BERT for Arabic Named Entity Recognition

1st Azroumahli Chaimae
SIGL Laboratory, ENSA
AbdelMalek Essaadi University
Tetuan, Morocco
c.azroumahli@uae.ac.ma

2nd El Younoussi Yacine
SIGL Laboratory, ENSA
AbdelMalek Essaadi University
Tetuan, Morocco
yacine.elyounoussi@uae.ac.ma

3rd Maciej Rybinski
Khaos Research
Universidad de Malaga
Spain
maciek.rybinski@lcc.uma.es

4th José F. Aldana Montes
Khaos Research
Universidad de Malaga
Spain
jfam@lcc.uma.es

2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT) | 978-1-6654-2222-2/20/$31.00 ©2020 IEEE | DOI: 10.1109/ISAECT50560.2020.9523676

***Abstract*—In the recent year, transfer learning and contextual word Embedding models have majorly improved the performance of several NLP tasks for many natural languages. In this paper, we tackle the Arabic NER task while exploiting the effectiveness of using a context-dependent and transformer based representational language model. We train Arabic transformer-based representational language models using BERT [1], and compare them to the pre-trained multi-lingual model by building and evaluating the resulted Arabic Named Entity Recognition systems. Our approach utilizes BERT as multi-layer bidirectional transformer encoder that helped in learning deep bidirectional Arabic word representations, by training and fine-tuning the pre-trained model to generate the most accurate NER system for our datasets. Our system achieved the accuracy of 0.91 which demonstrated that creating a language-specific model can improve the performance of a specific NLP task in comparison to already trained Multi-language model.**

***Index Terms*—Arabic NLP, NER, Word Embeddings, BERT**

I. Introduction

Named Entity Recognition (NER) is the task of identifying and extracting spans of texts called Named Entities (NE) from a larger set of text, then categorizing them into predefined semantic categories [2]. The tag sets of these NEs were defined in the Sixth Message Understanding Conference (MUC-6) as four categories. ENAMEX for people names, organizations and places, TIMEX for temporal expressions, NUMEX for numerical expression, and MISC for proper names that are not in the ENAMEX tag set [3]. NER is considered an important key part of many downstream Natural Language Processing (NLP) applications, its priority is justified by the fact that texts always revolve around the NE. Thus, the performance of a set NLP application can directly be affected by their quality. Furthermore, the ambiguity of most natural languages makes building a NER system a trivial task. We work on Arabic language, which is known as one of the Semitic languages with a complex morphology. Its characteristics increase the ambiguity of its syntactic and semantic features, thus, increase the complexity level of building a NER system [4].

In literature, there has been an increasing interest in using unlabeled data to learn word representations or Word Embeddings that can automatically learn the word's semantic and syntactic features. They are widely used in supervised or unsupervised NLP machine learning approaches, to overcome the languages' lack of resources and morphological complexity, which made it easier to process natural languages. Word2Vec [5], and Glove [6] were the first methods that were used to create these Word Embeddings. They relied on the distributional hypothesis ("You shall know a word by the company it keeps" (Firth, 1957)) to create word-level vector representations. Over the last two years, two models BERT [1] and ELMO [7] were introduced to the NLP community. They are built and fine-tuned using a slightly different architecture than the classical Traditional word-level vector representations, and they greatly improved several downstream NLP applications.

In this paper, we follow the BERT approach to pre-train and fine-tune BERT models for an Arabic NER task. As far as we know, there are no publicly-available pre-trained BERT models for the different varieties of Arabic. We address this need by using two different Arabic datasets to create Arabic BERT models and compare them to the multi-lingual pre-trained BERT model on the NER task.

The rest of this paper is structured as follow, we start by presenting a brief background on Word Embeddings in general and justify the choice of the BERT architecture for our work. We highlight the two major steps for creating an NLP application using BERT. We proceed by illustrating the experiments setups. We describe the training datasets, the procedure and the computational cost of pre-training two Arabic models, and fine-tuning the two pre-trained models in addition to a pre-trained multilingual model for an Arabic NER system. Before concluding, we discuss and compare the obtained results, and how our pre-trained model outperformed existing ones on the downstream NER task.

