Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 4904–4912
Marseille, 20-25 June 2022

Enhancing Deep Learning with Embedded Features for Arabic Named Entity Recognition

Ali Lotfy[1], Caroline Sabty[2], Slim Abdennadher[2]
[1]German University in Cairo, [2]German International University
[1]El Tagamoa El Khames, New Cairo, Cairo, Egypt, [2]Administrative Capital, Regional Ring Rd, Cairo, Egypt
ali.hatab@student.guc.edu.eg, {caroline.sabty, slim.abdennadher}@giu-uni.de

Abstract
The introduction of word embedding models has remarkably changed many Natural Language Processing tasks. Word embeddings can automatically capture the semantics of words and other hidden features. Nonetheless, the Arabic language is highly complex, which results in the loss of important information. This paper uses Madamira, an external knowledge source, to generate additional word features. We evaluate the utility of adding these features to conventional word and character embeddings to perform the Named Entity Recognition (NER) task on Modern Standard Arabic (MSA). Our NER model is implemented using Bidirectional Long Short Term Memory and Conditional Random Fields (BiLSTM-CRF). We add morphological and syntactical features to different word embeddings to train the model. The added features improve the performance by different values depending on the used embedding model. The best performance is achieved by using Bert embeddings. Moreover, our best model outperforms the previous systems to the best of our knowledge.

Keywords: Arabic Natural Language Processing, Named Entity Recognition, Deep learning

1. Introduction

The Arabic language is spoken in a large area, including North Africa and the Arab Peninsula. Although there is no agreed-on statistic estimating the number of speakers of each language, Arabic always comes among the top 10 spoken languages. Arabic is also one of the oldest living languages (Al-Jallad, 2017), which adds to the importance of its processing.

The processing of the Arabic language faces many challenges that arise from its unique nature. For instance, an Arabic word is often connected to one or many suffixes and/or one, or many prefixes (Awad et al., 2018a). In addition, some words are changed based on their position in a sentence. Due to this complex morphological system, the number of unique words in a document becomes relatively big. Every letter in Arabic can have one of four different states (the four basic diacritics). Another diacritic (Shadda) is also used to indicate two letters being written as one letter. Different diacritics on the same sequence of characters produce other words (Sabty et al., 2018). These diacritics are not written in most Arabic texts, as Arabic speakers can recognize them easily from context. Moreover, Arabic is a relatively low-resource language.

The named entity recognition (NER) task is one of the classical tasks of Natural Language Processing. There are particular challenges for the NER task, shared between all languages. They include naming organizations and locations after famous person names and the nested named entities. Due to the complex morphology of the Arabic language, the Arabic NER task faces more challenges than other languages, like lack of capitalization. Capitalization indicates that a word is a proper noun.

The implementation of the NER task has evolved according to advancements in algorithms and computational power. Early NER systems depended on hand-crafted rules. Afterward, machine learning approaches were trained on hand-crafted features to classify words into different named entity classes. Recently, deep learning techniques that rely on word embeddings became dominant. Word embedding models automatically capture hidden features in sentences (Sabty et al., 2019b). They also capture the semantic aspects of words, which enhances the performance and generalizability of NER models (Sabty et al., 2019a). In this work, we combine the advantages of different approaches by adding information from external sources as features to word embeddings. We used four different word embedding models. We also added character embeddings as a secondary representation of words. We used a language analyzer to generate three additional features and designed one hand-crafted feature. The added features and the character embedding significantly improved the performance of the NER BiLSTM-CRF model. To the best of our knowledge, this is the first work that incorporates features from external sources in a deep learning model that performs the NER task for the Arabic language. Our model achieved an F1-score of 86.71% on the ANERcorp 2 and 79.48% on the AQMAR corpus (Mohit et al., 2012), outperforming all previous works.

One of the drawbacks of assessing language tasks using the known evaluation metrics is that they reveal no information about the nature of failures or successes of the tested model. In addition, a natural phenomenon in datasets is the repetition of entities, and similar to this is the resemblance between different entities as they occur in the same article. This makes the scores achieved

by a model misleading. To overcome this drawback, we examined the mispredicted entities to know the limitations of the model and how they reflect on its performance.
The main contributions of this paper are evaluating the utility of adding information from an external knowledge source to word embeddings, achieving the state-of-the-art in Arabic NER, and describing the limitations of using deep learning to recognize Arabic named entities. The rest of the paper is organized as follows. Section 2 presents some related work for the Arabic NER task. Section 3 describes our deep learning NER model. Section 4 explains the experiments done to evaluate the model. Section 5 discusses the errors made by the model. Section 6 concludes the paper.

