ISSN 2007-9737

Comparative Study for Text Chunking Using Deep Learning: Case of Modern Standard Arabic

Nabil Khoufi[1,*], Chafik Aloulou[2]

[1] University of Sfax, ANLP Research Group, Sfax, Tunisia

[2] University of Sfax, MIRACL Lab, Sfax, Tunisia

nabil.khoufi@outlook.com, chafik.aloulou@fsegs.usf.tne

Abstract. The task of chunking involves dividing a sentence into smaller phrases by identifying a limited amount of syntactic information. This process involves grouping together consecutive words to form phrases, also known as shallow parsing. Chunking does not provide information on the relationships between these phrases. This paper describes our approach to building chunking models for Arabic text using deep learning techniques. We evaluated several training models and compared their results using a rich data set. The results we obtained were highly encouraging when compared to previous related studies.

Keywords. NLP, Arabic language, shallow parsing, chunking, deep learning, GRU, LSTM, BILSTM, ATB, Penn Arabic treebank.

1 Introduction

Text parsing is a critical aspect of natural language processing (NLP) and has received significant attention since the early days of NLP. The information generated by parsing is valuable for various NLP applications, such as automatic summarization, author profiling and named entity recognition [1]. Parsing can either be shallow or deep. Shallow parsing, also known as chunking, focuses on identifying the boundaries of larger constituents or phrases, while deep parsing goes further by identifying both the constituents and their internal structure. The two types of parsing require different amounts of information and produce different results.

The chunking task can be approached through two main methods: the grammar-based approach and the machine learning approach [2] The former uses a set of grammatical rules, while the latter employs machine learning techniques and relies on annotated data. This paper details our experiments on chunking Arabic text using various deep learning architectures. The structure of this paper is as follows:

In Section 2, we outline the fundamental concepts of the chunking task. Section 3 discusses the syntactic ambiguities in Arabic. Section 4 reviews prior research on chunking in Arabic. In Section 5, we describe our deep learning models and approach to chunking Arabic text. Section 6 presents the evaluation process and results obtained. Finally, in Section 7, we present our conclusions and suggest avenues for future research.

2 Chunking Task Background

In 1991, Steven Abney proposed an approach to parsing that involves identifying groups of words that are syntactically correlated [3]. He argued that when we read, we do so in chunks, and therefore chunking involves dividing a sentence into smaller parts that are syntactically related. Chunking can be seen as an intermediate step towards full parsing as it provides part of the complete syntactic structure of a sentence.

Initially, the chunking task focused on recognizing noun phrases (NPs), which is known as noun phrase chunking. Lance Ramshaw and Mitch Marcus tackled NP-chunking using a

ISSN 2007-9737

machine learning method [4]. They recognized various chunks but categorized every chunk that was not a NP as a VP chunk.

This work inspired many other studies that have investigated the application of learning methods to noun phrase chunking. Later, researchers focused on other constituents of sentences such as VP, PP, ADJP, or ADVP to provide a more comprehensive description of the sentence. The following example shows a chunked Arabic sentence [5].

3 Sources of Ambiguity in Arabic Language

Arabic, like all Semitic languages, has a complex morphology and a vast vocabulary, making it more challenging to parse than other natural languages [6]. In addition to common linguistic features like coordination, anaphora, and ellipsis found in Latin-based languages, Arabic has unique characteristics that pose difficulties in the parsing process [7].

3.1 Unvocalisation

The lack of vowels in written words, known as unvocalization, leads to grammatical ambiguity. Words without vowels in their written representation can't effectively differentiate between different grammatical interpretations and meanings, as a single word can have multiple grammatical variations.

As a result, unvocalized text is more ambiguous than text with vowels [8, 9]. According to Debili's statistics (Debili et al., 2002), 74% of Arabic words have more than one vocalization option.

The ambiguity rate for grammatical interpretation is higher in unvocalized words, with an average rate of 8.7, compared to an average rate of 5.6 for vocalized words. Table 1 provides an example of a single unvocalized word with its various vocalized forms.

3.2 Agglutination

In Arabic, there is a distinct occurrence known as agglutination, where words such as articles, prepositions, pronouns, etc. can be attached to

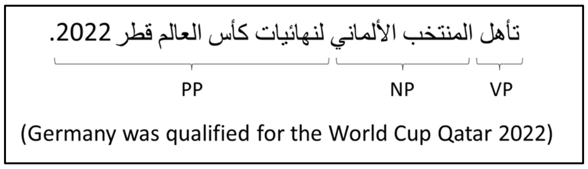

تأهل المنتخب الألماني لنهائيات كأس العالم قطر 2022.
PP
NP
VP
(Germany was qualified for the World Cup Qatar 2022)

Fig. 1. An instance of chunking in modern standard Arabic

adjectives, nouns, verbs, and particles that they are associated with.

