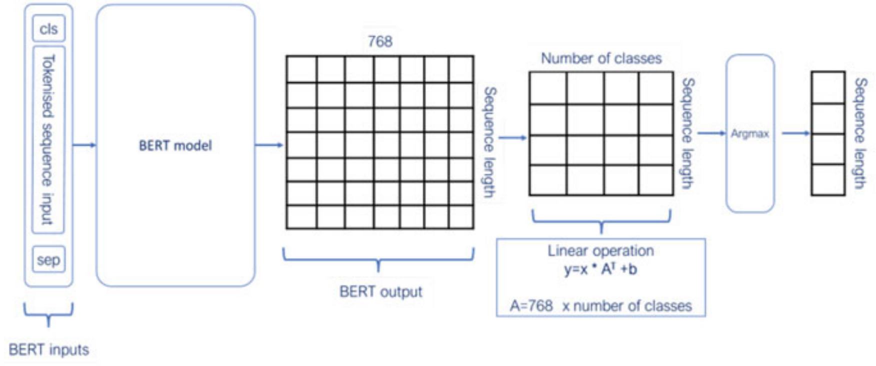


Fig. 2 The BERT output matrix with dimensions 786 x sequence length, the matrix of linear weights sized at 786 x number of classes, and the matrix that is the product of the linear transformations and has dimensions equal to sequence length x the number of classes

$y = x.A^T + b$, where A^T denotes model weight that is shaped by 786 dimensions (the number of classes). As we can see in Fig. 2, for each sequence, several different matrices are constructed, including the BERT output matrix with dimensions 786 x sequence length, the matrix of linear weights sized at 786 x number of classes, and the matrix that is the product of the linear transformations and has dimensions equal to sequence length x the number of classes. Key parameters of this model include linear weights, which represent the learnable weights determined based on the incoming and outgoing features, as well as the linear bias which can be learned based on solely the output features. All values are initialized starting from the formula $U(-\sqrt{k}, \sqrt{k})$, where $k = 1/(in_{features})$. In the fine-tuning stage, both the weights and the bias are adjusted in the linear layer.

4.3 CRF Tagging Algorithm

Using transformer-based models, we get full distribution over the potential labels of the actual input possibilities. From this distribution, the classifier can predict the most likely class that the input could belong to. Named Entity Recognition meets this description, as the need to apply rules for semantic interpretation stretches it to a breaking point. That is because I-PER cannot match B-LOC under such conditions, effectively preventing the algorithm from maintaining its independence. Because of this, the selection of tags is performed using the conditional random field (CRF) method.

CRF method was originally publicized in the early 2000s [25] and has since been much discussed and broadly transferred to a number of different fields. This machine learning approach has proven to be useful in applications as diverse as genetic sequencing and linguistics. In the recent years, models of this type are frequently used alongside LSTM (Long Short-Term Memory) networks to obtain the highest performance possible. This is especially noticeable in the natural language processing field, where this combination is increasingly becoming a standard due to its ability to raise the accuracy level for all tasks where tagging of token sequences is required. Classification of token sequences has the ultimate goal to determine the likelihood that a particular sequence belongs to the class Y from incoming data in vector form X. This section will present a simple type of conditional random field known as the linear chain conditional random field [26]. We motivate the use of conditional random fields in the context of named entity classification where we want to be able to jointly model the full sequence of labels y associated with a sequence of inputs X as follows: For a given training set $\{(X, y)\}$, where $X = [x_1, \ldots, x_k]$ is a set of inputs and $y = [y_1, \ldots, y_k]$ is a target sequences, we calculate the conditional probability $p(y|X)$ as follows:

$$p(y|X) = \prod_{i=1}^{k} p(y_i|x_i) = \prod_{i=1}^{k} \frac{\exp(E(x_i, y_i))}{Z(x_i)} = \frac{\exp(\sum_{i=1}^{k}(E(x_i, y_i)))}{\prod_{i=1}^{k} Z(x_i)} \tag{1}$$

From Eq. 1, for a pair (X, y), the normal classification task can be resolved by calculating $P(y|X)$ in the way of multiplication of individual probabilities for all tokens in the sequence up to the position i, where the value of i is greater than one but lower than the total length of the sequence k. Normalization with exponential function is used in this model, rather than the softmax function that is commonly deployed in the same role in most deep learning models.

Two central variables in this equation of the model are marked by E and Z, with the following definitions:

$E(x, y)$ denotes the emission score and represents the score assigned for the class y based on the vector x after i iterations. In other words, it represents the output of a BERT model after i steps are completed. While the input vector can contain practically any type of information, it is typically populated by a combination of nearby tokens, i.e., words or sentence representations. Each score is assigned its relative weight based on the results of training BERT model.

$Z(x)$ refers to the partition function and can be viewed as a specific type of normalization function as it serves to discover a distribution of probabilities. All probabilities for different classes must always add up to 1. In this sense, it is a component of the softmax activation and can be calculated as follows:

$$Z(X) \;=\; \sum_{y'_1} \sum_{y'_2} \ldots \sum_{y'_i} \ldots \sum_{y'_k} \exp(\sum_{i=1}^{k} E(x_i, y'_i) \;+\; \sum_{i=1}^{k-1} V(y'_i, y'_{i+1})) \tag{2}$$

Due to numerous loops in the model, calculating the value of $Z(X)$ is not simple. This requires considering all iterations of the input for each step in the model, necessitating k repetitions of all calculations to get the value for the entire set.

While the complexity of this procedure is very high $O(|y|^k)$, it is feasible to use the recurrent properties of the model and decrease the computational requirements. This can be accomplished with the forward/backward algorithm, which is capable of processing the sequence in either direction. Once this parameter is determined, the CRF implementation can be undertaken.

Above is described a standard model that calculates probabilities for each class, but we need to expand it by introducing ponders that can be adjusted through learning. Basically, this means the possibility of following up the label y_i with y_{i+1} can be quantified, linking the neighboring labels to each other, which is why this variation is named Conditional Random Field with a linear chain. All probabilities are thus factored with $P(y_{i+1}|y_i)$, using the exponential function to reformat this value as an emission score $E(x, y)$ expanded by the transition score $V(y_i, y_{i+1})$ as follows:

$$p(y|X) = \frac{\exp(\sum_{i=1}^{k} E(x_i, y_i) + \sum_{i=1}^{k-1} V(y_i, y_{i+1}))}{Z(X)} \tag{3}$$

Parameter $V(y_i, y_{i+1})$ is defined as a matrix consisting of elements obtained through learning, while the model transitions from the position i in the sequence to the position $i + 1$ in the same sequence. In other words, it shows the chance that y_{i+1} succeeds after y_i.

4.4 Calculating the NLL Function

With every classification task, a crucial concern is to keep the errors to a minimum, while the model is being trained with input data. A common way to achieve this goal is to use a loss function (L) and feed model predictions into it along with the accurate labels. This function has two possible outputs, 0 or a value greater than 0, depending on whether the two input values match or not. When probabilities $P(y|X)$ are calculated, it is obviously important to eliminate erroneous predictions. This problem can be solved by using the negative log value of the probability of error. This quantity is often referred to as the loss based on negative log probabilities or NLL loss. It can be summarized with the formula $L = -log(P(y|X))$ and modulated with different types of log properties as follows:

$$\begin{aligned} -log(P(y|X)) &= -log(\frac{\exp(\sum_{i=1}^{k} E(x_i, y_i) + \sum_{i=1}^{k-1} V(y_i, y_{i+1}))}{Z(X)}) \\ &= log(Z(X)) - log(\exp(\sum_{i=1}^{k} E(x_i, y_i) + \sum_{i=1}^{k-1} V(y_i, y_{i+1}))) \end{aligned}$$

$$= log(Z(X)) - (\sum_{i=1}^{k} E(x_i, y_i) + \sum_{i=1}^{k-1} V(y_i, y_{i+1})) \qquad (4)$$

The quantity $log(Z)$ denotes the logarithmic value calculated while the partition was performed [26]. This value is very useful for the implementation of the forward algorithm. NLL loss represents the forward pass and can be calculated by reversing the sign before a normal value of log probabilities. Those probabilities can be obtained by calculating all the scores based on the partition and determining the difference between them. This procedure can be made significantly more efficient by the use of a mask matrix, which allows the model to skip any operations that refer to non-essential elements.

5 Experiment

In this section, we will evaluate the training techniques used to improve the algorithm, as well as its performance with different tasks and objectives and the relationship between architecture of the proposed model and quality of the output.

5.1 Tagging Types

The ultimate objective of NER procedure is to associate a label belonging with a particular class to each included word. It is important to note that some complex named entities could stretch across multiple words but are always contained in a single sentence. The predominant sentence representation form used in this field is IOB, with words that start a name of an entity marked with B, internally located words marked as I, and other tokens marked with O.

5.2 Data Samples

The model was evaluated using four Arabic available datasets including ANERcorp, AQMAR, CANERCorpus, and our dataset (Arabic legal content ALC). They are formulated specifically for the Arabic NER task.