2. Related Work

The three main approaches used in NER systems are the rule-based, machine learning-based, and deep learning approaches. Regarding the rule-based approach in (Mesfar, 2007), they developed an Arabic NER system using Nooj linguistic environment. The system achieved an F1-score of 85% for a person, 84% for organization, and 76% for location classes. Also, in (Maloney and Niv, 1998), they used a morphological analyzer to extract word features. They achieved an F1-score of 85% on five different named entity classes. The second main approach is machine learning. In (Benajiba and Rosso, 2008), they used Conditional Random Fields (CRFs) to classify words. Features were extracted using external tools that generate part-of-speech (POS) tags and base phrase chunk tags. This model achieved an F1-score of 79.21% on the ANERcorp. An SVM classifier was used in (Benajiba et al., 2008) to detect named entities. They used lexical, morphological, and contextual features. Gazetteers were also used as an external knowledge source. This system achieved an F1-score of 82.17% on the ACE 2003 data.
Also, a combination of both rule-based and machine learning-based approaches was used as a hybrid approach. In (Shaalan and Oudah, 2014) and (Meselhi et al., 2014), NER systems using hybrid techniques were implemented. The models proposed incorporated hand-crafted rules and gazetteers in a machine learning model. The first system was achieved using ANERcorp F1-score equal to 94.0% for the person, 90% for location, and 88% for the organization types. The second one reached 96.65% for a person, 92.90% for location, and 94.80% for organization classes.
After the neural network, systems have been proposed for the NER task, which improved performance significantly. In (Awad et al., 2018b), they used BiLSTM along with word and character embeddings. The model used a corpus composed of both AQMAR and ANERcorp. An F1-score of 76.65% was achieved on 20% of the data.
A BiLSTM-CRF model was used in (El Bazi and Laachfoubi, 2019), (Ali et al., 2019) and (Khalifa and Shaalan, 2019) along with different word and character embeddings models. In (El Bazi and Laachfoubi, 2019) the ANERcorp is split into 10% test, 10% dev, and 80% train. The model achieves an F1-score of 90.6%. In (Ali et al., 2019) they achieved F1-score is 91.3% on a corpus containing both ANERcorp and AQMAR without considering the miscellaneous entities. In (Khalifa and Shaalan, 2019) they achieved an F1-score of 88.77% on the ANERcorp.
The usage of AraBert word embedding with the BiLSTM model in (Antoun et al., 2020) achieved an F1-score equal to 83.1% on the Camel split of the ANERcorp. In (Obeid et al., 2020), AraBert was fine-tuned. In (Youssef et al., 2020), they concatenated multi embeddings together instead of choosing one embedding model to be used with a BiLSTM-CRF model. An F1-score of 77.62% was achieved on the AQMAR dataset. BERT was trained in (Helwe et al., 2020) in a semi-supervised learning approach for Arabic NER using labeled and semi-labeled datasets. They relied on the pre-trained model of AraBERT and followed the approach of the teacher-student learning mechanism. They achieved an F1-score equal to 65.5% on the AQMAR dataset.
A model was proposed in (Liu et al., 2019) that integrates various tailored techniques, including representation learning, feature engineering, sequence labeling, and ensemble learning. The final model achieves an F1-score of 75.82% on the AQMAR dataset.
The utility of using hand-crafted features in deep learning models was evaluated for the English NER task in (Wu et al., 2018) but was not evaluated for the Arabic NER before. This approach is similar to the approach we adopted here. Three hand-crafted features were concatenated to word embeddings to form the input representation for a BiLSTM-CNN-CRF model. Results on the Conll 2003 English dataset in (Sang and De Meulder, 2003) show that hand-crafted features significantly improve the F1-score of the NER task from 91.06% to 91.89%.

3. Arabic NER Model

The Arabic NER model we proposed is implemented using deep learning techniques. The architecture of the model is based on BiLSTM-CRF, and it is implemented using Keras framework (Chollet and others, 2015). The Bi-LSTM component has proven its ability to capture context, while the CRF component makes the best prediction for the whole sequence of words. Both layers were used in many sequence tagging tasks as in (Lample et al., 2016), (Chiu and Nichols, 2016) ,and (Huang et al., 2015). In Figure 1 the general model architecture is illustrated. It is composed of a single BiLSTM layer. The input sentences are represented by the concatenation of word embedding, character embedding, and the set of additional features. We tuned the hyper-parameters of the model by manually choosing the parameters that achieve the best performance.