This leads to complex syntax, resulting in unusual sentence structures. In fact, an agglutinative form can even make up an entire sentence, as demonstrated in table 2. In such instances, specific processing is necessary to determine the correct syntactic structure of the sentence.

3.3 Words Order

The arrangement of words in Arabic is flexible. Typically, the word that is intended to be emphasized is placed at the beginning of the sentence and the word with the most meaning or tone is placed at the end. This freedom in word order results in artificial syntactic confusion and makes constructing grammar more difficult.

To account for all possible correct word arrangements in a sentence, grammar rules must include all combinations. Table 3 demonstrates the impact of changing the order of words. The order of words in this sentence can be rearranged, resulting in the two structures shown in Table 4 and 5.

3.4 Recursive Structure

The frequent use of recursive structures is another characteristic of Arabic texts. The presence of nested structures is common in Arabic as well as in other natural languages, but it occurs more frequently in Arabic due to some propositions being able to play a role within other propositions. For example:

الشرطة هي التي قبضت على المجرم الذي ضل هارباً مدة طويلة.

(The police have arrested the criminal who remained on the run for a long time).

Table 1. An illustration of ambiguity due to the unvocalization phenomenon

Unvocalized Word	Vocalized Forms	Buckwalter Transliteration	Translation
فهم	فَهِمَ	fahima	He understood
	فَهَّمَ	fah~ama	He explained
	فُهِمَ	fuhima	It has been understood
	فَهْمٌ	fahomN	Comprehension
	فَهُمْ	fahumo	Then them
	فَهَمَّ	faham~a	Then started
	…	…	…

Table 2. A sample of a sentence in an agglutinative form (one word)

Sentence	Buckwalter transliteration	Gloss
واستقبلهم	wastaqbalahum	(Then he welcomed them).

Table 3. Arabic sentence, order 1

Arabic sentence	إلى بعلملا	الولد	ذهب
English translation	to the stadium	the boy	went
Form	complement	subject	verb

Table 4. Arabic sentence, order 2

Arabic sentence	إلى بعلملا	ذهب	الولد
English translation	to the stadium	went	the boy
Form	complement	verb	subject

Table 5. Arabic sentence, order 3

Arabic sentence	.الولد	ذهب	إلى بعلملا
English translation	the boy	went	to the stadium
Form	subject	verb	complement

It is a nominal sentence, while the proposition (خبر) is also a nominal sentence:

هي التي قبضت على المجرم الذي ضل هارباً مدة طويلة

(Have arrested the criminal who remained on the run for a long time).

In the aforementioned example, it is even challenging to segment the text into sentences because of the numerous propositions that are interdependent and do not belong to the same syntactic level. This lack of independence leads to Arabic sentences being of unlimited length.

4 Related Works

Compared to the research conducted in English and other languages, there is a scarcity of studies on the Arabic chunking problem. Only four works can be found that specifically address the Arabic chunking task. In 2004, Mona Diab and colleagues [11, 12] carried out tokenization, POS tagging, and used an SVM-based method for Arabic text chunking.

They utilized an existing SVM tool [13]. The features used in their system were words and POS annotations, along with a context window of -2/+2. The system was evaluated on 400 sentences and produced a chunking performance of 92.06% precision, 92.09% recall, and 92.08 F-measure. This research was the first of its kind. Diab later used the same SVM tool and trained their model using the Arabic Treebank with a modified POS tag set [12].

They reported an F-measure chunking performance of 96.33%. Mohammed and Omar

Table 6. Comparative summary of related works

Works	Approach	Training Data	Testing data	Results
[12]	Machine learning-Based	18 970 sentences	2337 sentences	F-measure 91.44%
[14]	Grammar-Based	-	70 sentences	F-measure 97%
[15]	Machine learning-Based	2300 words	283 sentences	Accuracy 80.46%
[16]	Machine learning-Based	10 100 sentences	2524 sentences	Accuracy 96.54% F-measure 76,23% Recall 73,86% Precision 81,57%

