Alex: A Statistical Dialogue Systems Framework

Filip Jurčíček, Ondřej Dušek, Ondřej Plátek, and Lukáš Žilka

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{jurcicek,odusek,oplatek,zilka}@ufal.mff.cuni.cz
https://ufal.mff.cuni.cz

Abstract. This paper describes the Alex Dialogue Systems Framework (ADSF). The ADSF currently includes mature components for public telephone network connectivity, voice activity detection, automatic speech recognition, statistical spoken language understanding, and probabilistic belief tracking. The ADSF is used in a real-world deployment within the Public Transport Information (PTI) domain. In PTI, users can interact with a dialogue system on the phone to find intra- and inter-city public transport connections and ask for weather forecast in a desired city. Based on user responses, vast majority of the system users are satisfied with the system performance.

Keywords: automatic speech recognition, spoken language understanding, dialogue state tracking, dialogue systems.

1 Introduction

This paper introduces the Alex Dialogue Systems Framework (ADSF). The ADSF is freely available for download on GitHub[1] and designed for easy adaptation to new domains and languages. The ambition of the ADSF is to provide a modular platform for experimenting with statistical methods in the area of spoken dialogue systems. As of now, the system already allows the use of different data-driven components for automatic speech recognition, spoken language understanding, and dialogue state tracking. Since the framework follows a modular design, the system allows for easy replacement of individual components.

The ADSF is used for the development of our experimental Czech spoken dialogue system in the Public Transit Information (PTI) domain [6]. In PTI, the dialogue system provides information about all kinds of public transport in the Czech Republic and is publicly available at a toll-free 800 telephone number. It was launched into public use as soon as a first minimal working version was developed. Nine months after the launch, we have collected nearly 900 calls from the general public. The domain supported by the system includes transport information among ca. 5,000 towns and cities in the whole Czech Republic, plus weather and time information.

In the next section, the overall structure of our ADSF is described. Section 3 describes the PTI domain in detail and analyses its public deployment. Finally, Section 4 concludes this work.

[1] https://github.com/UFAL-DSG/alex

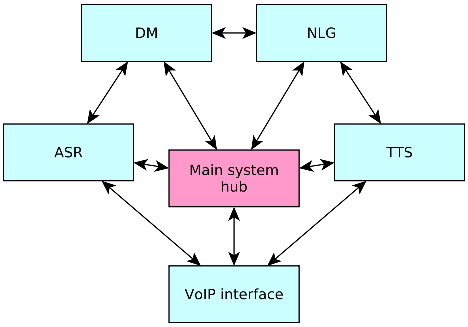

Fig. 1. The structure of our SDS framework

2 Framework Structure and the Main Components

The basic architecture of our system is modular and consists of the traditional Spoken Dialogue System (SDS) components: an automatic speech recogniser (ASR) including a voice activity detector, spoken language understanding (SLU), dialogue manager (DM), natural language generator (NLG), and a text-to-speech (TTS) module.

The framework is developed in Python, a high level programing language. If needed due to performance issues, C/C++ extensions can be easily integrated with Python. To ensure real-time responsiveness of the whole system, each of the components runs in a separate process on a multi-core machine and communicates with other components via operating system pipes. The central component is a hub which coordinates all components used within the framework. All components have access to a common logging process to allow for synchronised logging of events and data. The logged data includes input and output audio streams, speech segments identified by voice activity detector, ASR hypotheses, SLU hypotheses, dialogue state for each dialogue turn, the responses of the in the form of dialogue acts, and the output of the NLG module. A schema of the system is shown in Figure 1.

In the remainder of the section, the VoIP, ASR, SLU and DM components are described in detail as they already reached maturity. Concerning the NLG and TTS components, NLG uses a simple template-based approach and TTS uses web-based TTS services such as SpeechTech[2] or VoiceRSS[3].

2.1 Voice-Over-IP Interface

The Voice-Over-IP interface is based on the freely available PJSIP 2.1 library[4]. To bring real-time processing of voice audio into Python, a fork of the project[5] was created. This fork enables applications written in Python to perform in-memory audio signal recording and playing, which was not possible with the standard Python module

[2] `http://www.speechtech.cz`
[3] `http://www.voicerss.org/`
[4] `http://www.pjsip.org/`
[5] `https://github.com/UFAL-DSG/pjsip`

provided by the PJSIP 2.1 library. The interface allows for standard telephone network functionality such as answering incoming calls or dialling outgoing calls.

2.2 Automatic Speech Recognition

Same as in most traditional dialogue systems, the ADSF uses voice activity detection to split the incoming audio signal stream into speech and silence segments. To perform voice activity detection, the ADSF implements two algorithms: Gaussian Mixture Models and Feed-Forward Neural Networks. Both approaches are trained from data force-aligned by the HTK[6]. Once speech segments are identified, they are sent to an ASR component to be recognised. In ADSF, two ASR systems are currently supported: the Google cloud-based ASR and Kaldi OnlineLatgenRecogniser.