II. Bidirectional Encoder Representations from Transformers: Background

A. *Bert vs Traditional Word Embeddings*

Formerly, processing texts relied on creating word representations by searching key terms using their frequency in the training corpus. The generated representations held no word's syntactic or semantic features, which made it not as useful for machine learning mathematical and statistical models. This all changed however when word Embeddings models like Word2Vec [5] and Glove [6] were introduced to the NLP community. These word Embeddings are real-valued vectors built using the words' context in a large non-annotated corpus. Consequently, this kind of representation can carry semantic and syntactic properties of words, which is very useful for many NLP tasks [8]. Nonetheless, the word

Embeddings approach didn't mend all the complexities of natural languages. For instance, Word2vec can only store the feed-forward information which results in building similar representations for several polysemy words used in different contexts.

The approach of using word Embeddings in NLP application was revolutionized even more with ELMO and BERT. They presented an improvement on the traditional contextualized word representations. Thus Before, word Embeddings were generally context-independent so each word is associated with one real-valued vector that combines all the word's features. While ELMO and BERT can generate different representations for one word capturing its different contexts and positions in a sentence. Word2Vec and Glove, for instance, don't consider the order of the words while training, ELMO and BERT, however, acknowledge it [9].

For this paper, we chose to work with BERT instead of ELMO. Our decision was based on several arguments like the fact that BERT doesn't just provide the word Embeddings that can be used as features, it can incorporate the NLP task to be fine-tuned as an integrated task-specific architecture. Several research papers favoured using BERT; The authors of [10] for instance, stated that BERT can generally be found superior to ELMO on a variety of NLP tasks, including in the clinical domain. Besides, the authors of [1] the creators of BERT model demonstrated on their paper that BERT outperformed human performance by 2.0.

B. *Bert architecture*

BERT architecture is a stack of Transformers, where the models are built using a structure of a multi-layer bidirectional Transformer encoder-decoder. BERT is based on the transformer concept that was introduced in [11], it adopted the transfer learning and the tunable language model for training NLP tasks by using the encoders of the transformer and masked language models. The Transformers were used to replace the LSTM layers and improve on long-term dependencies. Figure 1 illustrates how the transformers are structured; The first component is the stack of encoders, the second component is the stack decoders, and the third component is the self-attention layers.

The attention is a mechanism employed by the transformers that decide the context of words that can contribute to the Embedding of a specific word. It is calculated to output the weighted sum value Vector V by using the matrix of the queries Q, and the matrix of the keys K (1).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{dk}}\right)V \qquad (1)$$

The Embeddings of words occurs as inputs on the lowest encoder in the transformer structure. The input of the encoders goes through a self-attention layer that considers the existence of other words while encoding a specific word and outputs a layer that gets fed to a feed-forward neural network. The input of a decoder goes through a self-attention layer, encoder-decoder attention layer, then a feed-forward neural network. The benefits of the replacement of the LSTMs by the Transformers appear in this architecture; The words in different positions follow their different paths throughout the encoders. So, while the dependencies exist between the paths of the self-attention layer, they do not exist in the feed-forward layer. Hence, the different paths that can be executed in parallel, can follow through feed-forward layer [12].

C. *Pre-training and Fine-tuning BERT*

Creating an NLP application using BERT involves two steps; Pre-training the models on unlabeled data, and Fine-tuning the pre-trained models' parameters using the labelled data for the downstream task [1].

*1) **Pre-training BERT**:* BERT pre-training includes the masked language model task, and the next sentence prediction task. The masked language model was used in BERT to train deep bidirectional representations while avoiding words seeing themselves in a multi-layer context. It masks 15% of the input tokens randomly, then tries to predict them. The masked words, however, are not always replaced with the masked token, the training data generator chooses 15% of the tokens at random. This is done to avoid the mismatches created between the pre-training and the fine-tuning since the masked tokens are never seen during the fine-tuning step [1]. The masked language model doesn't achieve the understanding of the relationship between two sentences. The next sentence prediction task is used to pre-train a binarized sentence that can be trivially generated from any monolingual corpus. This task relies on the idea that while choosing sentences A and B for each pre-training example, there is a possibility of 50% that the next sentence that follows A is B, and another 50% possibility that a random sentence from the corpus is the sentence that follows A. The random sentence is chosen completely at random, and the writers of [1] claim that the final pre-trained model achieves an accuracy of 97%-98% at the next sentence prediction task.