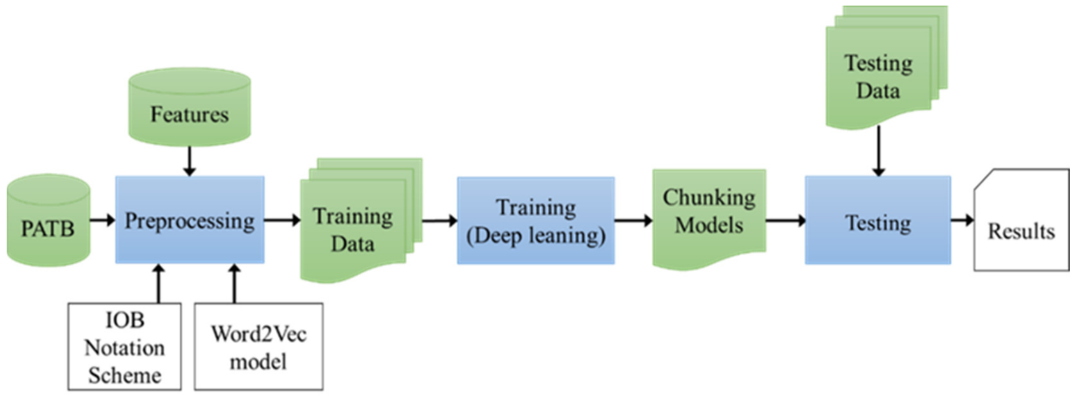


Fig. 2. Proposed method architecture

Table 7. Features list

Feature	Description
w[t]	the word being proceeded
w[t+1]	the word on right vicinity at position t+1
w[t+2]	the word on right vicinity at position t+2
pos[t]	the POS annotation of w[t]
pos[t+1]	the POS annotation of w[t+1]
pos[t+2]	the POS annotation of w[t+2]

[14] describe the development of an Arabic shallow parser based on a rule-based approach.

The chunking which constitutes the main contribution are achieved on two successive stages that include grouped sequences of adjacent words based on linguistic properties to identify each of NPs, VPs and PPs. Since the aim of the research is to generate results at two levels, the final results adopted were based on the second level results.

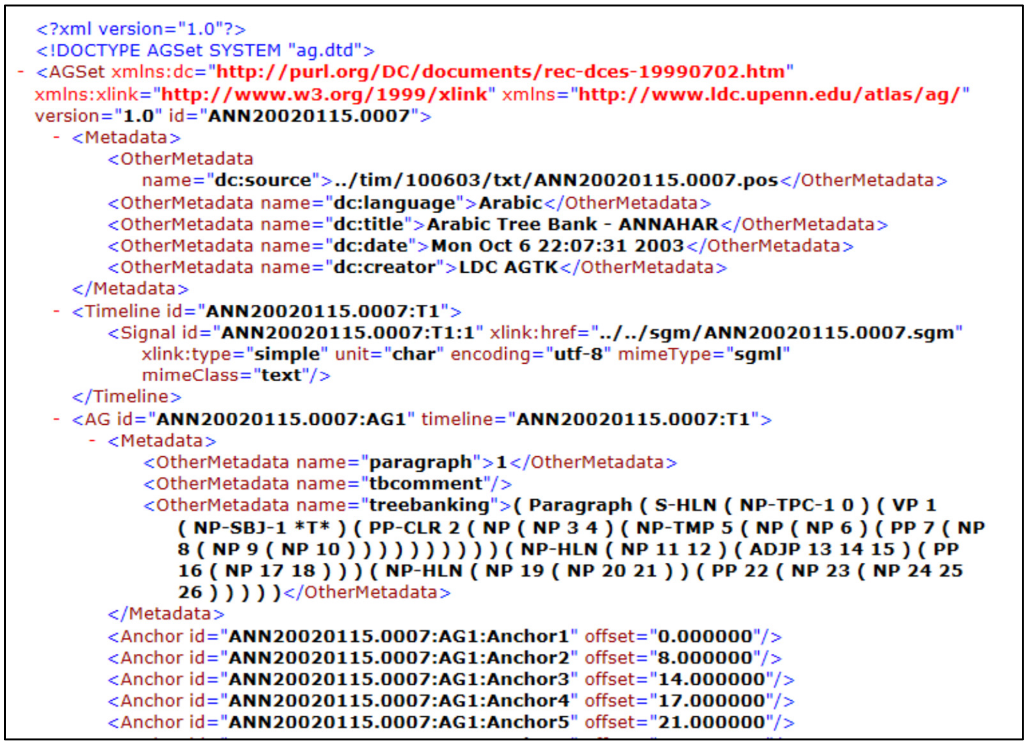
Tested on only 70 sentences, their system achieved F-measure of 97%. Ben-Fraj and Kessentini [15] proposed an approach for chunking Arabic texts based on a combinatorial classification process.

It is a modular chunker that identifies the chunk heads using a combinatorial binary classification before recognizing their types based on the parts-of-speech (POS) of the chunk heads, already identified.

For the experimentation, the authors used 226 sentences as training data. They obtained 80.46% accuracy for the full chunking process.