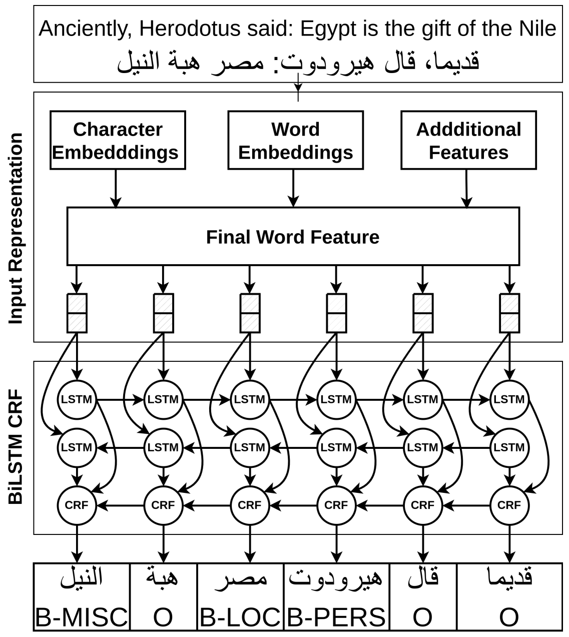

Figure 1: Model Architecture

The dropout of the embedding layer is equal to 0.3, and the dropout of the BiLSTM layer is equal to 0.5. The number of LSTM units is 200. The activation function is Hard-Sigmoid, and the optimizer is Nadam. The batch size is 5. We used no validation as experiments showed that saving the validation data for training achieves higher performance.

3.1. Character Embedding

The first principal component of the input sequence for our NER model is the character embeddings. In Arabic, words can have many suffixes and prefixes. They can also have different versions. Using character-level representations can help the model recognize these differences between versions of a word and overcome the complex morphology of Arabic. We used CNN to generate character embeddings of words as they proved that they could capture the morphological feature of language (Zhang et al., 2015). As shown in Figure 2 character encodings are fed to a 2D CNN layer followed by a max-pooling layer. The activation function used in the CNN layer is Tanh. The number of filters is 25, and kernels are 3. The dimension of the character embedding is 100. The output character embeddings are then concatenated with word embeddings and additional features to train the BiLSTM-CRF part of the model.

3.2. Word Embedding

The second principal component of the input sentence for our NER model is word embedding. In general, introducing a contextualized word embedding model led to a revolution in Natural Language Processing. The utilization of BERT embeddings led to state-of-the-art results in many tasks. We used several types of embeddings to represent the input words. The embeddings used are AraBert, which is the Arabic version of Bert (Antoun et al., 2020). We used AraBert large version 2.0. The generated vectors are the mean of all transformers layers with mean subtoken pooling. Neither context was used in generating the embeddings, nor were the embeddings finetuned. In addition, we used classical word embeddings models such as FastText (Bojanowski et al., 2017), GloVe (Tarekeldeeb, 2018), and AraVec (Soliman et al., 2017), which is the Arabic version of Word2Vec. The dimensions of word embeddings are 1024, 300, 256, and 300, respectively.

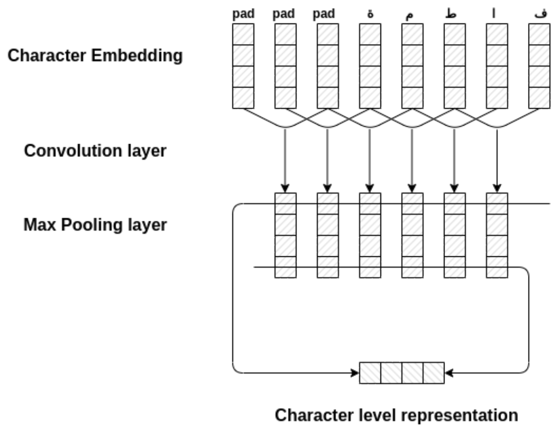

Figure 2: CNN Architecture

3.3. Additional Features

We added four external features to the word and character embeddings to form the final word feature. We used the 2019 release of the Madamira tool (Pasha et al., 2014) to generate the Part-of-speech (POS), capitalization, and word analysis features. Madamira can predict POS with an accuracy of 95.9% and has a complete evaluation accuracy of 84.1%. Moreover, we designed a fourth feature which is the quote feature. The chosen features were previously used in traditional machine learning and rule-based approaches related to named entities. All features were represented as one-hot vectors and concatenated to the word and character embeddings. Table 1 shows examples of some of the features generated by Madamira.

Capitalization feature indicates that a word is a proper noun. Arabic letters do not have cases like Latin letters. The English translation of words compensates for capitalization. The capitalization feature of an Arabic word is positive if the translation provided by Madamira is capitalized. This feature has a length of 1.

