Deep Learning–Based Constituency Parsing for Arabic Language

Amr Morad, Magdy Nagi, and Sameh Alansary

1 Introduction

Developing an effective algorithm to generate constituency parse tree is a major challenge in Natural Language Processing (NLP). Constituency parse tree is the first and most important step, prior to several tasks in NLP such as word processors grammar checking [1], language translation [2], Arabic diacritization, and many more.

The past few years have witnessed a significant increase in adopting neural network–based approaches for generating constituency parse tree [3, 4]. These approaches, which have proved to be performant and adaptive, are called data-driven ones as they extract the rules from the data itself, hence they are adaptive to any language, provided that the dataset is changed. Moreover, most of these techniques rely on dense input representations [5] for words in the given sentences. In such representation, semantic meaning of the words is disclosed by making the representation of similar words closer according to the task at hand. These

A. Morad (✉)
Bibliotheca Alexandrina, Information and Communication Technologies (ICT), Alexandria, Egypt
e-mail: amr.morad@bibalex.org

M. Nagi
Bibliotheca Alexandrina, Faculty of Eng., Alexandria University, Computer and Systems Department, Alexandria, Egypt
e-mail: magdy.nagi@bibalex.org

S. Alansary
Bibliotheca Alexandrina, Arabic Computational Linguistic Center, Faculty of Arts, Alexandria University, Phonetics and Linguistics Department, Alexandria, Egypt
e-mail: sameh.alansary@bibalex.org

H. R. Arabnia et al. (eds.), *Advances in Artificial Intelligence and Applied Cognitive Computing*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70296-0_4

word representations are typically achieved by deep learning analysis based on the input sentences. This representation of words is called “dense input representation,” which in turn needs deep learning–based technique, and the dataset used in training of these models could be in different languages; however, it needs different resources for the required language in order to find this similarity.

In this chapter, a neural network syntactic distance-based model [6] is applied on Arabic sentences. The Arabic language is challenging because of its history, culture, and literary heritage, as well as being very complex language because of its linguistic structure [7]. Some words spell the same, but have different meanings, depending on their context and diacritization. Moreover, there are multiple Arabic words spellings corresponding to one word in other languages, especially in case of known entities, for example, the city of Washington could be spelled ‘واشنطن’ ‘وشنطن’, ‘واشنجطن’. As a result, novel challenges are considered when applying deep learning techniques to Arabic language. Arabic is not only a complex language, but also has very limited linguistic resources, specially parsed corpora (Fig. 1).

The model, adopted in this chapter, is based on the concept of syntactic distance [6], which is defined for each split point of the sentence. For implementation, dense input representation is achieved by using Glove [8] (see Sect. 3 below), for Arabic corpus. Several experiments are carried out in this work, with various split points, based on linguistic factors and length in Arabic sentences (see Sect. 6).

In order to predict the correct label (nonterminal tag) accurately, a list of these labels is determined, and its dimensionality should be affordable, meaning that this list should not be too long to avoid the problem of dimensionality curse. The curse of dimensionality refers to a well-known problem encountered during data analysis, in high-dimension space, however, as previously mentioned, our Arabic dataset is limited. Therefore, the full list of Arabic labels, nonterminal tags, is reviewed by linguists and then grouped. This happens by grouping similar labels together in one label. For example, label “NOUN.VN + NSUFF_FEM_SG + CASE_ DEF_GEN” is turned into “NOUN.” The rest of the label linguistic features, denoting the gender of the noun, whether it is definite or not and so on, is stored in the node displayed in the resultant constituency parse tree. More explanation will be further illustrated in the experiments section (VI).

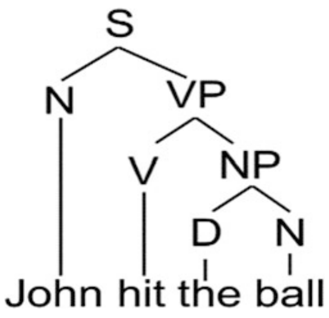

Fig. 1 A constituent parse tree

The model is fully parallel [6], which means that it is performant and capable of handling long sentences efficiently. Mapping from syntactic distance to the parse tree and vice versa can be done in O (n log n), which makes the decoding computationally efficient.