*2) **Fine-tuning BERT**:* One of the advantages of using BERT is that a few parameters need to be learned from scratch. BERT uses a straightforward fine-tuning approach, where a classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream NLP task [1]. The representation of the first token in the input sequence is obtained by taking the output of the Transformer. The first token of every sequence is represented as the special classification embeddings vector $C \in \mathbb{R}^H$. The parameters of BERT, and the parameters of the added classification layer $W \in \mathbb{R}^{K \times H}$ are fine-tuned to maximize the log-probability of the correct label $P \in \mathbb{R}^K$ (2). P is computed using a standard SoftMax, while K represents the number of the classifier labels.

$$P = softmax\left(C \cdot W^T\right) \qquad (2)$$

III. Pre-training and Fine-tuning an Arabic BERT model for a NER system

As it was pointed out in the Introduction, this work aims to build an Arabic NER system. We use BERT to pre-train two

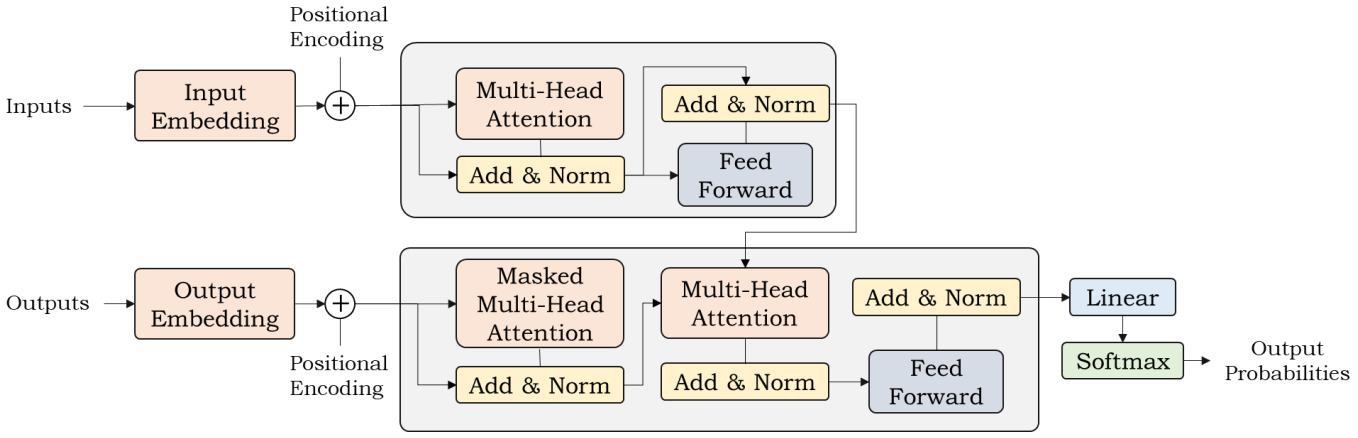

Fig. 1. The architecture of the Transformers

Arabic BERT models using two different Arabic datasets. We fine-tune the created two models in addition to the pre-trained BERT multi-lingual model[1] on the NER task and compare their performances. In this section, we describe our approach thoroughly from the creation of the non-annotated and the annotated datasets to the pre-training and the fine-tuning of the BERT model for an Arabic NER task.

A. *Datasets*

*1) **The Pre-training datasets**:* The DIGLOSSIA phenomena exhibited by the Arabic speaking community makes it as different varieties of Arabic can be spoken by the same speakers for different specific purposes [13]. The most used Arabic varieties are the modern standard Arabic (MSA), and the Arabic colloquial dialects (AD). MSA is the official language used in education, media and formal communications across the different Arabic speaking countries. AD is the language used in daily informal conversations, it has no orthographic standards, and can vary from a region to another across the Arabic countries.