Word Analysis feature is a set of attributes for every word. They include gender (male, female), number (single, dual, or plural), state (indefinite, definite, or construct), suffixes, prefixes, voice of a verb (active or passive), person (first, second, or third), case

word	gloss	gen	num	stt	prc3	prc2	prc1	prc0	enc	vox	per	cas	asp	mod	pos
ودعا	call	m	s	na	0	wa	0	0	0	a	3	na	p	i	verb
الجميع	all_of	m	s	d	0	0	0	Al_det	0	na	na	n	na	na	noun
لبذل	spending	m	s	c	0	0	li_prep	0	0	na	na	g	na	na	noun
الجهود	efforts	m	s	d	0	0	0	Al_det	0	na	na	g	na	na	noun
اللازمة	necessary	f	s	d	0	0	0	Al_det	0	na	na	g	na	na	adj
لوضع	laying_down	m	s	c	0	0	li_prep	0	0	na	na	g	na	na	noun
حد	stop	m	s	i	0	0	0	0	0	na	na	g	na	na	noun
لإراقة	pouring_out	f	s	i	0	0	li_prep	0	0	na	na	g	na	na	noun
الدماء	blood	m	s	d	0	0	0	Al_det	0	na	na	g	na	na	noun

Table 1: Example of Features by Madamira

(nominative, genitive, or accusative), aspect of a verb (command, perfective, or, imperfective) and mood of a verb(indicative, jussive, or subjunctive). These attributes allow the model to learn patterns of named entities. Some patterns can also indicate the absence of a named entity. Every attribute is represented as a one-hot vector. All vectors are concatenated to represent the final feature vector. The final vector has a length of 91.

Part-of-Speech (POS) feature is a category indicating a grammatical property for a word. Arabic has three main POS tags: noun, verb, and particle. The main categories are further divided into subcategories. The Madamira tool generates 29 possible POS values making the length of the feature vector also 29. POS tags that belong to the verb or the particle category indicate the absence of named entities. Moreover, the POS tag indicates if a word is a proper noun, .i.e, a name of a particular entity. POS tag can also help the model learn patterns of named entities.

Quote feature indicates whether a word or a set of words is between quotes or not. We added the quote feature to help recognize the miscellaneous entities, which are usually the least recognizable ones. Miscellneous tags ,such as art works, are sometimes put between quotes. The quote feature has a length of 2 to represent the existence of a quote and its type.

4. Experiments

This section explains the two datasets used, along with their preprocessing. Then the training process is discussed, and the results of the different experiments are listed.

4.1. Datasets

ANERcorp is the most used corpus for Arabic NER. It was collected and annotated by (Benajiba et al., 2007). Text in the ANERcorp forms 316 articles extracted from different news websites. Articles included have different topics such as sports, politics, and society. The ANERcorp makes 150,268 tokens, 11 % of which are named entities. Table 2 shows the number of entities in the ANERcorp.

AQMAR is the American Qatari Modeling of Arabic. It is an Arabic NER corpus consisting of 28 Wikipedia articles (Mohit et al., 2012). The topics of the included articles are history, science, sports, and technology. AQMAR makes 73,910 tokens, 12.6% of which are named entities. Table 3 shows the number of entities in AQMAR.

Class	Train	Test
Person	2724	882
Location	3778	656
Organization	1579	450
Miscellaneous	889	229
All	8970	2217

Table 2: Number of Entities in the ANERcorp

Class	Train	Test
Person	1048	424
Location	1117	325
Organization	354	102
Miscellaneous	1763	721
All	4282	1572

Table 3: Number of Entities in AQMAR corpus

4.2. Data Pre-processing

Before generating word embeddings, some forms of noise such as diacritics and special characters connected to words were removed. We set the length of the maximum sequence to be equal to 512. Sentences were added together to meet the maximum length without being cut from the middle, as context plays an essential role in predicting the NER tag. The BiLSTM layer depends on previous and next tokens to encode context. Also, the CRF layer makes predictions for the sequence as a whole.

Embedding	Alone	+Features	+Character Embedding	+Features & Character Embedding
One Hot	54.90	59.7S0	60.00	61.80
GloVe	61.52	72.33*	72.27*	75.22*
AraVec	73.46	77.668*	77.09*	78.10*
FastText	78.34	79.85*	78.87 *	79.44*
AraBert	85.97	**86.47***	85.89	86.00

Table 4: Average F1-scores of 6 runs on ANERcorp in (%)
* indicates statistical significance on the test set against mere embeddings by a paired sample t-test at level $\alpha = 0.05$.

Embedding	Alone	+Features	+Character Embedding	+Features & Character Embedding
FastText	70.66	71.77*	71.63	71.95*
AraBert	77.82	**78.51***	77.42	77.56

Table 5: Average F1-scores of 6 runs on AQMAR in (%)
* indicates statistical significance on the test set against mere embeddings by a paired sample t-test at level $\alpha = 0.05$.