Table 8. An instance of an annotated sentence based on the IOB notation model

Sentence	.	2016	العربية	الثقافة	عاصمة	صفاقس
Transliteration	.	2016	AlErbyp	AlvqAfp	EASmp	SfAqs
Traduction	.	2016	Arab	Culture	Capital	Sfax
Annotation IOB	O	I-NP	I-NP	I-NP	B-NP	B-NP



```
  <?xml version="1.0"?>
  <!DOCTYPE AGSet SYSTEM "ag.dtd">
- <AGSet xmlns:dc="http://purl.org/DC/documents/rec-dces-19990702.htm"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.ldc.upenn.edu/atlas/ag/"
  version="1.0" id="ANN20020115.0007">
  - <Metadata>
      <OtherMetadata
        name="dc:source">../tim/100603/txt/ANN20020115.0007.pos</OtherMetadata>
      <OtherMetadata name="dc:language">Arabic</OtherMetadata>
      <OtherMetadata name="dc:title">Arabic Tree Bank - ANNAHAR</OtherMetadata>
      <OtherMetadata name="dc:date">Mon Oct 6 22:07:31 2003</OtherMetadata>
      <OtherMetadata name="dc:creator">LDC AGTK</OtherMetadata>
    </Metadata>
  - <Timeline id="ANN20020115.0007:T1">
      <Signal id="ANN20020115.0007:T1:1" xlink:href="../../sgm/ANN20020115.0007.sgm"
        xlink:type="simple" unit="char" encoding="utf-8" mimeType="sgml"
        mimeClass="text"/>
    </Timeline>
  - <AG id="ANN20020115.0007:AG1" timeline="ANN20020115.0007:T1">
    - <Metadata>
        <OtherMetadata name="paragraph">1</OtherMetadata>
        <OtherMetadata name="tbcomment"/>
        <OtherMetadata name="treebanking">( Paragraph ( S-HLN ( NP-TPC-1 0 ) ( VP 1
          ( NP-SBJ-1 *T* ) ( PP-CLR 2 ( NP ( NP 3 4 ) ( NP-TMP 5 ( NP ( NP 6 ) ( PP 7 ( NP
          8 ( NP 9 ( NP 10 ) ) ) ) ) ) ) ) ) ) ) ( NP-HLN ( NP 11 12 ) ( ADJP 13 14 15 ) ( PP
          16 ( NP 17 18 ) ) ) ( NP-HLN ( NP 19 ( NP 20 21 ) ) ( PP 22 ( NP 23 ( NP 24 25
          26 ) ) ) ) )</OtherMetadata>
      </Metadata>
      <Anchor id="ANN20020115.0007:AG1:Anchor1" offset="0.000000"/>
      <Anchor id="ANN20020115.0007:AG1:Anchor2" offset="8.000000"/>
      <Anchor id="ANN20020115.0007:AG1:Anchor3" offset="14.000000"/>
      <Anchor id="ANN20020115.0007:AG1:Anchor4" offset="17.000000"/>
      <Anchor id="ANN20020115.0007:AG1:Anchor5" offset="21.000000"/>
```

Fig. 3. A segment of the training corpus in its original XML tree format

Khoufi et al. developed a machine-learning based model for Arabic text chunking [16]. They utilized CRFs (Conditional Random Fields) for training the model.

The training data was derived from the PATB syntactic trees (80% of the ATB) and included words and their part of speech annotations as features, with a context window of -2/+2 words centered around the word being processed.

The model also considered bigrams and trigrams of words and POS annotations.

The model was tested on 20% of the PATB corpus, representing 2524 sentences, and achieved an accuracy of 96.54%.

A summary of their study of Arabic chunkers is presented in Table 6.

5 Proposed Method

This section presents the design of our proposed method. Our approach is based on deep learning technology for chunking Arabic text, as opposed to a rule-based method.

This is because constructing a grammar that encompasses all the exceptional and specific syntactic structures in Arabic is highly challenging, if not impossible. Our method involves using an annotated corpus, a set of features, and a model, which will be discussed in the subsequent section.

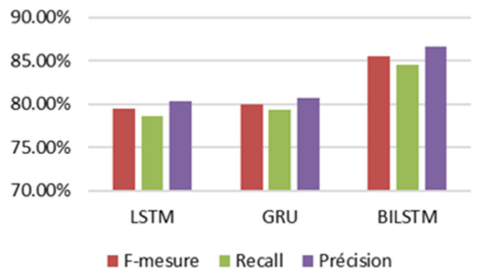


Fig. 4. Evaluation results comparison

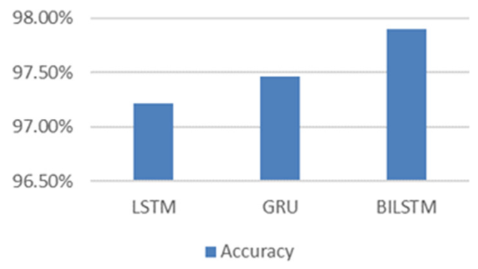


Fig. 5. Accuracy results

Figure 2 illustrates the design of our approach, where a portion of the annotated corpus (80%) undergoes pre-processing using features, the Word2Vec model, and the IOB notation scheme.

This generates a model that is used to analyze sentences. The model's performance is then evaluated using the remaining portion of the annotated corpus (20%) through cross-validation. Detailed information regarding our method can be found in subsequent sub-sections.