In prior work [13], we collected data from different text domains. We use the same datasets to create BERT models for the different Arabic varieties. The datasets were collected from two main web resources; Wikipedia[2] the online encyclopedia that provides more the 810k Arabic articles. Social media because the majority of the Arabic content in social media is known for containing AD such as Egyptian, Gulf, Moroccan, Tunisian, Algerian, Levantine, ...etc. The Wikipedia dataset that will serve as the MSA dataset was harvested using a function-based algorithm provided by Wikipedia Library[3]. The social media dataset, however, was collected using Twitter API[4], Facebook API with its Excel Add-In[5]. To make sure that the AD dataset contains the majority of the AD varieties, we chose the two platforms (i.e. Twitter, Facebook) because the middle eastern use both platforms, but the North African tend to utilize Facebook more than Twitter [13]. Table III-A1 shows the statists of the datasets used to pre-train the Arabic BERT models.

TABLE I
THE DATASETS STATISTICS

Source	Wikipedia	Social-media	
		Facebook	Twitter
The targeted Arabic variety	MSA	MSA & North African dialects (Egyptian, Tunisian, Algerian, Moroccan, ...)	MSA & Middle Eastern dialects (Gulf, Syrian, Jordanian, ...)
Vocab size	325544	256226	

*2) **The Fine-tuning dataset**:* The created pre-trained BERT models need to be fine-tuned using a NER annotated corpus. For our experiment, we used an annotated dataset provided by AQMAR[6] project. This dataset contains 28 hand-annotated articles. They are tagged to 9 named entities using the BIO system tags i.e. O (outside), B-PER (Beginning of person's entity), B-MIS (Beginning of miscellaneous' entity), B-ORG (Beginning of an organization's entity), B-LOC (Beginning of location's entity), I-PER (Inside of person's entity), I-MIS (Inside of miscellaneous' entity), I-ORG (Inside of an organization's entity), I-LOC (Inside of location's entity). Table III-A2 illustrates the statistics of the NER annotated dataset.

*3) **Pre-processing the datasets**:* Arabic is a highly inflected language, thus before pre-training and fine-tuning the BERT models, several pre-processing steps needs to be applied to the training datasets respecting the Arabic characteristics. Pre-processing social media datasets includes removing the non-Arabic letters thus removing all numbers, URLs and social media users' mentions (@user). The diacritics[7] were removed from both social media and Wikipedia datasets be-

[1]https://github.com/google-research/bert
[2]https://en.wikipedia.org/wiki/List_of_Wikipedias
[3]https://pypi.org/project/wikipedia/
[4]An open source python Library used mainly to access the Twitter API.
[5]https://download.cnet.com/Excel-Add-In-for-Facebook
[6]http://www.cs.cmu.edu/ãrk/ArabicNER/
[7]Diacritical marks are symbols play the role of Latin vowels for altering the pronunciation of phoneme

TABLE II
STATISTICS OF THE NER ANNOTATED CORPUS

Distribution of the entity tokens	Beginning				Inside				Outside
	PER	MISC	ORG	LOC	PER	MISC	ORG	LOC	
	1328	2258	388	1369	875	3505	453	572	60952
Total number of the tokens	71247								
Example of the annotated tokens	مرورا بنظريات التوحيد الكبرى ، و انتهاء بنظرية الأوتار الفائقة... O, B-MIS, I-MIS, I-MIS, O, O, O, B-MIS, I-MIS, I-MIS, ...								

cause their existence is inconsistent. The datasets were further pre-processed by disregarding the Arabic stop words provided in Taher's list[8], and the characters were normalized so that the shape of the Arabic letters will be consistent throughout the different varieties. And finally, the text of both datasets was tokenized into different sentences using the NLTK[9] library [13].

B. Pre-training BERT models

In this experimentation run, we pre-train two Arabic BERT models and use the pre-trained multilingual BERT model to study the impact of BERT unsupervised pre-training on the performance of the NER task. We conduct three major experiments: pre-training unsupervised BERT models on the two pre-training datasets, fine-tune the three BERT model on a NER task, then compare their performances. Our implementations are in TensorFlow that contains several libraries to create state-of-the-art NLP models, and BERT original codebase[10].