5.2.1 ANERcorp Dataset

ANERcorp is a high-quality, annotated dataset containing Arabic language content gathered from a variety of publically available media sources. It was created in

Table 1 Overview of sources of articles container in ANERcorp dataset

Source	Ratio %
http://www.aljazeera.net	34.8
http://www.raya.com	15.5
http://ar.wikipedia.org	6.6
http://www.alalam.ma	5.4
http://www.ahram.eg.org	5.4
http://www.alittihad.ae	3.5
http://www.bbc.co.uk/arabic/	3.5
http://arabic.cnn.com	2.8
http://www.addustour.com	2.8
http://kassioun.org	1.9
Other newspapers and magazines	17.8

2008 and has since been widely accepted as one of the standard datasets used for a variety of linguistic tasks. There are four classes of named entities included in this set, namely Persons, Locations, Organizations, and Miscellaneous. Based on those classes, the dataset includes a number of tags that pertain to named entities, including: *B-PERS*: Beginning of the name of a PERSon *I-PERS*: Continuation (Inside element) of the name of a PERSon *B-LOC*: Beginning of the name of a LOCation *I-LOC*: Inside element present within the name of a LOCation *B-ORG*: Beginning of the name of an ORGanization *I-ORG*: Inside element present within the name of an ORGanization *B-MISC*: Beginning of the name of an entity which doesn't belong to any of the previous classes (Miscellaneous) *I-MISC*: Inside element present within the name of a miscellaneous entity *O*: The word is not a named entity (Other)

There are a total of 316 articles within the ANERcorp dataset, all taken from journalistic sources and online publications. An overview of the content of the dataset regarding the sources, expressed in percentages as in Table 1.

In total, there are more than 150,000 tokens of more than 32,000 types within this dataset, with an average of 4.67 tokens per type. 11% of the content consists of proper names. In collaboration between the original creator of the corpus Yassine Benajiba and the researchers from CAMeL Lab and Mind Lab, the dataset was slightly revised in 2020 to correct some imperfections and make it better suited for the type of studies it is needed for. Some of the corrections agreed upon involved spelling errors, blank Unicode characters, and diacritical marks. At this time, the dataset was also divided into a training portion (125K words) and testing portion (25K words) to facilitate even better performance. In this study, the latest version of the ANERcorp corpus was used.

5.2.2 AQMAR Dataset

AQMAR (American and Qatari Modeling of Arabic) is a relatively small Arabic language dataset created specifically for the purpose of natural language processing evaluation, including named entity recognition (NER). It was created in collaboration between the US-based Carnegie Mellon University and Qatari governmental institutions and is widely used in projects of various types. This dataset consists of more than 3000 sentences taken from around 30 representative Wikipedia articles in Arabic, touching on a wide range of topics from history and science to politics and sports. This dataset was manually annotated with four standard NER classes—persons, locations, organizations, and miscellaneous, in addition to other many tags pertaining to sentiments, relations, etc. The total number of tokens contained in the AQMAR dataset is at 74,000, providing more than enough variability for NLP research purposes.

5.2.3 CANERCorpus Dataset

CANERCorpus stands for a Classical Arabic Named Entity Recognition Corpus that was built by [36] in 2018. This dataset was used by [7], and we compare our proposed model with theirs as shown in Table 6. CANERCorpus was compiled starting from more than 7000 hadiths—religious texts written by Islamic scholars that include mentions of a large number of named entities, totaling around 250 thousand words. There were approximately 13,000 named entities identified in the reviewed texts, and they were separated in 20 different classes based on the general category they are related to. In the preprocessing stage [36], the texts were segmented into sentences before they were annotated by three human operators, with majority opinion taken as valid in cases when there was disagreement. Words were annotated in the IOB2 format, which determines whether they are included in the beginning, middle, or end of the entity name. This dataset is notable for very fine granulation with a lot of unique classes related to Islamic tradition (i.e., Allah, Prophet, Clan) in addition to standard NER classes such as person, time, or organization. For this reason, the dataset provides a strong foundation for testing Arabic language AI tools with sophisticated NLP capacities.

5.2.4 Our Arabic Legal Content (ALC) Dataset

This dataset was created inside the research department at Elm company. This dataset was extracted from 3648 Arabic legal case documents, which have been collected form Board of Grievances (BoG). It contains more than 345K tokens including almost 6K distinct entities of four classes in the initial stage of our project.
The used NER classes are
PER such as

الملك فيصل ، الامام بن باز

, LOC such as

الحرم المكي الشريف ، المنطقة الجنوبية

, ORG such as

بوزارة العمل ، المحكمة الإدارية بجدة

and JOB such as

مدير شرطة محافظة ، رئيس بلدية

5.3 Fine-Tuning Process

Our proposed model was trained and tested on all datasets mentioned in Sect. 5.2, with the idea of fine-tuning the hyperparameters in every iteration. Hyperparameters are set up to create the best possible conditions for recognizing named entities in a labeled dataset based on the words found in the input sequence as shown in Table 2. When the dataset is used for model training and testing, individual sentences from unseen content are selected at random and fed into the model as input. 80% of the dataset is typically used for training, 10% as a validation set, and the remaining 10% to test the performance of the model on unseen examples.

All the datasets are very useful in model evaluation due to their versatility and the relevance of the content, making it a logical choice to include in this study. We have fine-tuned three pretrained models, including AraBERT, AraElectra, and XLM-Roberta. In the first experiment, we fine-tuned AraBert V2, the large model containing 24 layers of encoders stacked on top of one another, 16 self-attention heads, and a hidden size of 1024. In the second experiment, we used only the discriminator base model for AraElactra. This model has 12 attention heads, 12 hidden layers, and 768 hidden states size. In the third experiment, we used XLM-Roberta, a model with 12 attention heads, 12 hidden layers, and 768 hidden states.

During the three experiments, we used an AdamW optimizer, which is useful when fine-tuning a pretrained model as frozen layers. In all experiments, the learning ratio was $5e - 5$, and the number of epochs was 5. The model input is

Table 2 Table of hyperparameters for training the proposed Arabic NER model

Parameter	AraBERT	AraElectra	XLM-Roberta
Max sequence length	[128,256,512]	[128,256,512]	[128,256,512]
No. heads	16	12	12
No. hidden layers	24	12	12
Hidden layer size	1024	768	768
Batch size	4	16	4
Vocab size	64000	64000	250002
loss	Crf loss	Crf loss	Crf loss
Dropout prob	0.1	0.1	0.1
Optimizer	AdamW	AdamW	AdamW
Learning rate	5e−5	5e−5	5e−5
Number of epochs	5	5	5

a sequence of tokens that are processed to two vectors, input IDs, and attention masks using the BERT tokenizer. The tested sequence lengths were 128, 256, and 512 tokens. Dropout optimization was applied where a dropout step was introduced immediately before the data reach the linear layer. We used the same dropout ratio of 0.1.

6 Results

In Table 3, the output of the proposed model is compared with state-of-the-art NER models for the Arabic language. Since the proposed solution does not use any sources other than the training sample, the performance of the competing methods was presented without access to such sources for the sake of fair evaluation. The AraBertv2 + CRF model achieved the most impressive result; the F1-macro score of almost 89.6% for this model concatenates the last 6 hidden layers. However, this model achieved these results on a sequence length of 256 tokens. Therefore, we applied the same model on a 512 token length, and we found that the best result is attained when we sum the 11 hidden layers; the F1-macro has not significantly change. In general, all the fine-tuned models outperform the state-of-the-art models, with significant improvements by almost 5% more than the best one, as we can see in Table 3. Tables 4 and 5 show the experimental results of our proposed model with respect to the AQMAR dataset, using AraBert and AraElectra models, respectively. Besides ANERCorp and AQMAR, we also tested the model on CANERCorpus data that are MSA data taken from the books of Hadith and Islamic history described in Sect. 5.2.3. The results were as in Table 6 showing the superiority of our model

Table 3 The performance of the proposed Arabic NER model vs state of the art on ANERCorp. Bold values are the highest F1 macro that the model achieved

Model	Seq	F1-Macro	Accuracy	Precision	Recall	F1-measure
AraBertv1	512	0.82767	0.971329	0.83902	0.816630	0.842
mBERT	512	0.76721	0.96293	0.79244	0.74354	0.784
gigaBERT	512	NA	0.969889	0.82820	0.812253	0.82015
AraBertv2	512	0.8043	0.97004	0.82900	0.81050	0.81965
Mawdoo3	512	NA	0.964331	0.79183	0.755798	0.77339
MARBERT	512	NA	0.966730	0.81267	0.774617	0.79318
ARBERT	512	NA	0.97224	0.84656	0.82582	0.83606
AraBertv2 + CRF (last)	512	0.88888	0.9916173	0.905367	0.912455	0.90889
AraBertv2 + CRF (concat 6)	256	**0.8956**	**0.9919526**	**0.913135**	**0.91378**	**0.91345**
AraBertv2 + CRF (concat 9)	512	0.8933	0.9916918	0.910310	0.913536	0.91192
AraBertv2 + CRF (sum 11)	512	**0.8946**	**0.9917663**	**0.908192**	**0.915954**	**0.91205**
AraElectra +CRF (concat 5)	512	0.872115	0.98755	0.9104595	0.894163	0.9022379
XLM-Roberta +CRF (concat 9)	256	0.875697	0.98968	0.90608	0.90181	0.90181

Table 4 The performance of the proposed Arabic NER model using bert-large-arabertv2+crf test results on AQMAR dataset