Google ASR. The main advantage of the Google cloud-based ASR[7] is that it is relatively fast and can be used off-the-shelf without any additional modifications. However, the Word Error Rate (WER) of the Google ASR in the PTI domain is about 45% and its average latency is well above 400 ms. The high WER is likely caused by a mismatch between Google's acoustic and language models and the test data, and the high latency is presumably caused by the batch processing of audio data and network latency.

Kaldi OnlineLatgenRecogniser. The OnlineLatgenRecogniser is our own extension of the state-of-the-art Kaldi automatic speech recognition toolkit [5]. The OnlineLatgenRecogniser recogniser may use acoustic models trained using techniques such as Linear Discriminant Analysis (LDA), Maximum Likelihood Linear Transform (MLLT), Boosted Maximum Mutual Information (BMMI), or Minimum Phone Error (MPE). In the PTI domain, OnlineLatgenRecogniser yields a performance of 18.54% WER when trained on VYSTADIAL 2013 – Czech telephone speech data[8] and the PTI speech data using BMMI discriminative training with LDA and MLLT feature transformations. A class based N-gram language model with a vocabulary consisting of approximately 52k words was estimated only from the PTI speech data transcriptions. Informal experiments show that the OnlineLatgenRecogniser has a latency of about 60 ms on average. The scripts used for training the acoustic models are available in the ADSF repository and a detailed description of the data and training procedure is given in [12]. The OnlineLatgenRecogniser is distributed under the Apache 2.0 license and the source code is available in the Kaldi repository [9].

[6] `http://htk.eng.cam.ac.uk/`

[7] The API is accessible at `https://www.google.com/speech-api/v1/recognize`, and its use is described in a blog post at `http://mikepultz.com/2013/07/google-speech-api-full-duplex-php-version/`.

[8] `http://hdl.handle.net/11858/00-097C-0000-0023-4670-6`

[9] `https://sourceforge.net/p/kaldi/code/HEAD/tree/sandbox/oplatek2/src/dec-wrap/`

Czech utterance:	“v kolik hodin to bude na stanici veletržní palác”
English translation:	*what time does it arrive to the veletržní palác stop*
Dialogue act:	`inform(to_stop="Veletržní palác")&request(arrival_time)`

Fig. 2. Example dialogue act representation of an utterance

2.3 Spoken Language Understanding

The meaning extracted from the recognised utterances is represented using dialogue acts. A dialogue act (DA) is composed of one or more dialogue act items, where each dialogue act item represents basic intents (such as `inform`, `request`, etc.) and optionally the semantic content, also referred to as slots, in the input utterance (e.g. `vehicle=bus`, `time=1:30`). In some cases, the value of a slot can be omitted, for example when the intention is to query it, as in `request(arrival_time)`. An example of this is shown in Figure 2.

In ADSF, trainable spoken language understanding is provided by a probabilistic discriminative module based on dialogue act item classifiers using logistic regression (DAICLR). Since the number of possible slot values for each slot is generally very high, a form generalisation using gazetteers with surface forms for slots categories such as city and stop is implemented [8,11,16,17]. The DAICLR parser may be directly trained on the output of the ASR component to learn to deal with systematic speech recognition errors. In the DAICLR parser, only lexical N-gram features are extracted from the input utterance. To prevent overfitting, simple feature pruning based on counts of occurrences in the training data was introduced. To process ASR N-best lists, the same feature extraction process is performed for each hypothesis in the list. However, the final feature set is a weighted combination of features extracted from individual ASR hypotheses where the weights correspond to the probabilities of the hypotheses. The DAICLR parser is very computationally efficient on domains such as PTI because the typical number of classifiers is small. There are 24 unique dialogue act types and 21 unique slots in the PTI domain and the total number of classifiers trained is 135. The source code of the DAICLR parser is available in the ADSF repository[10].

2.4 Dialogue Management

Dialogue management in ADSF is split into two components: dialogue state tracking and dialogue policy.

The purpose of state tracking is to monitor dialogue progress and provide a compact representation of the past user input and system output in the form of a dialogue state. To represent the uncertainty in the estimate of the dialogue state, statistical dialogue systems maintain a probability distribution over all dialogue states called the *belief state* [4].

The dialogue policy is then responsible for decisions on how to continue in a conversation. This can be implemented, for example, using handcrafted rule-based policies or data driven policies optimised by reinforcement learning.

