Arabic Keyphrase Extraction: Enhancing Deep Learning Models with Pre-trained Contextual Embedding and External Features

Randah Alharbi and **Husni Al-Muhtaseb**
King Fahd University of Petroleum
Minerals (KFUPM)
Dhahran-Saudi Arabia
`g201907330, muhtaseb@kfupm.edu.sa`

Abstract

Keyphrase extraction is essential to many Information retrieval (IR) and Natural language Processing (NLP) tasks such as summarization and indexing. This study investigates deep learning approaches to Arabic keyphrase extraction. We address the problem as sequence classification and create a Bi-LSTM model to classify each sequence token as either part of the keyphrase or outside of it. We have extracted word embeddings from two pre-trained models, Word2Vec and BERT. Moreover, we have investigated the effect of incorporating linguistic, positional, and statistical features with word embeddings on performance. Our best-performing model has achieved 0.45 F1-score on ArabicKPE dataset when combining linguistic and positional features with BERT embedding.

1 Introduction

Keyphrases are the phrases that best represent a document. They play an essential role in many Natural Language Processing (NLP) and Information Retrieval (IR) tasks, such as indexing, summarization, categorization, and opinion mining (Merrouni et al., 2020) (Hasan and Ng, 2014). Manual extraction of keyphrases is time-consuming and requires experts' knowledge; thus, the extraction needs to be automated (Merrouni et al., 2020). Although many studies have been proposed to address automatic keyphrase extraction and generation, the performance is still moderate due to the task's difficulty (Merrouni et al., 2020). Several approaches have been proposed; one of the earliest approaches is the two-step ranking, in which candidate phrases are extracted with several heuristics and then ranked using supervised or un-supervised methods (Hasan and Ng, 2014). Another approach is the classification approach, in which candidate phrases are classified as keyphrases or not (Papagiannopoulou and Tsoumakas, 2020). A more recent approach is formulating keyphrase extraction as a sequence labeling task in which each word in the documents is labeled as part of a keyphrase or not (Alzaidy et al., 2019). Another recent approach is to consider formulating the task as a generation task utilizing sequence-to-sequence models in order to be able to generate keyphrases that are not available in the source text, i.e., keyphrase generation (Meng et al., 2017).

Word embeddings prove their effectiveness in many NLP tasks. Several word embeddings are proposed, such as Word2Vec (Mikolov et al., 2013) and FastText (Bojanowski et al., 2017). The earliest proposed word embeddings generate the same vector for the word regardless of the word context hence called static word embeddings (Pilehvar and Camacho-Collados). Recently, several word embeddings generate different embeddings for the word depending on its context hence called contextualized word embeddings (Pilehvar and Camacho-Collados) such as BERT (Devlin et al., 2019), and ELMo (Peters et al., 2018). Several studies have utilized various types of word embeddings into supervised and unsupervised keyphrase extraction, and they positively affect performance.

Arabic has its own characteristics that pose many challenges on any IR or NLP task (Darwish and Magdy, 2014) (Habash, 2010). Thus, it is crucial to investigate the performance of state-of-the-art techniques of keyphrase extraction on Arabic, which might differ in terms of performance from other languages. Several datasets are available for Arabic keyphrase extraction; The Arabic keyphrase extraction Corpus (AKEC) (Helmy et al., 2016), Arabic Dataset proposed by (Al-Logmani and Al-Muhtaseb), WikiAll [1] from Arabic Wikipedia documents, and ArabicKPE (Helmy et al., 2018) .

During our investigation of the keyphrase extraction studies, we have found that studies on

[1] `https://github.com/anastaw/Arabic-Wikipedia-Corpus`

Arabic keyphrase extraction are falling behind in applying state-of-the-art technologies. For example, only a few studies have utilized word embeddings; Suleiman et al. 2019a have investigated using Word2Vec and semantic similarity to generate keyphrases for three documents only. Helmy et al. 2018 have investigated using Word2Vec and Bidirectional-Long Short Term Memory (Bi-LSTM) in keyphrase extraction. To fill this gap, we aim to apply deep learning approaches to keyphrase extraction utilizing the static and contextualized word embeddings for Arabic keyphrase extraction. Thus, we have formulated the task as a sequence labeling task and have used a Bi-LSTM classifier with token representation extracted from two types of pre-trained word embeddings. Additionally, we aim to investigate the effect of incorporating statistical, positional, and linguistic features with static and contextual embeddings.

In this study, we have used Bidirectional Encoder Representations from Transformers (BERT) in Arabic keyphrase extraction and have compared it to Word2Vec embedding. Additionally, we have investigated three ways of utilizing BERT for Arabic keyphrase extraction; extracting the output of the last encoder of the BERT model and using it as a feature, concatenating the output of the last four encoder layer of BERT, and Fine-tuning BERT. To the best of our knowledge, this is the first study incorporating contextualized word embedding from BERT into the Arabic keyphrase extraction task. We have found that utilizing contextual embeddings vastly enhances the performance of Arabic keyphrase extraction model. Moreover, adding features to the Arabic keyphrase extraction Bi-LSTM model, in general, has a positive effect on the model performance. The rest of the paper is organized as follows: we present the related works in section 2, Section 3 presents our methodology, and section 4 presents experiments results and discussion. Our conclusion is presented in section 5.