The obtained model is used in a complete workflow for linguists and editors, predicting the constituency parse tree for the Arabic sentence. A constituency parse tree viewer is implemented as a web-based application for linguists, to display the predicted tree. Tree viewer is user-friendly, enabling the linguists to edit the resultant displayed parse tree easily, and save the modified one. These valid and curated sentences will be used later to retrain the model, assisting in the existing challenge of lack of Arabic language resources.

2 Survey of Related Work

In Natural Language Processing (NLP), designing fast and accurate parsing algorithms for constituencies is a major challenge. Constituency parsing is necessary, as a first step toward the interpretation of the semantic meaning, which in turn is used as a foundation for other tasks, like translation [2] and grammar checking in word processors [1].

Traditionally, syntactic parsers represent the sentence as one-hot vector of a collection of unique features. This is accomplished by having only "1" in a location unique to the given word; "0" otherwise. However, enormous drawbacks were discovered for this representation technique. Apart from the length of this representation, it mainly handles all words as independent entities, with no relation to each other [8].

Transition-based dependency parsing [10] represents a step toward word similarity, which assigns similar words to the same buffer, in order to provide contextual meaning for these words. The problem in the case of getting word vectors is the necessity of a certain knowledge to the language of the task at hand, in order to extract the appropriate rules. For each language, various rules should be applied which makes rule-based techniques harder in application, as a general solution to achieve the words vectors representation. Hence, deep learning approaches are the most appropriate in tackling this issue.

GloVe (Global Vectors for Word Representation) [8] is a well-known unsupervised deep learning algorithm to generate vectors representation for words, called word vectors. It defines some sort of similarity between words by using deep learning approach and creates word vectors by using matrix factorization, with local window methods. In brief, the output word vectors represent the semantics of the word, which is relevant to the current task. These vectors are dense, meaning that their entries are typically nonzero, according to the features extracted from given sentences. GloVe could be considered a general technique for word embeddings for any language where a general deep learning model could be trained with different resources in various languages [9].

In this chapter, we trained GloVe model with multiple Arabic resources such as Arabic corpus, Elwatan news, and many others. As a result, GloVe is a global technique for any language, which is unsupervised learning of word representations.

The challenge represented by the Arabic language to researchers is mainly the inherent linguistic structure. Some Arabic words have different POS tags according to their place in the sentence or according to the vocalization. Another problem is when Arabic texts include many translated and transliterated known places, whose spelling might be inconsistent in Arabic texts. Moreover, Arabic language is one of the non-Latin languages, written from right to left, so the Arabic characters need different encoding techniques. In order to overcome some of these challenges, Buckwalter for Arabic characters transliteration [11] was introduced. Although Buckwalter addresses some of Arabic language challenges, it adds more complexities because it requires more encoding and decoding steps.

Another challenge in Arabic language labels is represented in the dimensionality of labels selected. Each Arabic word could have up to three prefixes and two suffixes. Accordingly, a label could be a combination depending on the context, whether it is singular or plural and so on. For example, label "NOUN.VN + NSUFF_FEM_ SG + CASE_DEF_GEN" refers to a NOUN with a suffix for a singular female, it also denotes that this word has definite and in genitive case. Hence, given these different components in each single label, it could be concluded that the number of possible labels will be huge.

Designing a model which has to predict a label among the large number of choices is really hard, whether this model is rule based or data driven (deep learning based). A short list, in cooperation with linguists, should be determined while preserving the original one, and without sacrificing the context of the word.