Model	#state	Seq	F1-Macro	Accuracy	Precision	Recall	F1-measure
last	1	128	0.862648	0.984333	0.8567	0.879594	0.8680425
concat	12	128	0.88176	0.985708	0.87530	0.895202	0.8851435
sum	8	128	0.88222	0.984944	0.864197	0.88832	0.876095
last	1	256	0.852139	0.983815	0.848448	0.878862	0.8633879
concat	6	256	0.875361	0.985300	0.866348	0.888616	0.8773413
sum	9	256	0.8772	0.98478	0.85322	0.8982412	0.875
last	1	512	0.86588	0.984261	0.8651	0.8809234	0.8729
concat	10	512	0.87350	0.98500	0.87350	0.8766467	0.875
sum	11	512	0.87789	0.985374	0.871121	0.8902439	0.88

Table 5 The performance of the proposed Arabic NER model using araelectra-base-discriminator +crf test results on AQMAR dataset

Model	#states	Seq	F1-Macro	Accuracy	Precision	Recall	F1-measure
last	1	128	0.869587	0.978834	0.875871	0.86620689	0.871012482
concat	10	128	0.865022	0.97758	0.86192	0.8583333	0.860125261
sum	2	128	0.859811	0.978417	0.853556	0.8547486	0.854152128
last	1	256	0.862752	0.978032	0.864902	0.8758815	0.870357393
concat	2	256	0.875771	0.978344	0.87883	0.87638888	0.877607788
sum	3	256	0.861978	0.978552	0.864902	0.8601108	0.8625
Last	1	512	0.872761	0.979593	0.862116	0.8792613	0.870604781
concat	10	512	0.862209	0.977719	0.86908	0.85714285	0.863070539
Sum	2	512	0.8559	0.977719	0.866295	0.859116	0.862690707

Table 6 The performance of the proposed Arabic NER model using bert-large-arabertv2+crf test results on CANERCorpus dataset. Bold values are the highest F1 macro that the model achieved

Model	Seq	F1-Macro	Precision	Recall	F1-measure
[7]	54	NA	94.10	95.54	94.76
AraBertv2 + CRF (concat 6)	54	**91**	**98.11**	**98.42**	**98.26**

Table 7 The performance of the proposed Arabic NER model using bert-large-arabertv2+crf test results on our Arabic legal dataset. Bold values are the highest F1 macro that the model achieved

Model	Seq	F1-Macro	Precision	Recall	F1-measure
xlm-roberta-large	256	0.824	0.874	0.871	0.872
araelectra-base-discriminator	256	0.762	0.784	0.852	0.815
AraBertv2 + CRF (concat last 6)	256	**0.851**	**0.9**	**0.894**	**0.897**

over the model mentioned in [7] with the same hyperparameter settings. Finally, we tested our model on customized data for Arabic legal content and found that the model achieved better results than the previous models and an increase in F1-macro by 3% over the best model as shown in Table 7.

Thus, it is fair to say that stack architecture of the transformer-based models in combination with conditional random field currently represents one of the most successful NER methodologies that are independent from external sources. Its surprising performance can be explained by the strength of contextualization unique for this approach, which allows it to be accurate even when only limited information is available.

7 Conclusion

In some studies, scalability and versatility of the NER solution were considered alongside accuracy, reflecting the objective to create tools that could be used in practice without too many limitations. In this study, a simple architectural design of transformer-based model was presented, created specifically for tagging of word sequences in the context of Named Entity Recognition. This solution outperforms most of the state-of-the-art methods for this task, including those that rely on knowledge bases. A crucial element of the proposed solutions is their ability to track interdependencies between labels. This can be done with a CRF layer. Also, the process of summing and concatenating vectors has been shown to be effective in generating additional information that helps to improve the tagging accuracy. Since vector representations are made on the words level, the model is able to collect contextual clues related to both syntax and morphology. The work in the future will be in two parts as follows. The first is to work on disambiguating the label when there is a word that expresses two different entities, such as the name of a place and a person at the same time. The second is to increase the number of named entity classes as well as working on our own dataset.

Acknowledgments This work was supported by the Research Department in Elm Company under the Arabic language processing initiative.

References

1. Abdelali, A., Darwish, K., Durrani, N., Mubarak, H.: Farasa: A fast and furious segmenter for Arabic. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 11–16 (2016)
2. AbdelRahman, S., Elarnaoty, M., Magdy, M., Fahmy, A.: Integrated machine learning techniques for Arabic named entity recognition. IJCSI **7**(4), 27–36 (2010)
3. Abdul-Mageed, M., Elmadany, A., Nagoudi, E.M.B.: Arbert & MARBERT: deep bidirectional transformers for Arabic. Preprint (2020). arXiv:2101.01785
4. Aboaoga, M., Ab Aziz, M.J.: Arabic person names recognition by using a rule based approach. J. Comput. Sci. **9**(7), 922 (2013)
5. Al-Qurishi, M.S., Souissi, R.: Arabic named entity recognition using transformer-based-crf model. In: Proceedings of The Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021), pp. 262–271 (2021)

6. Alkhatib, M., Shaalan, K.: Boosting Arabic named entity recognition transliteration with deep learning. In: The Thirty-Third International Flairs Conference (2020)
7. Alsaaran, N., Alrabiah, M.: Classical Arabic named entity recognition using variant deep neural network architectures and BERT. IEEE Access **9**, 91537–91547 (2021)
8. Antoun, W., Baly, F., Hajj, H.: Arabert: Transformer-based model for Arabic language understanding. In: Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pp. 9–15 (2020)
9. Antoun, W., Baly, F., Hajj, H.: Araelectra: Pre-training text discriminators for Arabic language understanding. In: Proceedings of the Sixth Arabic Natural Language Processing Workshop, pp. 191–195 (2021)
10. Benajiba, Y., Rosso, P.: Arabic named entity recognition using conditional random fields. In: Proc. of Workshop on HLT & NLP within the Arabic World, LREC, vol. 8, pp. 143–153. Citeseer (2008)
11. Benajiba, Y., Rosso, P., Benedíruiz, J.M.: ANERsys: An Arabic named entity recognition system based on maximum entropy. In: International Conference on Intelligent Text Processing and Computational Linguistics, pp. 143–153. Springer (2007)
12. Benajiba, Y., Zitouni, I., Diab, M., Rosso, P.: Arabic named entity recognition: using features extracted from noisy data. In: Proceedings of the ACL 2010 Conference Short Papers, pp. 281–285 (2010)
13. Boudjellal, N., Zhang, H., Khan, A., Ahmad, A., Naseem, R., Shang, J., Dai, L.: Abioner: a BERT-based model for Arabic biomedical named-entity recognition. Complexity **2021**, Article ID 6633213 (2021). https://doi.org/10.1155/2021/6633213
14. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. Preprint (2020). arXiv:2003.10555
15. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. Preprint (2019). arXiv:1911.02116
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. Preprint (2018). arXiv:1810.04805
17. Elsebai, A., Meziane, F.: Extracting person names from Arabic newspapers. In: 2011 International Conference on Innovations in Information Technology, pp. 87–89. IEEE (2011)
18. Farber, B., Freitag, D., Habash, N., Rambow, O.: Improving NER in Arabic using a morphological tagger. In: LREC (2008)
19. Goyal, A., Gupta, V., Kumar, M.: Recent named entity recognition and classification techniques: a systematic review. Comput. Sci. Rev. **29**, 21–43 (2018)
20. Gridach, M.: Character-aware neural networks for Arabic named entity recognition for social media. In: Proceedings of the 6th workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016), pp. 23–32 (2016)
21. Gridach, M.: Deep learning approach for Arabic named entity recognition. In: International Conference on Intelligent Text Processing and Computational Linguistics, pp. 439–451. Springer (2016)
22. Helwe, C., Dib, G., Shamas, M., Elbassuoni, S.: A semi-supervised BERT approach for Arabic named entity recognition. In: Proceedings of the Fifth Arabic Natural Language Processing Workshop, pp. 49–57 (2020)
23. Helwe, C., Elbassuoni, S.: Arabic named entity recognition via deep co-learning. Artif. Intell. Rev. **52**(1), 197–215 (2019)
24. Khalil, H., Osman, T., Miltan, M.: Extracting Arabic composite names using genitive principles of Arabic grammar. ACM Trans. Asian Low Resource Lang. Inf. Process. (TALLIP) **19**(4), 1–16 (2020)
25. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
26. Larochelle, H.: Lectures on conditional random fields (2021)
27. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Trans. Knowl. Data Eng. (2020)

28. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized BERT pretraining approach. Preprint (2019). arXiv:1907.11692
29. Maloney, J., Niv, M.: TAGARAB: a fast, accurate Arabic name recognizer using high-precision morphological analysis. In: Computational Approaches to Semitic Languages (1998)
30. Mohammed, N.F., Omar, N.: Arabic named entity recognition using artificial neural network. J. Comput. Sci. **8**(8), 1285 (2012)
31. Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., Inoue, G., Eryani, F., Erdmann, A., Habash, N.: Camel tools: An open source Python toolkit for Arabic natural language processing. In: Proceedings of the 12th Language Resources and Evaluation Conference, pp. 7022–7032 (2020)
32. Oudah, M., Shaalan, K.: A pipeline Arabic named entity recognition using a hybrid approach. In: Proceedings of COLING 2012, pp. 2159–2176 (2012)
33. Oudah, M., Shaalan, K.: Person name recognition using the hybrid approach. In: International Conference on Application of Natural Language to Information Systems, pp. 237–248. Springer (2013)
34. Pasha, A., Al-Badrashiny, M., Diab, M.T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R.: MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In: Lrec. vol. 14, pp. 1094–1101. Citeseer (2014)
35. Safaya, A., Abdullatif, M., Yuret, D.: KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 2054–2059 (2020)
36. Salah, R.E., Zakaria, L.Q.B.: Building the classical Arabic named entity recognition corpus (CANERCorpus). In: 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP). pp. 1–8. IEEE (2018)
37. Schneider, N., Mohit, B., Oflazer, K., Smith, N.A.: Coarse lexical semantic annotation with supersenses: an Arabic case study. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 253–258 (2012)
38. Shaalan, K.: A survey of Arabic named entity recognition and classification. Computational Linguistics **40**(2), 469–510 (2014)
39. Shaalan, K., Raza, H.: NERA: Named entity recognition for Arabic. J. Am. Soc. Inf. Sci. Tech. **60**(8), 1652–1663 (2009)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural Information Processing Systems, pp. 5998–6008 (2017)
41. Yucel, O., Sen, S.: Language independent recommender agent. Knowl. Eng. Rev. **33**, e15 (2018)
42. Zaghouani, W.: RENAR: A rule-based Arabic named entity recognition system. ACM Trans. Asian Lang. Inf. Process. (TALIP) **11**(1), 1–13 (2012)