5.1 Our Features

The selection of features is crucial in machine-learning algorithms as it has a significant impact on the accuracy of labeling. Features determine what information is extracted from the annotated corpus during the training phase. In this study, we chose to use the word itself (W) and its Part-of-Speech (POS) annotation as our features.

These features capture the characteristics of the word at position t by utilizing information from the surrounding words. The features utilized for the training phase are shown in Table 7.

5.2 Used Tag Set

For our experiment, we employ the tag set of the PATB, which consists of 23 tags: S, NP, VP, SQ, PP, SBAR, SBARQ, NX, PRN, PRT, QP, ADJP, ADVP, FRAG, WHNP, WHPP, WHADJP, WHADVP, CONJP, INTJ, NAC, UCP, X.

In addition to these tags, we use the IOB notation model, where each word is tagged with a chunk label and one of three additional tags:

– B for the first word of a chunk,
– I for a non-initial word in a chunk,
– and O for a word outside of any chunk.

This increases the number of tags in the tag set to 46, as each tag in the tag set becomes either an I or B tag. For example, NP can be represented as two chunk types, I-NP or B-NP. Table 8 provides an example of an Arabic sentence tagged using the IOB scheme, with the English Translation shown in the right-to-left direction.

5.3 Training Corpus

The Penn Arabic Treebank was established by the Linguistic Data Consortium (LDC) at the University of Pennsylvania [17]. It is composed of data obtained from standard and modern Arabic linguistic sources, consisting of 402,291 tokens and 12,624 sentences.

The texts in the corpus do not contain any vowels, as is typical in most written Arabic texts. In our experiments, we used version 3.2 of this corpus. In order to perform the training phase, the training corpus must be pre-processed to incorporate the IOB notation scheme and the selected features along with their context window.

This pre-processing step transforms the PATB corpus from its original tree format into a vector format. Figure 3 displays the original format of the training corpus.

5.4 Training Experiments

For the training stage, we utilize the Word2Vec model for constructing word embeddings. Word2Vec is a widely used method for building word embeddings and was first introduced by [18]. There are two variations of the Word2Vec model for learning word embeddings:

The Skip-gram and CBOW (Continuous Bag of Words) models. Each of these models consists of three layers: an input layer, a hidden layer, and an

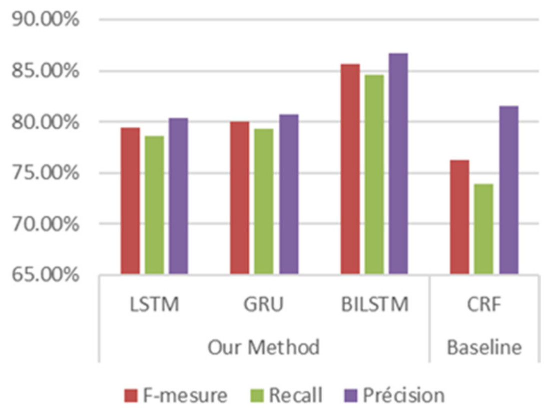


Fig. 6. Results comparation between our DL models and CRF model

output layer, with the output layer consisting of neurons with a SoftMax activation function.

- The CBOW architecture enables prediction of a word based on its context, using a word window to the left and right. This model assumes that the order of the context words has no effect on the projection, and the projection layer is shared among all words. Learning word embeddings with the CBOW architecture involves predicting a word based on its context, by calculating the vector obtained by summing the embeddings of the context words, and then applying a log-linear classifier to predict the target word.

- The Skip-gram architecture is also a three-layer log-linear neural network. Unlike the CBOW model, it allows for predicting a context window given the word at the center of the context. The central word serves as the input to the network, and the words in the context form the output. The goal is to predict, for a given word, its context, so that the embedding of any word is close to the embeddings of words in the same context.

In this study, we utilized the CBOW architecture for word embedding construction as it is a widely used method in NLP and has produced positive outcomes in previous NLP studies such as [19]. Today, Recurrent Neural Networks (RNNs) are the most commonly used systems in various machine learning tasks.

They are frequently utilized in computer vision (such as image classification, object detection, segmentation, etc.) and natural language processing (such as automatic translation, voice recognition, language models, etc.). In our Arabic chunking task, we experiment with three different RNN architectures (LSTM, BILSTM, GRU), which we present below, to compare their performance.

5.4.1 LSTM

Long Short-Term Memory (LSTM) is a deep learning-based RNN architecture that has feedback connections, unlike regular feedforward neural networks. LSTM can handle not just individual data points (such as images), but also sequential data (such as speech or video) [20].

LSTMs are designed to address the vanishing gradient problem that occurs in traditional RNNs, which can make it difficult for the network to capture long-term dependencies in sequential data. In an LSTM, the network has a hidden state that is updated at each time step.