Recently, deep learning–based techniques relying on dense word vectors representation for constituency parsing are widely adopted [3, 4]. Some of them decompose the problem of generating the parse tree into local decisions sequentially. However, this makes the accuracy of such models low, because they were never exposed to their mistakes during the training phase. In order to solve this problem, a technique such as spa-based constituency parsing with a structure-label system [12] is proposed. The drawback of this technique is that the training phase is complicated.

One of these techniques is accomplished by adopting the concept of syntactic distance [6]. Syntactic distance is a vector of real valued scalars for each split point in the given sentence (e.g., white space). A fully parallel model for constituency parser is introduced for English and Chinese languages using syntactic distance notion [6]. Mainly, it uses a top-down approach to construct the parse tree by splitting larger constituents into smaller ones. A combined neural network is built to estimate the syntactic distances vector for a sentence. Algorithms for converting syntactic distances into binary parse trees and vice versa are explained with average running time of O (n log n) [6].

In this chapter, a combined neural network model is built to generate the parse tree for Arabic sentences without the need to use Buckwalter, because there is no known word embeddings for transliterated words. This step requires preprocessing and postprocessing as intermediate stages. Therefore, those two steps are eliminated in our solution as it deals with Arabic language directly. Arabic language sentences impose some complications such as the length of sentences and the ambiguity of words. As previously mentioned, words with the same spelling may have multiple different POS tags according to the context of this word or its diacritization. These challenges are addressed in this chapter by conducting several experiments to show the effect of these factors on the accuracy of the given model.

The model is fully parallel, which can make use of efficient and powerful Graphical Processing Units (GPU) capabilities. Several experiments have been established to discover the effect, in terms of the length of Arabic sentences, as well as the effect of split points, on the model's accuracy, running times, and accuracy. In addition, the model predicts the constituency parse-tree efficiently. This is important when the model is considered a step in a larger workflow, as presented in this chapter (see Sect. 5).

3 Dense Input Representation

Generating the constituency parse tree depends on the given words, Part Of Speech (POS) tags, and the semantic meaning of given phrases. Traditionally, words representation depends on technique called "one-hot vector" in which "1" is added in a unique location for the word and "0" in the remaining vector bits. Enormous drawbacks are faced in this representation, because it requires huge vectors for the handled vocabulary, as well as treating all words as independent entities. "One-hot vector" is not the ideal representation technique [8], because parse tree depends also on the semantic meaning and the context of words.

Other techniques are developed to overcome the mentioned complexities based on the use of pretrained embedding vectors, as additional features to the model. One of the most common embedding word vectors known is "word embeddings." In "word embeddings," each word is linked to a vector representation in a way that captures the semantic meaning of relationships of words. This is used as in transition-based dependency parsing, which defines concept of buffer to denote the contextual meaning of the words [10].

GloVe (Global Vectors for Word Representation) [6] is one of the well-known deep learning–based techniques, used for generating word vector representations. It is an unsupervised method for generating word representations, based on the word occurrences statistics in a given context. GloVe is proven that it outperforms other models on word similarity tasks.

Using precalculated word vectors in a deep learning solution to generate constituency parse tree is proven to be efficient and accurate [9]. The precalculated word vectors, called pretrained, are used as initial values and additional features, for a network within the model neural network. Such approaches depend on the input dataset, making it extensible and applicable to any language, to calculate both the unique vocabulary in the given dataset, as well as the word vectors representation. These pertained values are injected then to the main model which is used to generate the constituency parse tree. As the dataset for generating the parse tree could be different from the dataset used to calculate the word vectors representations, these vectors could be fine-tuned during the training of the constituency parse tree model, using the dataset for that model. In other words, the whole model network could allow these word vectors to be changed, according to the dataset of the given task.

In this chapter, GloVe is chosen to obtain the dense word representations. GloVe model is trained using Arabic resources, such as Arabic Corpus, Arabic Wikipedia articles, Arabic tweets, and Elwatan newspaper. These resources result into a huge number of words (1.9 billion words), in addition to comprising large number of unique Arabic vocabulary (1.5 million unique vocabulary). As this dataset contains numerous collection of unique words, more parameters are required to be learned. Therefore, the model is more prone to overfitting, which is one of the known problems in Arabic language called lexical sparsity. To overcome sparsity problem, a large amount of tokens or words is required.