4.3. Training

For the ANERcorp, we trained our BiLSTM-CRF model on four different word embedding models with and without adding the external features and character embedding to each. We trained the model for 40 epochs saving the best performance achieved. We also trained the four word embedding models with each additional feature separately. For the AQMAR corpus, we repeated the first experiment using only AraBert and FastText as they achieved the highest performance among the four embedding models. Glove embeddings are outdated, and they achieve the least performance. FastText compensates for the absence of Word2Vec (AraVec) as they both use the same skip-gram model.
The training of neural networks is a non-deterministic process as it typically depends on a random number generator to initialize the weights of the network. We ran the model six times to overcome this problem and computed the average performance. We also ran a paired sample t-test for the experiments that show small differences in performance.

4.4. Results

Table 4 summarizes the performance of the model using different word embedding models using the ANERcorp dataset. Adding external features increased the F1-score from 54.9% to 59.7% when using one hot embedding. Adding Character embedding alone on one hot embedding increased the F1-score from 54.9% to 60%. Adding the two of them together increased the F1-score from 54.9% to 61.8%. The remarkable increase in performance proves the effectiveness of using features and character embedding together. The best performance comes from AraBert embeddings. AraBert alone achieved 85.97%, and adding features increased the F1-score to 86.47%. However, adding character embeddings to AraBert did not enhance the performance. As shown in Table 5 similar performances were achieved when the model was trained and tested on AQMAR corpus using AraBert and FastText embeddings.

Table 6 shows the performance of the model on the ANERcorp using each additional feature separately with each word embedding model. The results show that the word analysis feature is the most effective. Both capitalization and POS improved the performance. The quote features did not enhance the performance.For AraBert embedding, each feature alone does not improve performance significantly; however, all features together do as shown in Table 4.

Features and character embeddings increased the performance significantly when concatenated with GloVe, AraVec, and one hot embedding. However, they only improved the performance by a small margin when added to AraBert and FastText embeddings. AraVec and GloVe models miss many words as the process of generating embeddings in these models is a static one .i.e it depends on a fixed vocabulary. The number of out-of-vocabulary words in Arabic is relatively significant due to its complex nature; therefore, adding features and character embeddings improved the performance significantly as they compensated for the missing vectors. AraBert and FastText generate embeddings dynamically. AraBert uses an external tool called Farasa to tokenize Arabic words, .i.e to separate words from suffixes and prefixes. FastText splits words into syllables before generating embeddings. FastText and AraBert can handle out-of-vocabulary words, making the performance improvement smaller. In addition, AraBert and FastText exploit more sophisticated training algorithms on more data. Tokenization in FastText and AraBert can also explain Why adding character embedding to AraBert did not improve the performance. Similarly, adding character embedding to FastText with the features did not enhance performance.

Tables 7 and 8 show the precision, recall, and F1-score of the four named entity classes presented in the ANERcorp and AQMAR. Using AraBert embeddings and the additional features achieved the highest F1-score, 86.71%, and 79.48%, respectively.

Embedding	None	+Part of Speech	+Capitalization	+Word Analysis	+Quote
GloVe	61.52	67.97*	64.97 *	69.82 *	61.43
AraVec	73.46	76.31*	75.01*	77.46*	73.90*
FastText	78.34	79.29*	79.13*	80.16*	78.23
AraBert	85.97	86.17	85.87	86.26	86.04

Table 6: Average F1-score of 3 runs in (%) after adding features one at a time
* indicates statistical significance on the test set against mere embeddings by a paired sample t-test at level $\alpha = 0.05$.

Class	Recall	Precision	F1
Location	95.59	89.08	92.22
Miscellaneous	64.83	74.63	69.39
Organization	74.00	83.04	78.26
Person	89.75	92.03	90.88
All	85.66	87.77	86.71

Table 7: Performance measures in (%) of the best model on the ANERcorp

Class	Recall	Precision	F1
Location	90.49	84.05	87.15
Miscellaneous	66.90	72.63	69.65
Organization	75.25	71.70	73.43
Person	91.96	89.63	90.78
All	79.07	79.88	79.48

Table 8: Performance measures in (%) of the best model on AQMAR

4.5. Comparison with Existing Approaches

Different researchers used the ANERcorp with different train-test split ratios. Some researchers do not mention how they split the corpus. Most of them do not consider the miscellaneous entities in evaluation. Moreover, most previous systems were evaluated using different evaluation criteria from the one followed by the author, which led to higher scores. Due to differences in evaluation, the exact split of ANERcorp was published in 2020 in (Obeid et al., 2020). It is called the Camel split. All researchers must use the same training and testing configuration to make results comparison meaningful.