*1) **BERT base model**:* The pre-trained BERT model we use in the fine-tuning stage is the BERT-base multilingual model that was released on the BERT paper [1]. It was unsupervised pre-trained on a large amount of cased text containing the top 104 languages with the largest Wikipedia including Arabic. The unsupervised pre-training model is composed of the "Masked-LM" and "Next Sentence Prediction", trained on 12 layers, 768 hidden units, 12 attention heads and 110M hyperparameters.

This multilingual version is generally used to build language models that can be used for the task-specific NLP application in every other language. However, as many papers reported [14] [15], its performance isn't as that adequate as the single-language model. Hence, in addition to fine-tuning this multi-lingual model, we pre-trained single-language Arabic BERT model and fine-tuned them as well to have a fair comparison between the two models for Arabic.

*2) **Pretraining MSA and AD BERT models**:* To pre-train the Arabic single-language BERT models, we strictly follow BERT original pre-training script (the python version). The input of this script is a text file containing a large number of separated sentences, and the model configuration including the vocabulary size of the input text file. Since Wikipedia dataset is relatively large, we had to shard the input file text that should be stored in memory while training. For the MSA model, we implemented both "the masked learning" task, and "next sentence prediction" task. But for the social media model, we only implemented "the masked learning" excluding "next sentence prediction" task. This decision was taken because the social media content (i.e. Tweets, Facebook comments) doesn't have a cognition of a flow of sentences as it happens in dialogues. This model will not be proper for question answering tasks, but it won't affect the performance of the NER task (i.e. a classification task). The MSA and AD BERT models were trained using the following configurations:

- **Creating the training instances:** False for the lower case (i.e. there is no use for cased letters in Arabic). 512 as a maximum of a sequence length. 0.15 as the probability of the masked LM. 77 as the maximum number of masked LM predictions per sequence.
- **The base configuration:** bidirectional transformer pre-training. 0,1 as the dropout ratio for the attention probabilities. "Gulu" as the non-linear activation function. 0,1 the dropout probability for all fully connected layers in the embeddings, encoder, and pooler. 768 as the dimensionality of the encoder layers. 0.02 as the deviation used for initializing the weight matrices. 3072 as the dimensionality of the feed-forward layer of the transformer encoder. 2048 as the maximum sequence length that this model might ever be used with. 12 as the number of attention heads for each attention layer in the transformer encoder. 12 as the number of hidden layers in the transformer encoder. 2 as the vocabulary size of the tokens IDs that will be passed into the BERT model. 156226 as the vocabulary size of the BERT model (325544 for AD), which represents the different tokens passed to the forward method of the BERT model.
- **Input data pipeline configuration:** 32 as the training batch size. 512 as the maximum of a sequence length (128 for the AD model). 77 as the maximum number of masked LM predictions per sequence (20 for the AD model). 10000 as the number of training steps. 100 as the number of the warmup steps. $2e^5$ as the learning rate.

C. Fine-tuning for a NER system

We used the three pre-trained BERT models (i.e. BERT-base, MSA model and AD models) to fine-tune each dataset's models independently to obtain a classifier for the NER task for all Arabic varieties. Again, we strictly follow BERT

[8]The MIT License ©2016 available at: https://github.com/mohataher/arabic-stop-words/blob/master/list.txt

[9]Natural language Toolkit for building Python programs to work with human language data

[10]TensorFlow code and pre-trained models for BERT https://github.com/google-research/bert

original fine-tuning script (the python version), which is easy to adopt for the many NLP specific tasks including NER. Before starting the fine-tuning process, we prepared the training dataset to be able to use it with BERT. The training NER annotated dataset was divided into three parts; 15% for development, 15% for testing and 70% for training.
The fine-tuning BERT script wraps the BERT model and adds token-level classifier on top of the BERT pre-trained model. This classifier is a linear layer that takes as input the last hidden state of the sequence. The pre-trained BERT model is loaded to provide the number of possible labels. The optimizer and the tuning parameters are updated, and a schedule is set to linearly reduce the learning rate throughout the epochs.
The following configurations used for the training are as follow: The models are pre-trained for three epochs (2 epochs for the multilingual model), using a learning rate of $5e^{-5}$ and 10000 training steps. The training batch size was set to 32, and 512 as the maximum of a sequence length.