GloVe model is modified in order to handle Arabic characters and overcome some encoding problems. After that, the modified model is used to calculate word vectors representations from the dataset mentioned previously.

GloVe model is trained using the mentioned dataset for 20 epochs, which results into getting the similarity between Arabic words. The model is slightly changed in order to handle the non-Latin nature of Arabic language. Changes mainly take place in encoding and memory management for the Arabic corpus.

The obtained Arabic vocabulary and Arabic words vectors, after the training process, are used within the constituency parse tree generator model network (as illustrated in the next section). The model itself fine-tunes these word representations according to the new dataset used to train the constituency parse tree generator model. This is done using RNN network, which inputs the word representation vector (256d vector) as initial values resulting in obtaining the best-predicted parse tree from the model faster and more accurate (Fig. 2).

4 Parse Tree Generator Model

The model used to generate the constituency parse tree depends mainly on syntactic distance [6]. To explain the meaning of a syntactic distance, the top-down scheme for construction the parse tree from a sentence needs more elaboration. One can process the sentence by splitting larger constituent into smaller constituents, in which the order of the split points defines the hierarchical structure. Hence, the syntactic distance is defined for the split points. The model uses neural networks to estimate the vector of syntactic distances for a given sentence. Figure 2 shows an overview of the model.

As mentioned previously, Arabic GloVe with dimension of 256 is used for Arabic word embeddings. The model fine-tunes word vectors. Not only the words are represented as dense vectors, but also tags, which helps in increasing the accuracy of the predicted parse trees. The result of these embeddings (fine-tuned) is fed to

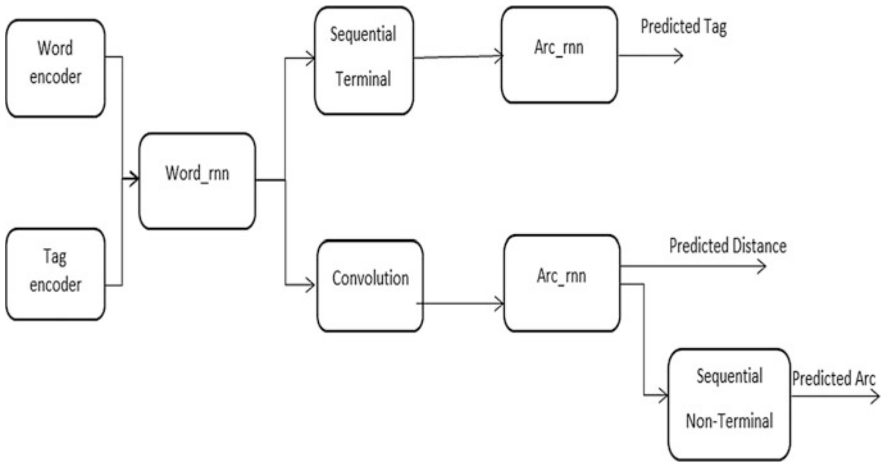

Fig. 2 Constituency parse tree generator overview

a BiLSTM (Bidirectional Long Short-Term Memory) word RNN. This makes the model captures a long-term syntactical relations between the words. Predicted tags could be calculated using feed forward network. Another bidirectional long short-term memory network called "arc_rnn" is used to calculate the syntactic distances. Constituent labels are then generated using the mentioned "arc_rnn," in addition to feed forward layer.

The model is fully parallel allowing it to be performed using a GPU. The model computes the distances in batch, and afterwards, it is decoded using an algorithm with complexity time of O (n log n). Moreover, the encoding and decoding algorithms are naturally adopted for execution in parallel environment, which can further reduce the time to O (log n). The estimated distance for each split point in the sentence is calculated independently from the others, allowing efficient parallelism.