The update is controlled by three gates: the input gate, the forget gate, and the output gate. The input gate determines how much of the new input should be added to the current hidden state, the forget gate determines how much of the previous hidden state should be forgotten, and the output gate determines how much of the new hidden state should be output.

Applications of LSTM include unsegmented handwriting recognition [21], speech recognition [22], and network traffic anomaly detection or intrusion detection systems (IDSs).

5.4.2 BILSTM

Bidirectional Long Short-Term Memory (BILSTM) is a type of recurrent neural network architecture. In a traditional LSTM, the input sequence is processed in one direction, from the beginning to the end. However, in a BILSTM, the input sequence is processed in two directions simultaneously: one forward and one backward.

The BILSTM consists of two LSTM layers, one processing the input sequence in the forward direction and the other in the backward direction. The outputs of both the layers are concatenated at each time step to form the final output of the BILSTM.

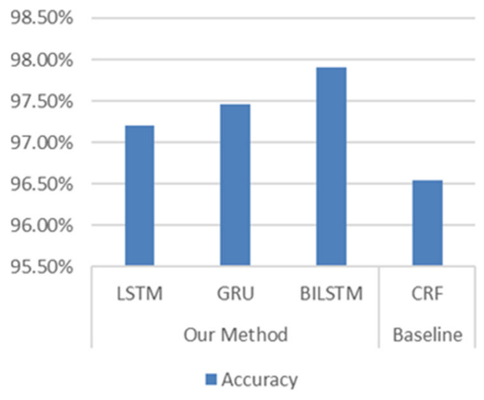


Fig. 7. Accuracy comparison between our DL models and our baseline

This allows the model to capture both the past and the future context of each input token, which can be useful in many sequence modelling tasks. The BILSTM has been widely used in various natural language processing tasks such as part-of-speech tagging, named entity recognition, sentiment analysis, and machine translation [23], among others. It has been shown to outperform traditional LSTM models in many of these tasks.

5.4.3 GRU

Gated Recurrent Units (GRUs) are gating mechanisms in RNNs, first introduced in 2014 by Kyunghyun Cho et al. [24]. GRUs are similar to LSTMs, but have a simpler structure with fewer parameters, making them faster to train and less prone to overfitting [25]. In a GRU, the network has a hidden state that is updated at each time step.

The update is controlled by two gates: the reset gate and the update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be added to the current hidden state.

The update gate and reset gate are both sigmoid functions that take as input the current input and the previous hidden state.

The output of these gates is then used to update the hidden state. Unlike LSTMs, which have separate memory cells, GRUs use a single hidden state to store information about the input sequence.

GRUs have shown comparable performance to LSTMs in tasks such as polyphonic music modelling, speech signal modelling, and natural language processing [26], and even better performance on smaller and less frequent datasets [27].

6 Evaluation Results and Discussion

In order to evaluate the models, we divided the PATB corpus into two parts, with 80% consisting of 10,100 sentences for training and 20% consisting of 2,524 sentences for testing.

The unvocalized version of the Treebank was used for all experiments and all the data was sourced from the parsed trees in the Treebank.

By using the unvocalized version of the Treebank, we ensure that our results are consistent and comparable with other studies that use the same corpus.

We measured the performance of the model's using precision, recall, F-measure and accuracy.

These metrics allows us to have a comprehensive view of the performance of the models and help us in choosing the best model for our Arabic chunking task. This evaluation method allows us to assess the ability of the models to correctly predict the chunk tags for the sentence chunks.

The results of these metrics are displayed in figure 4. When comparing the performance of the LSTM, GRU, and BILSTM models based on their precision, recall, and F-measure metrics, the results show that the BILSTM model achieved the highest precision (86.71%), recall (84.54%), and F-measure (85.59%).

This represents an enhancement of 6,37%, 5,94% and 6,13% respectively over the LSTM model's precision (80.34%), recall (78.60%), and F-measure (79.46%).

Similarly, the GRU model achieved an enhancement of 0.43%, 0.73%, and 0.63% respectively over the LSTM model's precision, recall, and F-measure by reaching a precision of 80,69%, a recall of 79,33% and an F-measure of 80%.

In addition, it's important to note that the BILSTM model achieved the highest performance in all three metrics, suggesting that it may be the most effective model overall.

Table 9. Chunking performance of BILSTM model calculated on major chunking tags

Tags	Precision	Recall	F-measure
NP	93,32%	95,55%	94,42%
VP	93,82%	94,11%	93,96%
PP	65,80%	65,94%	65,87%
ADJP	91,25%	86,83%	88,99%
ADVP	55,84%	57,41%	56,61%
CONJP	98,87%	98,83%	98,85%
PRT	95,82%	97,47%	96,64%
PRN	52,22%	48,38%	50,23%
O	96,67%	97,97%	97,32%
S	92,88%	96,87%	94,83%

This could be explained by the ability of BILSTM model to capture both past and future context when processing sequential data. BILSTM model results are confirmed by the accuracy metric.