D. *Computational cost*

Pre-training BERT models are known to be fairly expensive e.g. 4 days using a 16 Cloud TPUs, but when a model is built for a certain language, it can be used to create fine-tuning for a task-specific. The fine-tuning however is inexpensive e.g. 1 hour on a single TPU or a few hours on a GPU.
We followed the recommended settings to decrease the expenses of the computations such as: using an additional 10000 steps with a sequence length of 512. The BERT pre-training and the fine-tuning has been run on a GeForce GPU, with a significant CPU power memory for pre-processing tasks (64Gb RAM, and 1Tb SSD). Our pre-trained models took a significant runtime i.e. roughly 360 hours of computational runtime, while the fine-tuning of all the pre-trained models took 12 hours in total. This runtime doesn't include collecting the datasets, pre-processing and preparing the tokens of the training datasets.

IV. Results and discussion

The results reported in Table III shows the accuracy and loss of the NER task obtained by fine-tuning the three pre-trained models. A glance at the outcome of our experiments reveals a few interesting observations. Admittedly using BERT approach has produced better results than the accuracies obtained by previous Arabic NER systems in the literature e.g. 84.18% that was obtained by Recall-Oriented learning approach [16], 83.22% that was obtained by Combining the Maximum Entropy with POS-tag Information [3], and 73.06% that was obtained by a hybrid approach [17]. This can be explained by the fact that the supervised approaches are limited, and these limits can be overcome with the knowledge of the unsupervised approaches.
The model that was pre-trained using Wikipedia dataset has achieved 0.915 as best accuracy with the minimum loss, followed by the AD model then the multilingual BERT model. Even though the MSA is the best model of the three, the accuracies of the three models are relatively very close with a marge of a difference of 2%.

TABLE III
Results obtained using BERT-base, the MSA model and the AD model

	BERT-base: Multilingual model	MSA model	AD model
Evaluation accuracy	0.8935824	0.9150726	0.9024357
Evaluation loss	0.4889339	0.38895547	0.4774333

It is worth mentioning that the training dataset that was used for fine-tuning contains only the MSA variety, and the content of the social media dataset used for pre-training the AD BERT contains a fraction of the MSA content in addition to the different AD varieties. Also, the Wikipedia dataset used to pre-train the MSA BERT model is a lot larger than the social media dataset. These facts didn't have a major effect on the accuracy of the NER task since both models achieved significant results with a difference of 1.2%. We believe that this outcome will remain true even if the pre-training dataset size was increased.
As for the BERT-base (i.e. the multilingual model), it did perform slightly inferior to the single language model for Arabic, similarly to what was reported for other languages like the work that was described in [14] and [18]. However, the accuracy that was reported by the BERT-base is very impressive for a complex language like Arabic.
The use of the BERT approach and the requirement of a task-specific model architecture proved to be very beneficial for creating an Arabic NER system. Besides, the impressive results reported by the pre-trained model (i.e. the single language pre-trained models and the multi-lingual model), the computational cost of creating a specific NLP-task system (e.g. Sentiment Analysis, Document Classification) can be noticeably decreased as it was reported in section III-D; Once the expensive representation (i.e. the single language BERT model) is pre-computed, many task-specific applications can be generated at a much lower computational cost using BERT fine-tuning or combining the fine-tuning classification layers with other connected network layers.