For Arabic dataset, the model is slightly changed to handle the new dimensionality of dense Arabic words representations obtained from GloVe. Moreover, the model is now modified so as to handle non-Latin nature of Arabic language. The model is performant and is able to predict constituency parse tree even for very long Arabic sentences fast.

5 Workflow

In this chapter, a complete workflow is built upon the constituency parse tree model. The workflow typically enables linguists to predict the parse tree of a sentence from International Corpus of Arabic (ICA) and then they can edit the resultant tree. Moreover, these modified sentences could be used later to retrain the model. This section could be divided into two parts for explaining the workflow and how the dataset for retraining the model is extracted.

5.1 Workflow

The workflow consists of a series of stages. First, a list of sentences is shown to linguists using a web portal application. Linguists, users for this system, have accounts in this web portal. There are three main types of users to this web portal: a linguist, editorial reviewer (who is a more experienced linguist), and scientific reviewer.

Firstly, linguists need to be authenticated to be able to log in through the portal and once logged-in, a list of sentences is shown to the linguists by an administrator. The linguist selects the desired sentence in order to generate its constituency parse tree and then the sentence is sent to the model, pretrained with Elnahar data, so as to predict the constituent of this sentence.

Constituency parse tree is shown to the linguist in a viewer that enables him to modify the resultant tree, if and only if the final order of words remains the same as the given one. In brief, the linguist could edit a constituent label (nonterminal label, predicted by the model), modify the order of the parse tree nodes, or even delete a node. After amending the needed changes, the linguist could save the parse tree. As previously mentioned, the modified parse tree will be kept only on condition that the order of the sentence words is still the same as the input sentence.

Progressively, a new group of modified parse trees, which are curated by group of specialists (linguist, editorial, or even managerial layer), will be obtained. Since these sentences are verified, they will be used for model's incremental training. Incremental training means retraining the model using the new verified sentences. Initially, the saved model weights, obtained from previous training times, are loaded, followed by feeding the verified sentences to the model as a new dataset to be trained with. Thereafter, the model is trained, not only with Elnahar dataset, but also by the curated sentences resulting in the creation of a trusted model, which can generate and Arabic constituency tree efficiently. Figure 3 exhibits the workflow steps.

As a conclusion, the workflow can be easily operated and deployed in any hardware configuration. The main challenge tackled by this workflow, is how linguists could alter a deep learning model in order to benefit from it. The linguist uses the model to generate an initial constituency parse tree, which he could either accept or modify. In case of modifications, as long as it is curated, considerations are made to obtain a more accurate model. The resultant model is not only trained on the original Arabic dataset given (Elnahar) with limited number of sentences, but also a list of curated sentences obtained from a trusted dataset of International Corpus of Arabic (ICA).

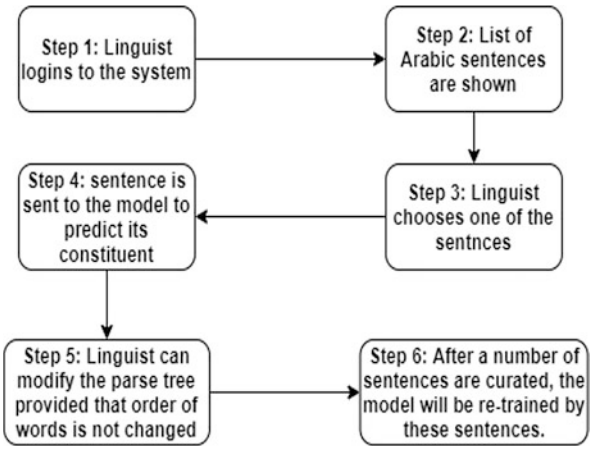
Step 1: Linguist logins to the system
Step 2: List of Arabic sentences are shown
Step 3: Linguist chooses one of the sentnces
Step 4: sentence is sent to the model to predict its constituent
Step 5: Linguist can modify the parse tree provided that order of words is not changed
Step 6: After a number of sentences are curated, the model will be re-trained by these sentences.