Static Fuzzy Bag-of-Words: Exploring Static Universe Matrices for Sentence Embeddings

Matteo Muffo, Roberto Tedesco, Licia Sbattella, and Vincenzo Scotti

Abstract Vector semantics has slightly become a key tool for natural language processing, especially concerning text analysis. This kind of vector representation is usually encoded through embeddings that can be used to encode semantic information at different levels of granularity. In fact, through the years, not only models for word embeddings have been developed but also for sentence and documents. With this work, we address *sentence embeddings*, in particular the *non-parametric* ones, which offer a good trade-off between performance and inference speed. We present *Static Fuzzy Bag-of-Word* (SFBoW) model, a refinement of the Fuzzy Bag-of-Words approach yielding fixed-dimension sentence embeddings. We targeted fixed-size embeddings to promote caching a re-usability, speeding the inference of a system that relies on our model. In this paper, we explore various approaches for the construction of a *static universe matrix*, fundamental to make the sentence embeddings of fixed size. To show the validity of our approach, we benchmarked our model on a semantic similarity task, obtaining competitive performances.

1 Introduction

The advent of machine and deep learning-based models has influenced many areas of the artificial intelligence field [10, 17], including natural language processing (NLP). To enable the deep neural network models, which strongly rely on matrix multiplication operations, process data from NLP, vector semantics has played a

M. Muffo (✉)
Indigo.ai, Milano, Italy
e-mail: matteo@indigo.ai

R. Tedesco · L. Sbattella · V. Scotti
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milano, Italy
e-mail: roberto.tedesco@polimi.it; licia.sbattella@polimi.it; vincenzo.scotti@polimi.it

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1_10

crucial role. In fact, vector semantics rely on the concept that everything can be represented as real-valued vectors (or points) in a hyperspace. Moreover, according to vector semantics, the position of the object in the hyperspace represents its meaning.

In the case of NLP, words with similar meaning should be represented close in the hyperspace, and, analogously, words with different meaning should be far one from the other. This approach to word vector representation is called word embedding. These embeddings are computed through *self-supervised* representation learning [9]. There are many different models to extract these embeddings, at different levels of granularity, which have consistently taken the role of input representation for many NLP tasks [19].

In the last decade, the approach shifted from *shallow and static* word representations [11, 22, 25] towards *deep and contextual* ones [15, 26, 28], pushing forward incredibly the state of the art on NLP. However, for certain problems like web search and question answering, word-level representations are not sufficient. For this reason high-level models such as those for *sentence embedding* have been created [40]. These high-level models can be powered through either satic or contextual representations.

Shallow *word embedding* models immediately provided noticeable results [14]. Such representations were quickly adopted to provide an input for syntactic analysis: they helped improve results in part-of-speech (POS) tagging, named entity recognition (NER) and semantic role labelling (SRL). Shortly after, they were employed in more complex problems like language modelling, machine translation [36] and dialogue systems [35]. Although impressive, the results of these models were limited by the inability to model properly the context surrounding each word in the input sequence.

Neural language models (LMs) implemented through *transformer networks* [18, 37], on the other side, played a significant role for deep contextual representations. The hidden representations extracted through these huge models, trained on massive collections of unlabelled textual data, boosted the performances in many NLP tasks [29, 30, 38, 39] The trade-off with respect to shallow model is indeed in the amount of computational resources: both in terms of time and memory. This resource demand is especially high at train time.

In this vector semantics settings, with focus on the sentence embeddings, we present our Static Fuzzy Bag-of-Words (SFBoW) model. It's a model for non-parametric sentence embeddings based on the DynaMax Fuzzy Bag-of-Words model [42]. In particular, with this paper, we explore approaches to build the universe matrix, core component of the Fuzzy Bag-of-Words solutions, to be static. This model is designed to promote caching (in the sense of re-usability of the embeddings), short analysis time and valid performances; thus, it is advised for applications with limited resources or with power consumption issues, like embedded systems. To evaluate the goodness of the proposed universe matrices, we relied on the semantic textual similarity (STS) benchmark.

We organise the remainder of this paper into the following sections: in Sect. 2, we summarise the main concepts related to learnt word and sentence representations; in

Sect. 3, we introduce SFBoW, our model; in Sects. 4 and 5, we present, respectively, the evaluation approach we followed to evaluate SFBoW and the results of such evaluation; and, finally, in Sect. 6, we summarise the presented work and we present the expected future works.

2 Related Work

Our work revolves around the concept of *vector semantics*: the idea that the meaning of a word or a sentence can be modelled as a vector [23].

The first steps on this subject were made in information retrieval (IR) context with the vector space model [33], where documents and queries were represented as high-dimensional (vocabulary size) sparse embedding vectors. In this model, each dimension is used to represent a word, so that given a vocabulary $\mathcal{V}$:

- A word $w_i \in \mathcal{V}$, with $i \in [1, |\mathcal{V}|] \subseteq \mathbb{N}$, is expressed as a so-called "one-hot" binary vector $\mathbf{v}_{w_i} \in \mathbb{1}^{|\mathcal{V}|}$, where, calling $v_{w_i,j}$ the jth element of the word vector, it holds that $v_{w_i,j} = 1 \iff j = i$.
- A sentence S is expressed as vector $\boldsymbol{\mu}_S \in \mathbb{N}^{|\mathcal{V}|}$, where $\mu_{S,i}$, the ith element of vector $\boldsymbol{\mu}_S$, namely, $c_{S,i}$, represents the number of times word w_i appears in sentence S.

The resulting sentence representation, used also for text documents, is called *Bag-of-Words* (BoW) and can be summarised as

$$\boldsymbol{\mu}_S = \sum_{i=1}^{|\mathcal{V}|} c_{S,i} \cdot \mathbf{v}_{w_i}. \tag{1}$$

These representation models needed to be replaced because of the sparsity, which made them resource consuming, and the induced orthogonality among vectors with similar meanings.

2.1 Word and Sentence Embeddings

Word embeddings refer to the dense semantic vector representation of words; such representation can be divided into *prediction-based* and *count-based* [8].

The former group identifies the embeddings obtained through the training of models for next/missing word prediction given a context. It encompasses models like *Word2Vec* [21, 22] and *fastText* [11]. The latter group refers to the embeddings obtained leveraging word co-occurrence counts in a corpus. One of the most recent solutions of this group is *GloVe* [25].

All the models mentioned above belong to the class of *shallow* models, where the embedding of a word w_i can be extracted through lookup over the rows of the embedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$, with d being the desired dimensionality of the embedding space. Given the word (column) vector $\mathbf{v}_{w_i}$, the corresponding word embedding $\mathbf{u}_{w_i} \in \mathbb{R}^d$ can be computed as (see Sect. 2.2)

$$\mathbf{u}_{w_i} = \mathbf{W}^\top \cdot \mathbf{v}_{w_i}. \tag{2}$$

More recently, the introduction of transformer-based LMs [18], like *BERT* [15], *GPT* [12, 26, 27] or *T5* [28], has spread the concept of *contextual embeddings*; such embeddings proved to be particularly helpful for a wide variety of NLP problems, as shown by the leader boards of NLP benchmarks [29, 30, 38, 39].

The inherent hierarchical structure of the human language makes it hard to understand a text from single words; thus, the birth of higher-level semantic representations for sentences, which are the sentence embeddings, was just a natural consequence. As for the word embeddings, also sentence embeddings are organised into two groups, *parametrised* and *non-parametrised*, depending on whether the model requires parameter training or not.

Clear examples of parametric model are the *skip-thoughts vectors* [16] and *Sent2Vec* [24], which generalises Word2Vec. Non-parametric models, instead, show that simply aggregating the information from pre-trained word embeddings, for example, through averaging, as in *SIF weighting* [6], is sufficient to represent higher-level entities like sentences and paragraphs.

Transformer LMs are also usable at sentence level. An example is the parametric model *Sentence-BERT* [31], obtained by fine-tuning on natural language inference corpora.

All these models rely on the assumption that cosine similarity is the correct metric to compute “meaning distance” between sentences. This is why parametric models are explicitly trained to minimise this measure for similar sentences and maximise it for dissimilar sentences.