The testing configuration used in this work is the Camel split of the ANERorp. The ANERcorp is sequentially split into two corpora: one for training with a ratio of 83.33% and one for testing with a ratio of 16.67%. For AQMAR, we followed (Helwe and Elbassuoni, 2019) in testing the model on seven articles of the 28. We used the Seqevel Python library (Nakayama, 2018) for calculating the F1-score. Seqeval implements an exact match evaluation adopted by Conll shared tasks and the authors of the ANERcorp and AQMAR. The F1-score is the micro average of the four named entity classes. We follow (Obeid et al., 2020) in only comparing our results with models that used the same evaluation standard.

Model	F1-Score
(Benajiba and Rosso, 2008)	79.21
(Antoun et al., 2020)	83.10
(Obeid et al., 2020)	83.00
Our model	**86.71**

Table 9: Comparison with previous taggers results in (%) on the ANERcorp

Model	F1-Score
(Helwe and Elbassuoni, 2019)	67.22
(Bazi and Laachfoubi, 2018)	61.80
(Liu et al., 2019)	75.82
(Youssef et al., 2020)	77.62
Our model	**79.48**

Table 10: Comparison with previous taggers results in (%) on the AQMAR

Tables 9 and 10 show our results compared to previous models. Our model outperforms all of them. The two models that achieved 83% on the ANERcorp also used AraBert embeddings but with a CRF model. The achieving model 77.62% used a combination of word embeddings, including AraBert and a BiLSTM-CRF model. Adding the BiLSTM component, incorporating the external features, and our preprocessing steps are responsible for the significant performance improvement.

5. Error analysis

The performance of Arabic NER is generally lower than NER in other languages like English and Chinese. We examined the mispredicted tags to find the error sources. To the best of our knowledge, this is the first paper to examine the mispredictions of an Arabic NER system. We do error analysis by comparing the predictions of our model with the actual tags. For the ANERcorp, the total number of false-negative entities is 314, while false-positive entities are 262. After examining the mispredicted words, we classified them into six categories. Table 11 summarizes the error sources. Examples that we mention in the following sections frequently occurred in the test corpus.

5.1. Wrong Tags

After examining the mispredictions, we found that 44.9% of them are mislabelled. Table 12 shows a

Class	Wrong Tags	Tagging Convention	Unclear Context	Unclear Language	Nested Entities	Remaining Failures
Person	62.82	3.84	0	14.1	18.5	0
Location	50.46	15.88	16.82	1.87	0	15
Organization	40.21	4.9	19.56	10.32	4.34	20.7
Miscellaneous	25.58	6.98	20.15	13.18	9.3	24.8
ALL	44.96	13.88	10.41	8.50	7.11	15.14

Table 11: Error Sources Per Class in the ANERcorp in (%)

few examples. Although we only examined the mispredicted words in the testing set, we suppose there are many wrong tags in the rest of ANERcorp. High-quality annotations are critical for both model learning, and evaluation (Li et al., 2020). Incorrect tags make a model learn false knowledge, affecting the overall performance. We estimated the F1-score of the best model to be 93.6% after correcting these wrong tags. Moreover, we started annotating the ANERcorp to correct flaws not recognized by the model[1].

word	tag	prediction
إسبانيا (Spain)	O	B-LOC
الأنبار (Al-Anbar)	O	B-LOC
الزواوي (Al-Zawawy)	O	B-PERS
الوكالة (Agency)	O	B-ORG
الدولية (International)	O	I-ORG
للطاقة (Energy)	O	I-ORG
الذرية (Atomic)	O	I-ORG

Table 12: Examples of Wrong Tags

5.2. Tagging Convention

Many mispredicted words may be considered correct. Most cases are words that describe or identify an entity, for example, the word film festival [مهرجان] in Venice Film Festival [مهرجان البندقية]. One missing token results in a total failure as performance is evaluated using an exact match standard. What decides whether these words are part of the entities or not is the tagging convention. After examining these mispredictions and similar entities, we found that no unified criterion was followed in the tagging process. In addition, country names such as Egypt, France, and Canada are given the tag organization when they represent football teams of these countries, like in the sentence: England beat the Netherlands [فازت إنجلترا على هولندا]. Our model classifies these words as locations. After examining similar words, we found that this rule is applied only in some parts of the ANERcorp. In the Arabic language, a word like Egypt can represent the inhabitants, the government, the army, or the sports team of Egypt. Our model also classifies stadiums and airports as locations while classified as miscellaneous entities in the corpus. Car and airplane models were predicted as miscellaneous entities, while they were tagged as organizations in the corpus. We estimate errors were originating from these issues to be 13.88 % of the mispredictions.