Fig. 3 Workflow

5.2 Dataset

As mentioned previously, linguists choose sentence from a list of Arabic sentences in order to predict its constituency parse tree and do the needed modifications. These sentences are extracted from International Corpus of Arabic (ICA) [13], using a script to build the parenthesized pair of word and Part Of Speech (POS) tag. This corpus is initiated by Bibliotheca Alexandrina (BA), in order to fulfill research demands on Arabic language. ICA currently contains 100 million Arabic words from four main sources; press, net articles, books, and academics. Press source, previously mentioned, comprises newspapers, magazines, and electronic press (Fig. 3).

ICA not only covers these sources but also it has numerous genres, such as strategic sciences, sports, religions, and many others. These genres are divided into 24 subgenres such as economy, law, politics, and others. Moreover, these subgenres are divided into four main sub-subgenres, namely short stories, novels, children's stories, and plays. The effect of these sources or genres to the Arabic corpus is not equal. It depends on how common this source or genre is. Hence, balance is not measured by the amount of texts.

Most of these sources and genres are got from all over the Arab world. Egypt covered about 13 million words out of the 100 million words of the corpus. Saudi Arabia comes in the second place after Egypt covering about eight million words. Other countries like Kuwait and UAE covered about five million words. Not only Arab world contributes to these words, but also some countries outside the Arab region.

The International Corpus of Arabic analysis is achieved automatically, using rule-based and statistical approaches, in which BASMA [14] Arabic morphological analyzer is used. This analysis lists information like list of prefixes, suffixes, word class, stem, lemma, root along with the number, gender, and definiteness of the word, depending on the context of this word within the sentence in the corpus.

6 Experiments

The constituency parse tree generator model, along with pretrained Arabic GloVe, is trained on Arabic dataset of Elnahar, which contains more than 12.5 K sentences, varying from short to very long ones, with varying data contexts between politics, sport, health, and others. Three different (unique) sets are created from the previously mentioned dataset: training, validation, and testing. Training data contains 10 K statements, in different context, with varying lengths, while validation data comprises 1 K sentences, varying between long and short sentences.

The validation dataset is utilized to address overfitting problem, which occurs when the model tightly understands the training data, but fails in understanding and generating parse trees for data with either different vocabulary, lengths, or context, thereby allowing the parse tree generator model to self-correct itself. Finally, the test dataset contains more than 1.6 K sentences, with similar characteristics to both training and validation datasets.

The validation and testing datasets are chosen in a way, such that they are different and unbiased. In order to validate this claim, the model is used to generate the constituency parse trees for both of validation and test datasets. The model outputs almost the same result on both the validation and test datasets (F-measure = 86%).

In order to evaluate the accuracy of the model, F-measure [15] is used. F-measure could be considered a ratio of precision and recall; therefore, it captures efficiently both of them during the calculations. Standard Evalb tool, in its Python version, is used for this evaluation. This version, after some modifications in this work, assists in creating a cross-environment model, as well as helping in solving the encoding problems encountered, due to complexities in Arabic language (non-Latin nature).

F-measure is calculated by:

$$\mathrm{F-measure} = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

where

- Precision is considered the positive predictive value. It is calculated as follows:

$$\mathrm{Precision} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}}$$

  - tp is truly predicted result.
  - fp is a false alarm (predicted to be true, though it is false – Type I error).

- Recall is also called sensitivity.

$$\text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

  – tp is truly predicted result.
  – fn (predicted to be false, though it is true – Type II error).

Experiments are carried out on Bibliotheca Alexandrina (BA) High Performance Computing (HPC) cluster, which provides both CPU and GPU capabilities. We choose to use the GPU unit to achieve the results from the model faster since the problem can be handled in parallel nature. After the training process of the constituency parse tree generator model is complete, the model can run in any environment, with in any hardware configurations.