However, cosine similarity may not be the only and best measure. The *DynaMax* model [42] proposed to follow a fuzzy set representation of sentences and to rely on fuzzy Jaccard similarity instead of the cosine one. As a result, the DynaMax model outperformed many non-parametric models and performed comparably to parametric ones under cosine similarity measurements, even if competitors were trained directly to optimise that metric, while the DynaMax approach was utterly unrelated to that objective.

The use of fuzzy sets to represent documents is not new, and it was already proposed by [41]. With respect to DynaMax, previous results were inferior because of their approach to compute fuzzy membership.

2.2 *Fuzzy Bag-of-Words and DynaMax for Sentence Embeddings*

The *Fuzzy Bag-of-Words* (FBoW) model for text representation [41]—and its generalised and improved variant DynaMax [42], which introduced a better similarity metric—represents the starting point of our work, which is described in Sect. 3.

The BoW approach, described at the beginning of Sect. 2, can be seen as a multiset representation of text. It enables to measure similarity between two sentences with set similarity measures, like Jaccard, Otsuka and Dice indexes. These indexes share a common pattern to measure the similarity σ between two sets A and B [42]:

$$\sigma\left(A, B\right) = n_{shared}\left(A, B\right)/n_{total}\left(A, B\right) \tag{3}$$

where $n_{shared}\left(A, B\right)$ denotes the count of shared elements and $n_{total}\left(A, B\right)$ is the count of total elements. In particular, the Jaccard index is defined as

$$\sigma_{Jaccard}\left(A, B\right) = \left|A \cap B\right| / \left|A \cup B\right|. \tag{4}$$

However, the simple set similarity is a rigid approach as it allows for some degree of similarity when the very same words appear in both sentences, but fails in the presence of synonyms. This is where *fuzzy sets theory* comes handy: in fact, fuzzy sets enable to interpret each word in $\mathcal{V}$ as a singleton and measure the degree of membership of any word to this singleton as the similarity between the two considered words [41].

The FBoW model prescribes to work in this way [41]:

- Each word w_i is interpreted as a singleton $\{w_i\}$; thus, the membership degree of any word w_j in the vocabulary (with $j \in [1, |\mathcal{V}|] \subseteq \mathbb{N}$) with respect to this set is computed as the similarity σ between w_i and w_j. These similarities can be used to fill a $|\mathcal{V}|$-sized vector $\hat{\mathbf{v}}_{w_i}$ used to provide the fuzzy representation of w_i (the jth element $\hat{\mathbf{v}}_{w_i,j}$ being $\sigma\left(w_i, w_j\right)$).
- A sentence S is simply defined through the fuzzy union operator, which is determined by the max operator over the membership degrees. In this case, the S is represented by a vector of $|\mathcal{V}|$ elements.

The generalised FBoW approach [42] prescribes to compute the *fuzzy embedding* of a word singleton as

$$\hat{\mathbf{v}}_{w_i} = \mathbf{U} \cdot \mathbf{u}_{w_i} = \mathbf{U} \cdot \mathbf{W}^{\top} \cdot \mathbf{v}_{w_i} \tag{5}$$

to reduce the dimension of the output vector for S, where $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a word embedding matrix (defined as in Sect. 2.1), $\mathbf{u}_{w_i}$ is defined in Eq. (2) and $\mathbf{U} \in \mathbb{R}^{u \times d}$ (with u being the desired dimension of the fuzzy embeddings) is the *universe matrix*, derived from the *universe set* U, which is defined as "the set of all possible terms that occur in a certain domain". The generalised FBoW produces vectors of u elements, where $u = |U|$.

Given the fuzzy embeddings of the words in a sentence S, the generalised FBoW representation of S is a vector $\hat{\boldsymbol{\mu}}_S$ whose jth element $\hat{\mu}_{S,j}$ (with $j \in [1, u] \subseteq \mathbb{N}$) can be computed as

$$\hat{\mu}_{S,j} = \max_{w_i \in S} c_{S,i} \cdot \hat{v}_{w_i,j} \tag{6}$$

where $c_{S,i}$ and $\hat{v}_{w_i,j}$ are, respectively, the number of occurrences of word w_i in sentence S and the jth element of the $\hat{\mathbf{v}}_{w_i}$ vector.

The universe set can be defined in different ways, and the same applies for the universe matrix [42]. Among the possible solutions, the DynaMax algorithm for fuzzy sentence embeddings builds the universe matrix from the word embedding matrix, stacking solely the embedding vectors of the words appearing in the sentences to be compared.

Notice that in this way the resulting universe matrix is not unique, and as a consequence, neither are the embeddings. This condition can be noticed from the description of the algorithm and from the definition of the universe matrix: when comparing two sentences S_a and S_b, the universe set U used in their comparison is $U \equiv S_a \cup S_b$, so the resulting sentence embeddings have size $u = |U| = |S_a \cup S_b|$. In fact, the universe matrix is given by

$$\mathbf{U} = \left[\mathbf{u}_{w_i} \forall w_i \in U\right]^\top . \tag{7}$$

This characteristic is unfortunate as, for example, in IR, it requires a complete re-encoding of the entire document achieved for each query.

The real improvement of DynaMax is in the introduction of the fuzzy Jaccard index to compute the semantic similarity between two sentences S_a and S_b, rather than the generalisation of the FBoW, which replaced the original use of the cosine similarity [41]:

$$\hat{\sigma}_{Jaccard}\left(\hat{\boldsymbol{\mu}}_{S_a}, \hat{\boldsymbol{\mu}}_{S_b}\right) = \frac{\sum_{i=1}^{u} \min\left(\hat{\mu}_{S_a,i}, \hat{\mu}_{S_b,i}\right)}{\sum_{i=1}^{u} \max\left(\hat{\mu}_{S_a,i}, \hat{\mu}_{S_b,i}\right)}. \tag{8}$$

3 Static Fuzzy Bag-of-Words Model

Starting from the DynaMax, which evolved from the FBoW model, we developed our follow-up aimed at providing a unique matrix $\mathbf{U}$ and thus embeddings with a fixed dimension. In Fig. 1 is represented the visualisation of our approach.

3.1 Word Embeddings

Word embeddings play a central role in our algorithm as they also provide the starting point of the construction of the universe matrix. For this work, we leveraged pre-trained shallow models (more details in Sect. 4.1) for two main reasons:

- The model is encoded in a matrix where each row corresponds to a word.
- We want to provide a sentence embedding approach that does not require training, easing its accessibility.

The vocabulary of these models, composed starting from all the tokens in the training corpora, is usually more extensive than the English vocabulary, as it contains named entities, incorrectly spelt words, non-existing words, URLs, email addresses and similar. To reduce the computational effort needed to construct and use the universe matrix, we have considered some subsets of the employed word embedding model's vocabulary.

Depending on the experiment, we work with either the 100,000 most frequently used terms, the 50,000 most frequently used terms (term frequencies are given by the corpora used to train the word embedding model) or the subset composed of all the spell-checked terms present in a reference English dictionary (obtained through the Aspell English spell-checker[1]).

In the following sections, the $\check{\mathbf{W}}$ symbol refers to these as *reduced* word embedding matrices/models.

3.2 Universe Matrix

During the experiments, we tried four main approaches to build the universe matrix **U**: the first two – proposed, but not explored, by the original authors of DynaMax [42] – consist, respectively, in the usage of a clustered embedding matrix and an identity matrix with the rank equal to the size of the word embeddings. Instead,

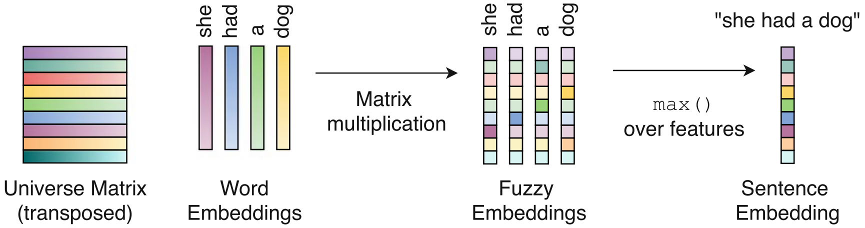


Fig. 1 Visualisation of the sentence embedding computation process using SFBoW

[1] http://aspell.net.

the third approach consists of applying multivariate analysis techniques to the word embedding matrix to build the universe one. The last approach considers the norm of the word vectors to filter out less significant words for the representation.

In the following formulae, we refer to d as the dimensionality of the word embedding vectors, while the SFBoW embedding of the singleton of word w_i is represented as $\check{\mathbf{v}}_{w_i}$. Clustering and multivariate analysis can be applied to the whole embedding vocabulary or the subsets of the vocabulary introduced in Sect. 3.1. Apart from reducing the computational time, we did so to see if these subsets are sufficient to provide a helpful representation.

3.2.1 Clustering

The idea is to group the embedding vectors into clusters and use their centroids; in this way, the fuzzy membership will be computed over the clusters—which are expected to host semantically similar words—instead of all the word singletons. The universe set is thus built out of abstract entities only, which are the centroids. Considering k centroids the k-dimensional embedding $\check{\mathbf{v}}_{w_i}$ of the singleton of word w_i is

$$\check{\mathbf{v}}_{w_i} = \mathbf{K}^\top \cdot \mathbf{u}_{w_i} = \left[\mathbf{k}_1, \ldots, \mathbf{k}_k\right]^\top \cdot \mathbf{u}_{w_i} = \mathbf{K}^\top \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{9}$$

where $\mathbf{k}_j$, the jth (with $j \in [1, k] \subseteq \mathbb{N}$) column of $\mathbf{K}$, corresponds to the centroid of the jth cluster. This approach generates k-dimensional word and sentence embeddings.