[1]https://github.com/CSabty/ANERcorp-Correction

5.3. Unclear Context

A BiLSTM-CRF model depends heavily on the context in making predictions. Moreover, contextual word embedding models such as Bert and AraBert use context to generate customized word vectors. When an entity appears in an unclear context, the model fails to predict it. An example of ambiguous context from the ANERcorp is one very long sentence that enumerates participants in a racecar followed by their countries and their car model. A part of the sentence is the word Muni[ميوني], which is a name of a person in the context: Muni Italy Ki Tim M [ميوني إيطاليا كي تم إم]. 10.41 % of the mispredictions are due to unclear context.

5.4. Unclear Language

A percentage of errors equal to 8.5 % occurred in sentences when the language is unclear, for example, when entities are composed of or containing non-Arabic words or numbers. Another repeated case is when two names are conjugated using the particle [و], which is not separated by a space from the second token. The model considers the two names as one entity and considers the [و]particle an original part of the second name. This failure only occurs with transliterated names, which rarely appear in Arabic text. An example is in Donald and Palmer [دونالد وبالمر]. This failure occurs although the context indicates that two different names are mentioned. The percentage of errors in this section is 8.5 % of the total mispredictions.

5.5. Nested Entities

Our model recognizes the nested entity when an entity is composed of another famous entity and other words. For example, in entities such as the "Bank of France" or "cup of Africa," the nested entity is only recognized. We considered entities named after entities from other classes in this section. For example, a company named after its founder is recognized as a person. The percentage of errors in this section is 7.11 % of the total mispredictions.

5.6. Remaining Failures

The remaining failures make 15.14% of mispredictions. These are clear, named entities, and we could not specify the source of the error. We believe that failures in this section are due to the nature of the data itself. The miscellaneous class has the worst performance, with an F1-score of 69.39%. The miscellaneous class includes all entities that do not belong to the three main categories: person, location, and organization. Entities that rarely appear in the corpus, such as music genres, currencies, historical eras, and animals, are consistently mispredicted as the model did not learn the common features of these entities. The model gave some named entities different tags, although they appear in similar contexts. We found that these entities and other entities were tagged differently in the training corpus, which confuses the model and results in many failures. Also, some failures exist because they are challenging for the model as they require a deep understanding of the language.

One of the failures of our model is the incapability of differentiating between two occurrences of a word, one of which represents a type of entity and the other represents another type .e.g Riyadh as a City name and as a newspaper name. To solve this issue, we suggest forcing a model to pay more attention to the context by utilizing the masking language model, .i.e hiding the word of interest.

Most researchers do not consider the miscellaneous class when calculating the F1-score. After examining miscellaneous entities, we found that they are very distant from each other. Miscellaneous entities include religions, God, Mosques, churches, artworks, movies, songs, books, animals, stadiums, airports, historical eras, historical events, car models, and currencies. We suggest dividing the miscellaneous class into many sub-classes for a NER model to learn the standard features of every type.

Only locations that humans can recognize because they have prior knowledge of them are predicted wrong by the model. For example, South Africa is a country name that can refer to the southern part of the continent of Africa. South Africa is one entity while North Africa is not. Such failure suggests that using external knowledge sources such as gazetteers is inevitable.

6. Conclusion and Future Work

In this paper, we used a language analyzer to generate morphological and syntactical features of words. We evaluated the utility of using the three additional features, one hand-crafted feature, and four different embedding models. We also added character embeddings as a secondary representation. Our experiments proved that the extra features improved the performance of all the embedding models when fed to a BiLSTM-CRF NER model using AraBert embeddings, and the additional features achieved an F1-score of 86.71% and 79.48% on AQMAR corpus, which outperforms all previous models. Furthermore, we analyzed the failures of the model to evaluate the limitations of using deep learning in NER.

In future work, we suggest solving the annotation quality problem by setting strict rules for the annotation process and applying these rules to existing corpora and new data. We also recommend designing separate tests that focus on different NER challenges and include all common occurrences of named entities. All NER taggers test on arbitrary sentences that do not necessarily represent the language of interest. In addition, we believe that using preprocessing tools that normalize data and correct common mistakes can improve the performance of NER taggers.

7. Bibliographical References

Al-Jallad, A. (2017). The earliest stages of arabic and its linguistic classification. In *The Routledge Handbook of Arabic Linguistics*, pages 315–331. Routledge.

Ali, M. N. A., Tan, G., and Hussain, A. (2019). Boosting arabic named-entity recognition with multi-attention layer. *IEEE Access*, 7:46575–46582.