Labels, nonterminal tags, are eliminated by grouping similar labels into one label. For example, "NOUN.VN + NSUFF_FEM_SG + CASE_DEF_GEN," is reduced to the POS "NOUN" tag. In this work, number of labels is reduced from around 600 labels to only 20 labels. Moreover, POS tags are reduced from around 400 tags to only 38. This reduction minimizes the number of labels to be predicted, resulting in a lower dimensionality, hence more accurate model.

In this chapter, four main experiments are evaluated, considering the length of the sentence, in addition to the different split points in it (Al-taareef position), as shown below. The result of them proves that the length of the sentence along with the split points does not have strong effect on the final result. These experiments show that the usage of word embeddings technique (GloVe) has a very important effect to the results and helps into getting the final result fast and more accurate.

The following shows the results obtained from the experiments.

6.1 Short Sentences with Minimum Split Points

Elnahar dataset contains very long lines, forming multiple sentences. In this experiment, very long sentences are split, if there is no direct association with the original sentence, which can occur in case of opening parenthesis while the closing one is not located at the end of it. Therefore, these types of sentences are split into multiple ones, according to the opening and closing parenthesis. Moreover, prefix such as Al-taareef (the) is not split from the word, to minimize the split points.

From the table below, it is noticed that at the first loop, F-measure was 70% for the testing dataset, which is better than the corresponding one without GloVe. As a result, this assures that using word embeddings techniques, along with the neural network approaches, for constituency parse tree generators could affect the accuracy (F-measure) of the model, especially at the beginning of training iterations (epochs). In addition, word embeddings techniques help in minimizing the number of epochs needed, in order to reach the final accuracy value, hence minimizing processing cost. Also, it is noticed that the F-measure rate increased rapidly for the first five epochs as shown in Table 1, to slow down until reaching the best accuracy (F-measure = 86%).

The following table shows the number of epochs needed, in order to achieve different percentages from the final result.

6.2 Short Sentences with Maximum Split Points

This experiment is similar to the previous one, however the only difference is that Al-taareef (the) is separated from the word, thus creating more split points to be used in the model measurements for distance and constituent.

It is noticed that the results are very similar to what we obtained from the previous experiment, concluding that adding or removing the Al-taareef (the) in short sentences has almost no difference noticed. The following Table 2 shows the number of epochs needed in order to achieve different percentages from the final result. It can be concluded from the table that the rate in which the model achieves 95% from the final result is almost the same between the current experiment and the previous one.

6.3 Long Sentences with Minimum Split Points

This experiment maintains long sentences as they are given in their original dataset without splitting, leading into very long sentences, but this time, Al-taareef (the) is not split from the original word. Consequently, one line, which might contain multiple sentences, is fed as a single entity to the model for training, validation, and testing.

Table 3 indicates the results obtained from this experiment. As observed, we can achieve 95% from the best accuracy obtained (F-measure = 86%) rapidly, when compared to the previous two experiments. This means that the length of the Arabic sentence does not cause an effect to the final results nor when these results are achieved.

Table 1 F-measure percentage from the best result

F-measure *(percentage from the final result which is 86%)*	81%	85%	90%	95%
Epochs	1	2	4	12

Table 2 F-measure percentage from the best result

F-measure *(percentage from the final result which is 86%)*	81%	85%	90%	95%
Epochs	1	3	4	13

Table 3 F-measure percentage from the best result

F-measure *(percentage from the final result which is 86%)*	82%	85%	90%	95%
Epochs	1	2	4	10

Table 4 F-measure percentage from the best result

F-measure *(percentage from the final result which is 86%)*	77%	85%	90%	95%
Epochs	1	2	5	11

6.4 Long Sentences with Maximum Split Points

This experiment is similar to the last one, yet adding the Al-taareef (the) to the words. In Table 4, which illustrates the obtained results, we can note that the result achieves 95% from the best accuracy (F-measure = 86%) in almost the same manner as the previous experiment. However, currently the first epoch shows that the starting accuracy is less than its corresponding one, from experiment number 3.