V. Conclusion

In this work, we describe the approach we adopted to pre-train and create a NER system for multiple Arabic varieties. We pre-train two Arabic models one for the MSA the official Arabic language, and AD that contains the majority of Arabic dialects and MSA, and compare their performances in addition to a multilingual model released by BERT on the NER task. The models were trained using the official BERT source code and fine-tuned on separate annotated training dataset available for several Arabic NLP tasks. The task-specific model's architecture is very beneficial computationally. Besides, the results were very promising; The MSA model has achieved an accuracy of 0.915% on the NER task. This study has its limitation, in future works, we plan on incorporating other NLP tasks to properly investigate the performance of BERT

on Arabic, and improve on the annotated training dataset so it will contain all the varieties of the Arabic dialects in addition to MSA.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[2] H. Graeme, H. Eduard, and J. Mark, *Natural Language Processing of Semitic Languages*, M. R. W. U. Imed Zitouni, Ed. Springer-Verlag Berlin, 2014. [Online]. Available: www.springer.com/series/8899 http://link.springer.com/10.1007/978-3-642-45358-8

[3] Y. Benajiba and P. Rosso, "ANERsys 2 . 0 : Conquering the NER Task for the Arabic Language by Combining the Maximum Entropy with POS-tag Information," *3rd Indian International Conference on Artificial Intelligence*, pp. 1814–1823, 2007.

[4] C. Azroumahli, Y. El Younoussi, and F. Achbal, "An overview of a distributional word representation for an arabic named entity recognition system," in *Advances in Intelligent Systems and Computing*, 2018, vol. 737, pp. 130–140.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4069–4076, 2013. [Online]. Available: http://arxiv.org/pdf/1301.3781v3.pdf

[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://aclweb.org/anthology/D14-1162

[7] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 2227–2237. [Online]. Available: http://arxiv.org/abs/1802.05365

[8] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment Embeddings with Applications to Sentiment Analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 2, pp. 496–509, 2016.

[9] A. Rajasekharan, "Examining BERT's raw embeddings - Towards Data Science," 2019. [Online]. Available: https://www.quora.com/q/handsonnlpmodelreview/Examining-BERT-s-raw-embeddings-1?ch=10&share=4afe89e5

[10] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. B. A. McDermott, "Publicly Available Clinical BERT Embeddings," in *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, 2019, pp. 72–78. [Online]. Available: http://arxiv.org/abs/1904.03323

[11] V. Ashish, S. Noam, P. Niki, U. Jakob, J. Llion, N. G. Aidan, K. Lukasz, and P. Illia, "Attention is all you need," in *Advances in Neural Information Pro- cessing Systems*, 2017, pp. 6000–6010.

[12] J. Alammar, "The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time," p. 1, 2018. [Online]. Available: http://jalammar.github.io/illustrated-transformer/%0Ahttps://jalammar.github.io/illustrated-transformer/%0Ahttp://jalammar.github.io/illustrated-transformer/

[13] C. AZROUMAHLI, Y. El Younoussi, O. Moussaoui, and Y. Zahidi, "An Arabic Dialects Dictionary Using Word Embeddings," *International Journal of Rough Sets and Data Analysis*, vol. 6, no. 3, pp. 18–31, 2019.

[14] M. Polignano, P. Basile, M. de Gemmis, G. Semeraro, and V. Basile, "AlBERTo: Italian BERT language understanding model for NLP challenging tasks based on tweets," *CEUR Workshop Proceedings*, vol. 2481, 2019.

[15] M. Polignano, M. De Gemmis, P. Basile, and G. Semeraro, "A comparison of Word-Embeddings in Emotion Detection from Text using BiLSTM, CNN and Self-Attention," *ACM UMAP 2019 Adjunct - Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, no. June, pp. 63–68, 2019.

[16] B. Mohit, N. Schneider, R. Bhowmick, K. Oflazer, and N. A. Smith, "Recall-Oriented Learning of Named Entities in Arabic Wikipedia," in *Proceedings of EACL*, 2012, pp. 162–173.

[17] K. Shaalan and M. Oudah, "A hybrid approach to Arabic named entity recognition," *Journal of Information Science*, vol. 40, no. 1, pp. 67–87, 2013.

[18] X. Li, H. Zhang, and X. H. Zhou, "Chinese clinical named entity recognition with variant neural structures based on BERT methods," *Journal of Biomedical Informatics*, vol. 107, no. August 2019, p. 103422, 2020. [Online]. Available: https://doi.org/10.1016/j.jbi.2020.103422