Indeed, BILSTM model reached an accuracy of 97.90% slightly exceeding LSTM and GRU models, respectively 97.21% and 97.46%. Accuracy values comparison are displayed in the following figure 5. It also interesting to compare obtained results with previous work which we consider as our baseline.

We have developed a machine learning model for chunking Arabic using Conditional Random Fields (CRF) (Khoufi et al. 2015). We used the same data for training and testing the CRF model. Results comparisons are illustrated in the following Figure 6.

As shown in Figure 5, the LSTM, GRU, and BILSTM models outperformed the CRF model, which achieved an accuracy of 81.57%, a recall of 73.86%, and an F-measure of 76.23%. Indeed, among the DL-based models, BILSTM achieved the best results and significantly improved our baseline with a 5.14% increase in precision, a 10.68% increase in recall, and a 9.36% increase in F-measure.

Even the accuracy measurement confirms the superiority of the BILSTM model (97,90%) over the CRF model (96,54%), as demonstrated in Figure 7 below. For a detailed idea about the performance of our model, we calculated the precision, the recall and the f-measure for the major chunking tags.

These results are exposed in the following Table 9. In relation to the outcomes displayed in Table 9, our model has successfully identified significant portions with acceptable accuracy.

We want to emphasize that the model has achieved a commendable overall performance in identifying chunks, which validates the obtained results. As shown in Table 9, the CONJP category is recognized with the highest precision of 98.87%, recall of 99.47%, and f-measure of 98.85%.

This is reasonable due to the limited occurrence of conjunctions in Arabic. The model also performs well in recognizing PRTs, with a precision of 95.82%, recall of 97.47%, and f-measure of 96.64%. PRTs are typically associated with CONJP, which facilitates their identification.

However, we have observed some difficulties in detecting PP and ADVP chunks, with f-measures of 65.87% and 56.61%, respectively. PPs consist of a preposition followed by an object of preposition, such as NP, and this relationship with other chunks may make it challenging to determine their boundaries.

We attempted to compare our findings with those of other studies, but encountered difficulties in performing an accurate analysis because they used different metrics and datasets.

However, our accuracy value was higher than that of (Ben Fraj et al. 2012), with a 17% increase.

Also, the authors used only 2300 words as training data which is not sufficient to obtain a viable model.

We found that the testing data used by (Mohamed et al. 2011) was insufficient to provide an accurate assessment of system performance, as they only tested on 70 sentences compared to our model's 2,524 sentences. Despite this, (Diab et al. 2007) achieved a higher F-measure of 96.33% using a testing set of 2337 sentences compared to our F-Measure.

In summary of this study's results, it can be said that models based on deep learning techniques achieve good results in processing the chunking task of Arabic texts. These models, especially the BILSTM model, improved our results compared to the classical models that used traditional machine learning algorithms, such as the CRF model we used for our baseline.

7 Conclusion

In this study, we presented our approach for chunking Arabic texts through the use of deep learning models. The models we built are LSTM, BILSTM, and GRU, and they are constructed with morphosyntactic features and the IOB notation system.

The training data for the models was obtained from the PATB corpus. Our models were trained on 80% of the PATB and tested on the remaining 20%. Evaluation results shows the supremacy of BILSTM model for this task with an f-measure of 85.59% and an accuracy of 97.90%.

References

1. **Khoufi, N. (2017).** Une approche hybride pour l'analyse syntaxique de la langue arabe. Doctoral dissertation, Université de Sfax Tunisie.
2. **Khoufi, N., Aloulou, C., Belguith, L. H. (2016).** Toward hybrid method for parsing modern standard Arabic. 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD, IEEE, pp. 451–456. DOI: 10.1109/SNPD. 2016.7515939.
3. **Abney, S. P. (1991).** Parsing by chunks. Principle-based parsing, Springer, Dordrecht, Vol. 44, pp. 257–278. DOI: 10.1007/978-94-011-3474-3_10.
4. **Ramshaw, L. A., Marcus, M. P. (1999).** Text chunking using transformation-based learning. In: Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., Yarowsky, D. (eds) Natural Language Processing Using Very Large Corpora, Text, Speech and Language Technology, Vol 11. DOI: 10.1007/ 978-94-017-2390-9_10.
5. **Khoufi, N., Aloulou, C., Belguith, L. H. (2014).** Chunking Arabic texts using conditional random fields. 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications, AICCSA, IEEE, pp. 428–432 DOI: 10.1109/AICCSA.2014. 7073230.
6. **Khoufi, N., Boudokhane, M. (2013).** Statistical-based system for morphological annotation of Arabic texts. Proceedings of the Student Research Workshop associated with RANLP 2013, pp. 100–106.
7. **Zitouni, I. (2014).** Natural language processing of Semitic languages, Berlin: Springer. pp. 299–334.
8. **Khoufi, N., Aloulou, C., Belguith, L. H. (2016).** Parsing Arabic using induced probabilistic context free grammar. International Journal of Speech Technology, Vol. 19, pp. 313–323. DOI: 10.1007/s10772-015-9300-x.
9. **Khoufi, N., Aloulou, C., Belguith, L. H. (2015).** Arabic probabilistic context free grammar induction from a treebank. Research in Computing Science, Vol. 90, pp. 77–86.
10. **Debili, F., Achour, H., Souissi, E. (2002).** La langue arabe Et L'ordinateur: de L'étiquetage grammatical à la Voyellation automatique. Correspondances, Vol. 71, pp. 10–28.
11. **Diab, M., Hacioglu, K., Jurafsky, D. (2004).** Automatic tagging of Arabic text: From raw text to base phrase chunks. Proceedings of HLT-NAACL 2004: Short papers, pp. 149–152.
12. **Diab, M. (2007).** Improved Arabic base phrase chunking with a new enriched POS tag set. Proceedings of the 2007 Workshop on