7 Conclusion

Constituency parse tree is the backbone of many Natural Language Processing (NLP) tasks, such as language translation and Arabic diacritization. This is why valuable research efforts have been exerted in order to find an algorithm which generates parse tree efficiently. Deep learning approaches are adopted, because they are efficient and can generate the parse tree using dataset without any predefined rules. In this chapter, deep leaning approaches are used to generate constituency parse tree for Arabic sentences. First, dense words representations are accomplished by GloVe, which is trained using various Arabic resources with billions of words for multiple epochs. Afterwards, this dense representation is fed into deep learning technique, which is responsible for generating parse trees for Arabic sentences. This model is trained initially by Elnahar dataset. This dataset contains more than 12 K Arabic sentences with varying data contexts between politics, sport, health, and others. This is considered a step in a complete workflow, which enables linguists to choose sentences and get the corresponding parse trees. The Arabic sentences are extracted from the International Corpus of Arabic (ICA) initiated by Bibliotheca Alexandrina, which contains 100 million words from multiple sources and genres. Linguists could accept the parse trees predicted or modify them, by editing the constituent, reorder the resultant constituent, or deleting nodes, provided that modified words order in the sentence remains the same as the input one. Finally, these sentences are used to retrain the model to obtain better results eventually and enrich the dataset. The used model is fully parallel and capable of predicting constituency parse tree for long sentences. High-performance computers could be used to facilitate the training process and cut down the training time.

References

1. C. Callison-Burch, Syntactic constraints on paraphrases extracted from parallel corpora, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (NLP)*, (2008), pp. 196–205
2. J.C. Maxwell, *A Treatise on Electricity and Magnetism*, vol Vol. 2, 3rd edn. (Clarendon, Oxford, 1892), pp. 68–73
3. A. Eriguchi, Y. Tsuruoka, K. Cho, Learning to parse and translate improves neural machine translation, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, (2017), pp. 72–78
4. O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, G. Hinton, Grammar as a foreign language, in *In Advances in Neural Information Processing Systems*, (2015), pp. 2773–2781
5. D. Chen, C.D. Manning, *A Fast and Accurate Dependency Parser Using Neural Networks* (Association for Computational Linguistics, USA, 2015), pp. 740–750
6. Y. Shen, Z. Lin, A. Paul Jacob, A. Sordoni, A. Courville, Y. Bengio, Straight to the Tree: Constituency Parsing with Neural Syntactic Distance. arXiv:1806.04168, 2018
7. A. Farghaly, K. Shaalan, Arabic natural language processing, in *ACM Transactions on Asian Language Information Processing*, (2009)
8. J. Pennington, R. Socher, C.D. Manning, GloVe: Global vectors for word representation, in *EMNLP*, (2014)
9. D. Vilares, M. Strzyz, A. Søgaard, C. Gomez-Rodrıguez, Parsing as Pretraining, arXiv: 2002.01685v1, 2020
10. C. Dyer, M. Ballesteros, W. Ling, A. Matthews, N.A. Smith, Transition-based dependency parsing with stack long short-term memory, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (2015), pp. 334–343
11. A. Bakar, O. Khairuddin, M. Faidzul, M. Zamrim, Implementation of Buckwalter transliteration to Malay corpora, in *International Conference on Intelligent Systems Design and Applications, ISDA ER*, (2013)
12. J. Cross, L. Huang, Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Pp 1–âA¸S11*, (2016)
13. S. Alansary, M. Nagi, The International Corpus of Arabic: Compilation, Analysis and Evaluation. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pp 8–17, October 25, 2014, Doha, Qatar
14. S. Alansary, BASMA: BibAlex Standard Arabic Morphological Analyzer. In *The 14th Egyptian Society of Language Engineering Conference*, December 2015, Cairo, Egypt
15. L. Derczynski, Complementarity, F-score, and NLP Evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 261–266, 2016