3.2.2 Identity

Alternatively, instead of looking for a group of semantically similar words that may form a significant group, useful for semantic similarity, we consider the possibility of re-using the word embedding dimensions (features) to represent the semantic content of a sentence. So, we just use the identity matrix as the universe, $\mathbf{U} = \mathbf{I} \in \mathbb{R}^{d \times d}$, so that $\check{\mathbf{v}}_{w_i} \in \mathbb{R}^d$ is

$$\check{\mathbf{v}}_{w_i} = \mathbf{I} \cdot \mathbf{u}_{w_i} = \mathbf{I} \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{10}$$

where this approach generates d-dimensional word embeddings and sentence embeddings.

3.2.3 Multivariate Analysis

The same idea moves our multivariate analysis proposal. Judging by previous results, word embeddings aggregated correctly might be sufficient to provide a semantically valid representation of a sentence.

What can bring better results might be as simple as roto-translate the reference system of the embedding representation. In this sense, we propose to use to compute the fuzzy membership, and hence the fuzzy Jaccard similarity index, over these dimensions resulting from roto-translation, expecting that this "new perspective" will expose better the semantic content. So, defining $\mathbf{U} = \mathbf{M}$, where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is the transformation matrix, we have that $\check{\mathbf{v}}_{w_i} \in \mathbb{R}^d$ is

$$\check{\mathbf{v}}_{w_i} = \mathbf{M} \cdot \mathbf{u}_{w_i} = \mathbf{M} \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{11}$$

thus yielding d-dimensional word and sentence embeddings.

3.2.4 Vector Significance

Early analysis of shallow word embedding models showed that word vectors providing stronger semantic representation have a higher norm [34]. Moreover, when comparing the norm of the vectors with their term frequency within the training corpus, it is possible to notice that highly frequent terms, as well as rare one, have considerably smaller norm.

This concept is not anew. In fact, in the *term frequency-inverse document frequency* (TF-IDF) approach for document representation, rare words, as well as highly frequent words, should give little if any contribution to the meaning representation [7, 20]. For similar reasons, in data mining and retrieval settings, *stop words*, which are the highly frequent words in a corpus, are discarded from the document analysis.

We propose to leverage the word embeddings with a significance level above a certain (custom) threshold to build the universe matrix, to retain only the most relevant vectors. Defining $\mathbf{U} = \mathbf{L}^\top$, where $\mathbf{L} \in \mathbb{R}^{d \times d}$ is the matrix whose columns are the first n word vectors in decreasing Euclidean norm $\|\mathbf{u}_{w_i}\|_2$ order, we have that $\check{\mathbf{v}}_{w_i} \in \mathbb{R}^d$ is

$$\check{\mathbf{v}}_{w_i} = \mathbf{L}^\top \cdot \mathbf{u}_{w_i} = \left[\ldots, \mathbf{u}_{w_j}, \ldots\right]^\top \cdot \mathbf{u}_{w_i} = \mathbf{L}^\top \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{12}$$

where the resulting sentence embeddings have as many dimensions as the number n of retained word vectors.

4 Experiments

In order to find the best solution in terms of word embedding matrix and universe matrix, we explored various possibilities. Then, to measure the goodness of our sentence embeddings, we leveraged a series of STS tasks and compared the results with the preceding models.

4.1 Word Embeddings

For what concerns the word embeddings, we have decided to work with a selection of four models:

- Word2Vec, with 300-dimensional embeddings
- GloVe, with 300-dimensional embeddings
- fastText, with 300-dimensional embeddings
- Sent2Vec, with 700-dimensional embeddings

As shown by the word embedding models list, we are also employing a Sent2Vec sentence embedding model. The embedding matrix of this model can be used for word embeddings too. During the experiments, we focused on the universe matrix construction. For this reason, we relied on pre-trained models for word embeddings, available on the web.

4.2 Universe Matrices

The universe matrices we considered are divided into four buckets, as described in Sect. 3.2.

4.2.1 Clustering

Universe matrices built using clustering leverage four different algorithms: k-means, spherical k-means, DBSCAN and HDBSCAN.

We selected k-means and spherical k-means because they usually lead to good results; the latter was specifically designed for textual purposes, with low demand in time and computational resources. For all algorithms, we considered the same values for k (the number of centroids), which were 100, 1000, 10,000 and 25,000.

For all the values of k, we performed clustering on different subsets of the vocabulary: k-means was applied on the whole English vocabulary as well as to the top 100,000 frequently used words subset, while spherical k-means was applied to the subset of the first 50,000 frequently used words (to reduce computational time).

We also explored density-based algorithms (DBSCAN and HDBSCAN), which do not require defining in advance the number of clusters, using Euclidean and cosine distance between the word embedding.

For what concerns DBSCAN with Euclidean distance, we varied the radius of the neighbourhood ε between 3 and 8 and worked over the same two subsets considered for k-means, while the cosine distance ε was between 0.1 and 0.55, and it was applied over the subset of the first 50, 000 frequently used words (for computational reasons, as we did for spherical k-means). Concerning HDBSCAN, we varied the smallest size grouping of clusters in the set $\{2, 4, 30, 50, 100\}$ and the minimum neighbourhood size of core samples in the set $\{1, 2, 5, 10, 50\}$. We considered this latter density-based algorithm since basic DBSCAN happens to fail with high-dimensional data.

4.2.2 Identity

This approach consists of using the identity matrix as the universe, and in this way, the singletons we use to compute the fuzzy membership are the dimensions of the word embeddings, which corresponds to the learnt features. This is the most lightweight method as it just requires to compute the word embeddings of a sentence and then the fuzzy membership over the exact d dimensions.

4.2.3 Multivariate Analysis

We adopted the principal component analysis (PCA) to get a rotation matrix to serve as a universe matrix to the SFBoW. In fact, through PCA, the d-dimensional word embedding vectors are decomposed along with the d orthogonal directions of their variance. These components are then reordered to decrease explained variance and represent our fuzzy semantic sets.

The principal component of the reduced word embedding matrix $\check{\mathbf{W}}$ is described by the matrix $\mathbf{T} = \mathbf{P}^\top \cdot \check{\mathbf{W}}$, where $\mathbf{P}$ is a $d \times d$ matrix whose columns are the eigenvectors of the matrix $\check{\mathbf{W}}^\top \cdot \check{\mathbf{W}}$. With our approach, the matrix $\mathbf{P}^\top$, sometimes called the *whitening* or *sphering transformation matrix*, serves as universe matrix $\mathbf{U}$. In this way, the SFBoW embedding of a word singleton becomes

$$\check{\mathbf{v}}_{w_i} = \mathbf{P}^\top \cdot \mathbf{u}_{w_i} = \mathbf{P}^\top \cdot \check{\mathbf{W}}^\top \cdot \mathbf{v}_{w_i} \tag{13}$$

where, as for the clustering approach, we experimented with both the whole vocabulary and the most 100,000 used words.

4.2.4 Vector Significance

As premised, we considered word embeddings norm to identify the significance of a term. We composed the universe matrix sorting the word vectors in decreasing Euclidean norm order and taking the first n. During the experiments, we varied n in the set $\{100, 1000, 10,000, 25,000\}$.

4.3 Data

We evaluated our SFBoW through a series of reference benchmarks; we selected the STS benchmark series, one of the tasks of the International Workshop on Semantic Evaluation (SemEval).[2]

SemEval is a series of evaluations on computational semantics; among these, the *semantic textual similarity* (STS) benchmark[3] [13] has become a reference for scoring of sentence embedding algorithms. All the previous models we are considering for comparison have been benched against STS; this is because the benchmark highlights a model capability to provide a meaningful semantic representation by scoring the correlation between model's and human's judgements. For this reason, and also to allow comparisons, we decided to evaluate SFBoW on STS.

We worked only on the English language, using the editions of STS from 2012 to 2016 [1–5]. Each year, a collection of corpora coming from different sources has been created and manually labelled; Table 1 shows a reference, in terms of support, for each edition. Thanks to the high number of samples, we are confident about the robustness of our results.

To preprocess the input text strings, we lowercased each character and tokenised in correspondence of spaces and punctuation symbols. Then, from the resulting sequence, we retained only the tokens for which a corresponding embedding was found in the vocabulary known by the model. Finally, we calculated the SFBoW sentence embedding from the word embeddings of such tokens.

The samples constituting the corpora are a pair of sentences with a human-given similarity score (the *gold labels*). The provided score is a real-valued index obtained averaging those of multiple crowd-sourced workers and is scaled in a $[0, 1] \in \mathbb{R}$

Table 1 Support of the corpora of the STS benchmark series

STS edition	2012	2013	2014	2015	2016
No. of sentence pairs	5250	2250	3750	3000	1186

[2] https://aclweb.org/aclwiki/SemEval_Portal.

[3] https://ixa2.si.ehu.eus/stswiki/index.php/Main_Page.

interval. The final goal of our work is to provide a model able to provide a score as close as possible to that of humans.

4.4 Evaluation Approach

To assess the quality of our model, we used it to compute the similarity score between the sentence pairs provided by the five tasks, and we compared the output with the target labels. The results are computed as the correlation between the similarity score produced by SFBoW and the human one, using Spearman's ρ measure [32]. SFBoW employs fuzzy Jaccard similarity index [42] to compute word similarity.