Awad, D., Sabty, C., Elmahdy, M., and Abdennadher, S. (2018a). Arabic name entity recognition using deep learning. In *International Conference on Statistical Language and Speech Processing*, pages 105–116. Springer.

Awad, D., Sabty, C., Elmahdy, M., and Abdennadher, S. (2018b). Arabic name entity recognition using deep learning. In *International Conference on Statistical Language and Speech Processing*, pages 105–116. Springer.

Bazi, I. E. and Laachfoubi, N. (2018). Arabic named entity recognition using word representations. *arXiv preprint arXiv:1804.05630.*

Benajiba, Y. and Rosso, P. (2008). Arabic named entity recognition using conditional random fields. In *Proc. of Workshop on HLT & NLP within the Arabic World, LREC*, volume 8, pages 143–153. Citeseer.

Benajiba, Y., Diab, M., Rosso, P., et al. (2008). Arabic named entity recognition: An svm-based approach. In *Proceedings of 2008 Arab International Conference on Information Technology (ACIT)*, pages 16–18. Citeseer.

Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

El Bazi, I. and Laachfoubi, N. (2019). Arabic named entity recognition using deep learning approach. *International Journal of Electrical & Computer Engineering (2088-8708)*, 9(3).

Helwe, C. and Elbassuoni, S. (2019). Arabic named entity recognition via deep co-learning. *Artificial Intelligence Review*, 52(1):197–215.

Helwe, C., Dib, G., Shamas, M., and Elbassuoni, S. (2020). A semi-supervised bert approach for arabic named entity recognition. In *Proceedings of the Fifth Arabic Natural Language Processing Workshop*, pages 49–57.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Khalifa, M. and Shaalan, K. (2019). Character convolutions for arabic named entity recognition with long short-term memory networks. *Computer Speech & Language*, 58:335–346.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Li, J., Sun, A., Han, J., and Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Liu, L., Shang, J., and Han, J. (2019). Arabic named entity recognition: what works and what's next. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 60–67.

Maloney, J. and Niv, M. (1998). Tagarab: a fast, accurate arabic name recognizer using high-precision morphological analysis. In *Computational approaches to semitic languages*.

Meselhi, M. A., Bakr, H. M. A., Ziedan, I., and Shaalan, K. (2014). A novel hybrid approach to arabic named entity recognition. In *China Workshop on Machine Translation*, pages 93–103. Springer.

Mesfar, S. (2007). Named entity recognition for arabic using syntactic grammars. In *International Conference on Application of Natural Language to Information Systems*, pages 305–316. Springer.

Nakayama, H. (2018). seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Sabty, C., Elmahdy, M., and Abdennadher, S. (2018). Arabic named entity recognition using clustered word embedding.

Sabty, C., Elmahdy, M., and Abdennadher, S. (2019a). Named entity recognition on arabic-english code-mixed data. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 93–97. IEEE.

Sabty, C., Sherif, A., Elmahdy, M., and Abdennadher, S. (2019b). Techniques for named entity recognition on arabic-english code-mixed data. *International Journal of Transdisciplinary AI*, 1(1):44–63.

Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Shaalan, K. and Oudah, M. (2014). A hybrid approach to arabic named entity recognition. *Journal of Information Science*, 40(1):67–87.

Wu, M., Liu, F., and Cohn, T. (2018). Evaluating the utility of hand-crafted features in sequence labelling. *arXiv preprint arXiv:1808.09075*.

Youssef, A., Elattar, M., and El-Beltagy, S. R. (2020). A multi-embeddings approach coupled with deep learning for arabic named entity recognition. In *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 456–460.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

8. Language Resource References

Antoun, W., Baly, F., and Hajj, H. (2020). Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Benajiba, Y., Rosso, P., and Benedíruiz, J. M. (2007). Anersys: An arabic named entity recognition system based on maximum entropy. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 143–153. Springer.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Mohit, B., Schneider, N., Bhowmick, R., Oflazer, K., and Smith, N. A. (2012). Recall-oriented learning of named entities in arabic wikipedia. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 162–173.

Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., Inoue, G., Eryani, F., Erdmann, A., and Habash, N. (2020). Camel tools: An open source python toolkit for arabic natural language processing. In *Proceedings of the 12th language resources and evaluation conference*, pages 7022–7032.

Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R. (2014). Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Lrec*, volume 14, pages 1094–1101. Citeseer.

Soliman, A. B., Eissa, K., and El-Beltagy, S. R. (2017). Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265.

Tarekeldeeb. (2018). tarekeldeeb/glove-arabic: Glove model for distributed arabic word representation. `https://github.com/tarekeldeeb/GloVe-Arabic`.