[10] `https://github.com/UFAL-DSG/alex/blob/master/alex/components/slu/dailrclassifier.py`

Belief tracking. In ADSF, we use a probabilistic discriminative tracker [7] which achieves near state-of-the-art performance while remaining completely parameter-free. This is possible because it uses a simple deterministic state transition probability distribution. Interestingly, this discriminative model gives performance comparable to more complex trackers; yet it is significantly more computationally efficient.

Dialogue policy. As of now, the ADSF only supports handcrafted dialogue policies. However, it can take take advantage of uncertainty estimated by the belief tracker. The main logic of the dialogue policy implemented for PTI is similar to that of [18]. First, it implements a set of domain-independent actions, such as: dialogue opening, closing, and restart, implicit confirmation of changed slots with high probability of the most probable value, explicit confirmation for slots with a lower probability of the most probable value, or a choice among two similarly probable values. Second, domain-specific actions are implemented for the PTI domain, such as: informing about a connection, informing about weather forecast, informing about current time.

3 Public Deployment: Transport Information Domain

The ADSF is used for the development of our experimental Czech spoken dialogue system in the PTI domain. In the PTI domain, a user can ask for information about public transport connections in Czech Republic, weather forecast, and current time. Users can control conversation by:

– asking for help,
– asking to “restart” the dialogue and start a new conversation,
– ending the call - for example, by saying “good-bye”,
– asking for a repetition of the last system utterance,
– confirming or rejecting system questions.

When asking for transport connections, the user provides his/her origin and destination and the application finds a connection. The user may specify departure or arrival time in absolute or relative terms (“in ten minutes”, “tomorrow morning”, “at 6 pm.”, “at 8:35” etc.) and request more details about the connection: number of transfers, journey duration, departure and arrival time. The user may travel not only among public transport stops within one city, but also among multiple cities or towns. The system is able to infer the city name from stop name and use a default stop for the given city. As of now, the system supports ca. 44,000 stops in 5,000 cities and towns using the Czech national public transport database provided by CHAPS[11].

Weather information service provided by OpenWeatherMap[12] covers all Czech cities. The user may ask for weather at the given time or on the whole day.

The system is accessible at a public toll-free 800 number. We advertised the service at our university, among friends, and via Facebook. We cooperate with the Czech

[11] `http://www.idos.cz`

[12] `http://openweathermap.org/`

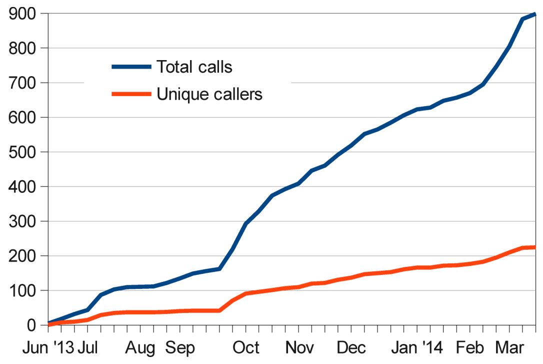

Fig. 3. Cumulative number of calls and unique callers from the public by weeks

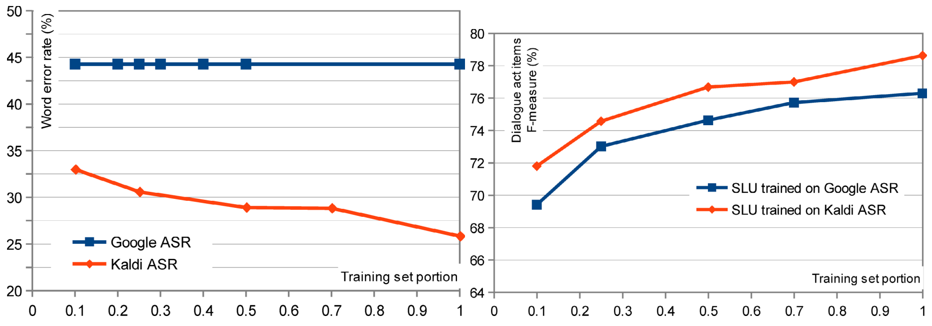

Fig. 4. ASR word error rate and SLU performance (F-measure on dialogue act items) depending on the size of in-domain training data

Blind United association[13], promoting our system among its members and receiving comments about its use.

We record and collect all calls to the system, including our own testing calls, to obtain training data for the satistical components of our system. The number of calls and unique users (caller phone numbers) grows steadily; so far, over 200 users from the public have made nearly 900 calls to the system (see Figure 3)[14]. Information was provided to the user in the vast majority of calls (740 out of 900). We included a measure of the subjective user success rate into our system. After the users says good-bye, the system asks them if they received the information they were looking for. By looking at the transcriptions of responses to this question, we recognise about 90% of them as rather positive (“Yes”, “Nearly” etc.).