To have terms of comparison, we establish a baseline through the most straightforward models possible, the average word embedding in a sentence, leveraging three different word embedding models: Word2Vec, GloVe and fastText. We also provide results from more complex models: SIF weighting (applied to GloVe), Sent2Vec, DynaMax (built using Word2Vec, GloVe and fastText) and Sentence-BERT.

All the embedding models except DynaMax and the baselines are scored using cosine similarity; DynaMax scores are obtained using fuzzy Jaccard similarity index.

5 Results

To analyse the results of the considered reference embeddings and the approaches to build the universe matrix, we reported, respectively, the aggregated Spearman's ρ correlation in the STS benchmark in Tables 2 and 3. Through these two tables, we highlight how the choice of an embedding model rather than a universe matrix approach affected the overall SFBoW performances in the STS benchmark. Additionally, we report a comparison in terms of Spearman's ρ correlation in the STS benchmark of our SFBoW against other sentence embedding models in Table 4. The comparison values, reported in the last three rows of Table 4, belong to the SFBoW configurations that achieved the best score, among the variants we considered for the experiments, in at least one task.

5.1 Individual SFBoW Results

As reported in Table 4, fastText yields the best absolute results among the four-word embedding models, confirming the results of DynaMax. The best scores in terms of universe matrix are achieved either with identity matrix or with PCA rotation matrix,

Table 2 SFBoW aggregated results over the STS benchmark. Results are aggregated on the employed word embedding model. Total scores are weighted averages across the STS editions and are expressed as *avg.*±*std.* Bold and underlined values represent, respectively, the first and second best results of a column

Reference embedding model	Results (Spearman's ρ)					
	STS					Total
	2012	2013	2014	2015	2016	
Word2Vec	51.25 ± 4.79	42.98 ± 5.27	57.62 ± 5.92	62.74 ± 6.90	**62.81** ± 5.91	54.71 ± 5.63
GloVe	52.71 ± 5.14	43.40 ± 5.36	54.47 ± 7.20	61.55 ± 7.47	62.61 ± 6.32	54.26 ± 6.21
fastText	**54.00** ± 4.90	**44.16** ± 4.86	54.89 ± 7.60	61.62 ± 7.57	62.13 ± 7.31	54.88 ± 6.26
Sent2Vec	53.13 ± 1.46	41.48 ± 2.42	**59.17** ± 2.70	**64.81** ± 2.97	**62.81** ± 2.18	**55.91** ± 2.25

Table 3 SFBoW aggregated results over the STS benchmark. Results are aggregated on the universe matrix building approach. Total scores are weighted averages across the STS editions and are expressed as *avg.*±*std.* Bold and underlined values represent, respectively, the first and second best results of a column

Universe matrix approach	Results (Spearman's ρ)					
	STS					Total
	2012	2013	2014	2015	2016	
Clustering	53.04 ± 3.60	42.69 ± 4.17	56.42 ± 5.45	62.81 ± 5.57	62.62 ± 4.42	54.99 ± 4.58
Identity	56.90 ± 3.87	**49.45** ± 3.61	**64.56** ± 2.20	**70.59** ± 1.82	69.33 ± 4.20	**61.29** ± 3.05
Multivariate analysis	**57.53** ± 3.27	48.64 ± 3.44	64.26 ± 1.77	70.20 ± 1.89	**69.80** ± 4.02	61.27 ± 2.72
Vector significance	48.49 ± 3.53	39.61 ± 2.48	51.05 ± 5.29	56.51 ± 5.36	57.18 ± 4.53	50.04 ± 4.24

highlighting how the features yield by word embeddings provide a better semantic content representation of sentences.

To have a better understanding of the results and the performances of different universe matrices, we broke down the results along two axes. On one side, we aggregated the results distinguishing among the different embedding models (see Table 2), and on the other, we distinguished among the different approaches to build the universe matrix (see Table 3).

From Table 2, we noticed that, despite being fastText the word embedding model yielding the best performances, Sent2Vec achieved the best results on average. While the remaining models achieved on average very similar scores—all differences in Spearman's ρ are < 1—Sent2Vec detached from fastText (the second best model on average) with a difference > 1 in Spearman's ρ score. We hypothesise that this is due to the fact that Sent2Vec, different from the other embeddings, is actually a parametric sentence embedding model, which yields embeddings for single words. However, despite being different, the average results of all models are quite close, especially if compared with the differences found among average the universe matrix results.

Table 4 Comparison of results over the STS benchmark. SFBoW models are in the last block. Total scores are weighted averages across the STS editions and are expressed as *avg.±std.* Bold and underlined values represent, respectively, the first and second best results of a column. Inference time refers to the time, in seconds, to carry out an evaluation on the entire STS corpus

	Results (Spearman's ρ)						
	STS						
Model	2012	2013	2014	2015	2016	Total	Analysis time [s]
Word2Vec[a]	55.46	58.23	64.05	67.97	66.28	61.21 ± 5.04	–
GloVe[a]	53.28	50.76	55.63	59.22	57.88	54.99 ± 2.80	–
fastText[a]	58.82	58.83	63.42	69.05	68.24	62.65 ± 4.20	–
SIF weighting[b]	56.04	62.74	64.29	69.89	70.71	62.84 ± 5.54	–
Sent2Vec	56.26	57.02	65.82	74.46	69.01	63.21 ± 7.13	–
DynaMax[c]	55.95	60.17	65.32	73.93	71.46	63.53 ± 6.92	–
DynaMax[b]	57.62	55.18	63.56	70.40	71.36	62.25 ± 5.85	–
DynaMax[d]	61.32	61.71	66.87	76.51	74.71	66.71 ± 6.10	–
Sentence-BERT	**72.27**	**78.46**	**74.90**	**80.99**	**76.25**	**75.81** ± 3.27	218.3
SFBoW[d,e,f]	61.31	51.21	67.47	72.90	73.88	64.55 ± 7.20	56.5
SFBoW[d,g,h]	61.42	51.36	66.44	72.74	73.72	64.32 ± 7.00	56.8
SFBoW[d,g,i]	60.03	51.96	66.36	72.39	73.25	63.81 ± 6.93	56.6

[a] Used as baseline
[b] Built upon a GloVe model for word embeddings
[c] Built upon a Word2Vec model for word embeddings
[d] Built upon a fastText model for word embeddings
[e] Best average score
[f] Universe matrix is the identity matrix
[g] Universe matrix is the PCA projection matrix
[h] Universe matrix is built from the English vocabulary
[i] Universe matrix is built from the top 100,000 most frequent words

From Table 3, instead, we noticed that there is a clear difference in performances among the considered approaches. Identity matrix and PCA universe matrices consistently outperform all the other considered approaches, achieving also very close scores between them—the difference between their average Spearman's ρ is only 0.02. Moreover, identity and PCA achieve scores very similar to the SFBoW predecessor (see Table 4). We hypothesise that this is due to the fact that these two techniques preserve the features extracted by the embedding models, which are very robust, as observed by other non-parametric sentence embedding models like SIF weighting.

Clustering, instead, presents way worse performances: the drop in Spearman's ρ is > 5 with respect to identity and PCA. Nevertheless, clustering scores are in line with the single word embedding model's averages.

Vector significance turned out to provide the worst overall results. We hypothesise it is due to the fact that the significance is not strongly related to the semantic representative capabilities.

5.2 Comparison with Other Models

As premised, we compare our results with three baseline models and other sentence embedding approaches, all reported in Table 4. The first group of scores is from the baselines, the second one is from other sentence embedding models, and, finally, the last group is from our SFBoW model. Additionally, the best values in each column are highlighted in bold, while the second ones are underlined.

The key features about our model, which can be derived from the results, are the following:

- Low number of parameters
- Faster inference time
- No training phase
- Results (in terms of ρ) comparable to similar models
- Fixed-size and easily re-usable embeddings

About the number of parameters, we can notice that even if Sentence-BERT outperforms all the other models in every task, it relies on a much deeper feature extraction model and was trained on a much bigger corpus. Moreover, this model requires a considerably higher computational effort without an equally consistent difference in performances. BERT alone requires more than 100 million parameters just for its base version (and above 300 million for the large one), hence taking a lot of (memory) space, not to mention the amount of time necessary for the self-supervised training and the fine-tuning. On the other hand, non-parametric models (like SIF, DynaMax or SFBoW) or shallow parametric ones (Sent2Vec) require fewer parameters: just those for the embedding matrix $|\mathcal{V}| \times d$.

A similar discourse applies to inference speed. Even though Sentence-BERT achieves the best results on all tasks, SFBoW turns out to be four times faster at inferring the similarity, as can be noticed by the reported analysis times.

Being a non-parametric model, SFBoW does not require a training phase. It may require clustering the embeddings to build the universe matrix, but our experiments showed that clustering does not yield good results. Because of its simplicity, SFBoW can generally be easily deployed, requiring only the word embedding model to compute the sentence representation. Notice also that the SFBoW algorithm is agnostic to the word embedding model.

Regarding the results we obtained, compared to other models, SFBoW provided interesting figures: either considering the majority of tasks with higher Spearman's ρ rank or higher average score, it outperforms all the baselines, as well as SIF weighting and Sent2Vec. Finally, we see as our model performs closely to its predecessor, especially considering the weighted average of the results of the single tasks. SFBoW bests out DynaMax in STS 2014 and gets almost the same results in STS 2012 (the difference is 0.01), which are the first two corpora in terms of samples; however, the difference in STS 2013 goes in favour of DynaMax.