Computational Approaches to Semitic Languages: Common Issues and Resources, pp. 89–96.

13. **Allwein, E. L., Schapire, R. E., Singer, Y. (2000).** Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research, Vol. 1, pp. 113–141.
14. **Mohammed, M. A., Omar, N. (2011).** Rule based shallow parser for Arabic language. Journal of Computer Science, Vol. 7, No. 10, pp. 1505–1514. DOI: 10.3844/jcssp.2011. 1505.1514.
15. **Allwein, E. L., Schapire, R. E., Singer, Y. (2000).** Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of machine learning research, Vol. 1, pp. 113–141.
16. **Fraj, F. B., Kessentini, M. (2012).** Combinatorial classification for chunking Arabic texts. International Journal of Artificial Intelligence & Applications, Vol. 3, No. 5, pp. 63–71. DOI: 10.5121/ijaia.2012.3506.
17. **Khoufi, N., Aloulou, C., Hadrich-Belguith, L. (2015).** Enhancing CRF model with N-grams for Arabic texts chunking. The 25th International Business Information Management Conference, IBIMA 2015, pp. 2877–2884.
18. **Maamouri, M., Bies, A., Buckwalter, T., Mekki, W. (2004).** The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. NEMLAR conference on Arabic language resources and tools, Vol. 27, pp. 466–467.
19. **Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013).** Efficient estimation of word representations in vector space. DOI: 10.485 50/arXiv.13 01.3781.
20. **Barhoumi, A., Camelin, N., Aloulou, C., Estève, Y., Belguith, L. H. (2019).** An empirical evaluation of Arabic-specific embeddings for sentiment analysis. International Conference on Arabic Language Processing, Springer, Cham. pp. 34–48.
21. **Sahidullah, M., Patino, J., Cornell, S., Yin, R., Sivasankaran, S., Bredin, H., Korshunov, P., Brutti, A., Serizel, R., Vincent, E., Evans, N., Marcel, S., Squartini, S., Barras, C. (2019).** The speed submission to DIHARD II: Contributions & lessons learned. arXiv preprint arXiv:1911.02388. DOI: 10.485 50/arXiv.1911.02388.
22. **Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J. (2009).** A novel connectionist system for improved unconstrained handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 3, No. 5.
23. **Sak, H., Senior, A. W., Beaufays, F. (2014).** Long short-term memory recurrent neural network architectures for large scale acoustic modeling. INTERSPEECH 2014, pp. 338–348.
24. **Sundermeyer, M., Alkhouli, T., Wuebker, J., Ney, H. (2014).** Translation modeling with bidirectional recurrent neural networks. Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP, pp. 14–25.
25. **Cho, K., van-Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014).** Learning phrase representations using RNN encoder-decoder for statistical machine translation. DOI: 10.48550/arXiv.1406.1078.
26. **Gers, F. A., Schmidhuber, J., Cummins, F. (2000).** Learning to forget: Continual prediction with LSTM. Neural computation, Vol. 12, No. 10, pp. 2451–2471. DOI: 10.1162/0899766 00300015015.
27. **Ravanelli, M., Brakel, P., Omologo, M., Bengio, Y. (2018).** Light gated recurrent units for speech recognition. IEEE Transactions on Emerging Topics in Computational Intelligence, Vol. 2, No. 2, pp. 92–102. DOI: 10.1109/TETCI.2017.2762739.
28. **Gruber, N., Jockisch, A. (2020).** Are GRU cells more specific and LSTM cells more sensitive in motive classification of text? Frontiers in artificial intelligence, Vol. 3, No. 40. DOI: 10.3389/frai.2020.00040.

Article received on 27/03/2023; accepted on 22/04/2024.
**Corresponding author is Nabil Khoufi.*