While the system was deployed, we have been gradually improving our ASR and SLU components. A performance comparison of Google ASR with Kaldi ASR trained on our data is shown in Figure 4 (left). One can see that the Kaldi ASR improves as

[13] `http://www.sons.cz`

[14] We only count calls with at least one valid user utterance, disregarding calls where users hang up immediately.

more training data is available over time and that it outperforms Google ASR very early. Figure 4 (right) shows that the performance of the statistical SLU module improves with more training data. Interestingly, the trainable SLU helps significantly to mitigate the high WER of Google ASR.

4 Conclusion

This paper presents the Alex Dialogue Systems Framework (ADSF). Although its development is in an early stage, it already includes Voice-over-IP interface, various automatic speech recognition options, statistical spoken language understanding, and probabilistic belief tracking. The ADSF is an extensible dialogue system framework and is available for download under the Apache 2.0 license.

The ADSF is currently used in a dialogue system within the public transport information domain. The system is publicly available on a toll-free phone number. We have already collected nearly 900 calls with real users and this is currently increasing by 10 calls per day on average. The analysis of our call logs shows that our system is able to provide information in the vast majority of cases. Success rating provided by the users themselves is very positive.

Future plans include development of data-driven models for belief tracking, dialogue policy, and natural language generation. The PTI domain is expected to serve as an evaluation platform with real users.

Acknowledgments. This research was partly funded by the Ministry of Education, Youth and Sports of the Czech Republic under the grant agreement LK11221, core research funding, grant GAUK 2058214, and 2076214 of Charles University in Prague. This work has been using language resources distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013).

References

1. Kate, R.J., Wong, Y.W., Mooney, R.J.: Learning to Transform Natural to Formal Languages. In: Proceedings of AAAI, pp. 1062–1068 (2005)
2. Mairesse, F., Gasic, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., Young, S.: Spoken language understanding from unaligned data using discriminative classification models. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4749–4752 (2009)
3. Thomson, B., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Yu, K., Young, S.: User study of the Bayesian update of dialogue state approach to dialogue management. In: Proceedings of Interspeech, pp. 483–486 (2008)
4. Williams, J., Young, S.: Partially observable Markov decision processes for spoken dialog systems. Computer Speech and Language 21(2), 393–422 (2007)
5. Povey, D., et al.: The Kaldi speech recognition toolkit. In: Proc. ASRU, Hawaii, US, pp. 1–4 (December 2011)

6. Public Transport Information System for Czech Republic (2014), `https://ufal.mff.cuni.cz/alex-dialogue-systems-framework/ptics`
7. Žilka, L., Marek, D., Korvas, M., Jurčíček, F.: Comparison of Bayesian Discriminative and Generative Models for Dialogue State Tracking. In: SIGDIAL 2013: Proc. of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Metz, France, pp. 452–457 (2013)
8. He, Y., Young, S.: Semantic processing using the Hidden Vector State model. Computer Speech & Language 19(1), 85–106 (2005)
9. Jurčíček, F., Švec, J., Müller, L.: Extension of the HVS semantic parser by allowing left-right branching. In: Proceedings of ICASSP, pp. 4993–4996 (2008)
10. Zhu, J., Hastie, T.: Kernel logistic regression and the import vector machine. Journal of Computational and Graphical Statistics 14(1), 1081–1088 (2005)
11. Zettlemoyer, L.S., Collins, M.: Online learning of relaxed CCG grammars for parsing to logical form. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 678–687 (2007)
12. Korvas, M., Plátek, O., Dušek, O., Žilka, L., Jurčíček, F.: Free English and Czech telephone speech corpus shared under the CC-BY-SA 3.0 license. In: Proceedings of International Conference on Language Resources and Evaluation, pp. 4423–4428 (2014)
13. The Kaldi ASR toolkit (2014), `http://sourceforge.net/projects/kaldi`
14. The Alex Dialogue Systems Framework (2014), `https://github.com/UFAL-DSG/alex`
15. Dahl, D.A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., Shriberg, E.: Expanding the scope of the ATIS task: The ATIS-3 corpus. In: Proceedings of the ARPA HLT Workshop, pp. 43–48 (1994)
16. Meza-Ruiz, I.V., Riedel, S., Lemon, O.: Spoken Language Understanding in dialogue systems, using a 2-layer Markov Logic Network: Improving semantic accuracy. In: Proceedings of Londial (2008)
17. Tür, G., Hakkani-Tür, D.Z., Hillard, D., Celikyilmaz, A.: Unsupervised Spoken Language Understanding: Exploiting Query Click Logs for Slot Filling. In: Proceedings of Interspeech, pp. 1293–1296 (2011)
18. Jurčíček, F., Thomson, B., Young, S.: Reinforcement learning for parameter estimation in statistical spoken dialogue systems. Computer Speech & Language 26(3), 168–192 (2012)