About the comparison against DynaMax, it is worth underlining a few additional points. Firstly, in both cases, fuzzy Jaccard similarity correlates better with human judgement as a measure of sentence similarity. Secondly, both models manage to achieve better results when using fastText word embedding, possibly underling that they lend better than other models at sentence-level combination; the baseline performances also show this.

Finally, we remind that SFBoW generates embeddings with a fixed size, resulting in much easier applicability with respect to DynaMax.

6 Conclusion

In this paper, we presented and evaluated the SFBoW model for sentence embedding. This model leverages the approaches proposed by the FBoW and DynaMax models, to compute static embeddings (in the sense of fixed-size embeddings). To extract such static embeddings, we rely on a static universe matrix. This matrix can be constructed in many different ways; thus, we explored them in order to find the most suitable. We considered approaches based on clustering, identity, multivariate analysis and vector significance. To evaluate the possible approaches, we benchmarked the model on the STS benchmark.

We divided the evaluation into an individual one, to observe the different results of the considered embeddings and approaches for the SFBoW universe matrix, and a compared one, to observe the results of SFBoW with respect to those of other sentence embedding models.

From the individual analysis, we derived that fastText and Sent2Vec are the two most suitable embeddings for our model and that identity and PCA are the most suitable universe matrix building approaches. From the compared evaluation, we derived that even if SFBoW does not outperform state-of-the-art models on STS, it performs comparably to DynaMax, its predecessor, and, different from DynaMax, yields re-usable embeddings, because of their fixed dimensionality. Due to its low computation demand (especially if compared with state-of-the-art Sentence-BERT) and re-usability of embeddings, SFBoW can be seen as a reasonable solution, especially for scenarios where low computational capabilities are essential.

In the future, we plan to carry out a deeper analysis of the results to identify the reasons behind the different scores achieved by the universe matrix approaches. Another idea for future evolution we considered is to combine the approaches we analysed to build the universe matrix, in order to extract a more robust one. For example, it would be possible to cluster the vectors with a significance above a certain threshold to obtain, possibly, better results.

Acknowledgments This work was partially supported by the European Union's Horizon 2020 project *WorkingAge* (grant agreement no. 826232).

References

1. Agirre, E., Cer, D.M., Diab, M.T., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Agirre, E., Bos, J., Diab, M.T. (eds.) Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012, Montréal, Canada, June 7-8, 2012, pp. 385–393. The Association for Computer Linguistics (2012). https://www.aclweb.org/anthology/S12-1051/
2. Agirre, E., Cer, D.M., Diab, M.T., Gonzalez-Agirre, A., Guo, W.: *sem 2013 shared task: Semantic textual similarity. In: Diab, M.T., Baldwin, T., Baroni, M. (eds.) Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA, pp. 32–43. Association for Computational Linguistics (2013). https://www.aclweb.org/anthology/S13-1004/
3. Agirre, E., Banea, C., Cardie, C., Cer, D.M., Diab, M.T., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G., Wiebe, J.: Semeval-2014 task 10: Multilingual semantic textual similarity. In: Nakov, P., Zesch, T. (eds.) Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014, pp. 81–91. The Association for Computer Linguistics (2014). https://doi.org/10.3115/v1/s14-2010
4. Agirre, E., Banea, C., Cardie, C., Cer, D.M., Diab, M.T., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., Wiebe, J.: Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In: Cer, D.M., Jurgens, D., Nakov, P., Zesch, T. (eds.) Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015, pp. 252–263. The Association for Computer Linguistics (2015). https://doi.org/10.18653/v1/s15-2045
5. Agirre, E., Banea, C., Cer, D.M., Diab, M.T., Gonzalez-Agirre, A., Mihalcea, R., Rigau, G., Wiebe, J.: Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In: Bethard, S., Cer, D.M., Carpuat, M., Jurgens, D., Nakov, P., Zesch, T. (eds.) Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016, pp. 497–511. The Association for Computer Linguistics (2016). https://doi.org/10.18653/v1/s16-1081
6. Arora, S., Liang, Y., Ma, T.: A simple but tough-to-beat baseline for sentence embeddings. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net (2017). https://openreview.net/forum?id=SyK00v5xx
7. Baeza-Yates, R., Ribeiro-Neto, B.A.: Modern Information Retrieval - The Concepts and Technology Behind Search, Second edition. Pearson Education Ltd., Harlow, England (2011). http://www.mir2ed.org/
8. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 238–247. Association for Computational Linguistics, Baltimore, Maryland (2014). https://doi.org/10.3115/v1/P14-1023. https://aclanthology.org/P14-1023
9. Bengio, Y., Courville, A.C., Vincent, P.: Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013). https://doi.org/10.1109/TPAMI.2013.50
10. Bengio, Y., LeCun, Y., Hinton, G.E.: Deep learning for AI. Commun. ACM **64**(7), 58–65 (2021). https://doi.org/10.1145/3448250
11. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017). https://transacl.org/ojs/index.php/tacl/article/view/999
12. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler,

E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

13. Cer, D.M., Diab, M.T., Agirre, E., Lopez-Gazpio, I., Specia, L.: Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In: Bethard, S., Carpuat, M., Apidianaki, M., Mohammad, S.M., Cer, D.M., Jurgens, D. (eds.) Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3–4, 2017, pp. 1–14. Association for Computational Linguistics (2017)
14. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011). http://dl.acm.org/citation.cfm?id=2078186
15. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019 (Volume 1, Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423
16. Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada, pp. 3294–3302 (2015). https://proceedings.neurips.cc/paper/2015/hash/f442d33fa06832082290ad8544a8da27-Abstract.html
17. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
18. Liu, Q., Kusner, M.J., Blunsom, P.: A survey on contextual embeddings. CoRR **abs/2003.07278** (2020). https://arxiv.org/abs/2003.07278
19. Liu, Z., Lin, Y., Sun, M.: Representation Learning for Natural Language Processing. Springer (2020). https://doi.org/10.1007/978-981-15-5573-2
20. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008). https://doi.org/10.1017/CBO9780511809071. https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf
21. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings (2013). http://arxiv.org/abs/1301.3781
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States, pp. 3111–3119 (2013). https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html
23. Osgood, C.E., Suci, G.J., Tannenbaum, P.H.: The measurement of meaning. Am. J. Sociol. **63**(5), 550–551 (1958)
24. Pagliardini, M., Gupta, P., Jaggi, M.: Unsupervised learning of sentence embeddings using compositional n-gram features. In: Walker, M.A., Ji, H., Stent, A. (eds.) Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018 (Volume 1, Long Papers), pp. 528–540. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/n18-1049

25. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pp. 1532–1543. ACL (2014). https://doi.org/10.3115/v1/d14-1162
26. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. OpenAI blog **1**(11), 12 (2018). https://openai.com/blog/language-unsupervised/
27. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019). https://openai.com/blog/better-language-models/
28. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140:1–140:67 (2020). http://jmlr.org/papers/v21/20-074.html
29. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100, 000+ questions for machine comprehension of text. In: Su, J., Carreras, X., Duh, K. (eds.) Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1–4, 2016, pp. 2383–2392. The Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/d16-1264
30. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for squad. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15–20, 2018, Volume 2: Short Papers, pp. 784–789. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/P18-2124. https://aclanthology.org/P18-2124/
31. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3–7, 2019, pp. 3980–3990. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/D19-1410
32. Reimers, N., Beyer, P., Gurevych, I.: Task-oriented intrinsic evaluation of semantic textual similarity. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 87–96. The COLING 2016 Organizing Committee, Osaka, Japan (2016). https://aclanthology.org/C16-1009
33. Salton, G.: The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall Series in Automatic Computation. Prentice-Hall (1971)
34. Schakel, A.M.J., Wilson, B.J.: Measuring word significance using distributed representations of words. CoRR **abs/1508.02297** (2015). http://arxiv.org/abs/1508.02297
35. Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.Y., Gao, J., Dolan, B.: A neural network approach to context-sensitive generation of conversational responses. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 196–205. Association for Computational Linguistics, Denver, Colorado (2015). https://doi.org/10.3115/v1/N15-1020. https://aclanthology.org/N15-1020
36. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13, 2014, Montreal, Quebec, Canada, pp. 3104–3112 (2014). https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017,

December 4–9, 2017, Long Beach, CA, USA, pp. 5998–6008 (2017). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

38. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: Linzen, T., Chrupala, G., Alishahi, A. (eds.) Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018, pp. 353–355. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/w18-5446
39. Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Superglue: A stickier benchmark for general-purpose language understanding systems. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada, pp. 3261–3275 (2019). https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html
40. Yih, W., He, X., Gao, J.: Deep learning and continuous representations for natural language processing. In: Mihalcea, R., Chai, J.Y., Sarkar, A. (eds.) NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31–June 5, 2015, pp. 6–8. The Association for Computational Linguistics (2015). https://doi.org/10.3115/v1/n15-4004
41. Zhao, R., Mao, K.: Fuzzy bag-of-words model for document representation. IEEE Trans. Fuzzy Syst. **26**(2), 794–804 (2018). https://doi.org/10.1109/TFUZZ.2017.2690222
42. Zhelezniak, V., Savkov, A., Shen, A., Moramarco, F., Flann, J., Hammerla, N.Y.: Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net (2019). https://openreview.net/forum?id=SkxXg2C5FX

Index

M. Abbas (ed.), *Analysis and Application of Natural Language and Speech Processing*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-031-11035-1