2 Related Work

Keyphrase extraction has two general approaches: unsupervised and supervised (Papagiannopoulou and Tsoumakas, 2020). Supervised approaches are powerful and perform better than unsupervised approaches. However, the unsupervised approaches are less expensive (Papagiannopoulou and Tsoumakas, 2020). Several unsupervised keyphrase extraction studies have been conducted. (Campos et al., 2020) have proposed YAKE!, an unsupervised keyphrase extraction system based on statistical features extracted from a single document. Their approach depends on six features; term frequency within the document, normalized term frequency, term relative position (sentence index), term relatedness to context, term case, and how often a term appears in different sentences (term different sentence). They have evaluated the system on 20 datasets in five languages. YAKE! proved its effectiveness generally compared to other systems with large text and performed well with shorter text on different domains and different document types. Moreover, the frequency feature has a more positive impact while removing term relatedness from the context and term different sentence features improves performance. Meanwhile, the term frequency feature is more useful when the document size increases, while the position feature is more beneficial in shorter texts. The case feature is more useful with mid to larger documents, while the term different sentence feature is better with short to mid documents. (Zhang et al., 2020) have leveraged word embedding for unsupervised graph-based keyphrase extraction. Their model selects candidate words based on their Part-Of-Speech (POS) tag; they have only selected nouns and adjectives. They have built three graphs; a word-word graph based on the word co-occurrence, a word-topic graph that connects words to their topics, and a topic-topic graph that is constructed when the same word appears in different topics. They have also proposed a modified random-walk model to rank candidate words and a new scoring model for candidate phrases based on the cosine similarity of the generated word embedding and the modified page rank score. The top scoring phrases are considered document keyphrases. Evaluating the type of word embeddings used shows that their embedding outperforms other embeddings on this task. They have reported that their model performs the best on all the tested datasets compared to other baselines. (Zu et al., 2020) have utilized word embedding with graph-based unsupervised keyphrase extraction along with document embedding. They have used a pre-trained Sent2Vec (Pagliardini et al., 2018) model trained on Wikipedia to create word embedding. The embedding vector is created by averaging all document words and n-gram embeddings. They have found that using the word as a node is better when dealing with a short text dataset and using a phrase as a node is better when dealing with a long text dataset.

Several studies have formulated the keyphrase extraction task as a sequence labeling task. (Basaldella et al., 2018) have proposed a deep learning model for automatic keyphrase extraction using Bi-LSTM and pre-trained GloVeembedding (Pennington et al., 2014). Their model outperforms CopyRNN (Meng et al., 2017) model on the same dataset. (Alzaidy et al., 2019) have utilized Bi-LSTM and CRF for keyphrase extraction from scientific documents using 100-dimension pre-trained GloVeembedding for embedding initialization. They have studied the role of each model layer on the performance and have compared CRF only, forward-LSTM, and Bi-LSTM. They have found that removing the Bi-LSTM layer negatively affects the recall, while removing the CRF layer increases the recall and decreases the precision. Hence, it indicates that Bi-LSTM can capture long-distance semantics and cause extraction of more gold-standard keyphrase. They have also found that CRF can capture the dependencies between labels leading to higher model precision. Combining Bi-LSTM and CRF has the best performance among the three created models. Additionally, the model outperforms CopyRNN (Meng et al., 2017). Many studies have used encoder-decoder architecture to generate absent and present keyphrases. The most popular study is the work by (Meng et al., 2017). They have proposed a generative model for keyphrase generation based on encoder-decoder architecture. They have used Bidirectional Gated Recurrent Units (Bi-GRU) for the encoder and forward-GRU for the decoder and incorporated attention mechanism (Bahdanau et al., 2015) (RNN-model) and copy mechanism (Gu et al., 2016) (CopyRNN-model) to deal with out-of-vocabulary words. CopyRNN model outperforms all the models they have compared by an average of almost 20% (Meng et al., 2017). Moreover, the CopyRNN model outperforms RNN in predicting both present and absent keyphrases. (Kehua Yang, 2019) encoder-decoder model is entirely based on the self-attention mechanism. They have incorporated semantic similarity between keyphrases. The model outperforms all baselines in predicting the present keyphrase. Additionally, it outperforms CopyRNN in predicting absent keyphrases. Targeting the problem of overlapping phrases generated by sequence-to-sequence models, (Zhao and Zhang, 2019) have proposed (ParaNet). The model consists of two parallel encoders; one to encode the text and the other to encode the linguistic constraints introducing coverage attention. They have used multi-task learning on two parallel decoders to generate the keyphrase and POS tag for each word in the keyphrase. They have tested different settings for combining the vector of the words and their syntactic tags, using the hyperbolic tangent function, using tree-LSTM, and adding coverage attention to the previous two. On the evaluation of present keyphrases, all their model settings outperform the extraction and generation methods baselines, including CopyRNN (Meng et al., 2017). Their best performing setting is when using tree-LSTM to combine vectors along with coverage attention.

Several studies have used BERT contextualized word embedding in two strategies; feature-based strategy or fine-tuning-based strategy. Word feature is extracted from the pre-trained BERT in the feature-based strategy. In fine-tuning based strategy, BERT model parameters are fine-tuned with the new smaller dataset for the downstream task adding one fully connected layer on top of it (Devlin et al., 2019). (Sun et al., 2020) have utilized BERT embedding in multi-task learning for keyphrase extraction. (Lim et al., 2020) have fine-tuned BERT and SciBERT (Beltagy et al., 2019) for keyphrase extraction. They have found that the best performance happens within the first three epochs of fine-tuning and that SciBERT performs better than BERT on scientific datasets. (Dascalu and Trăuşan-Matu, 2021) have experimented with four neural network architectures based on Bi-LSTM and multi-head attention on top of the transformer models BERT and SciBERT. A recent study has combined graph embedding and BERT embedding for keyphrase extraction is PhraseFormer (Nikzad-Khasmakhi et al., 2021). They have concatenated the resulting graph embedding and word embedding for each word and have used the resulting encoding as input. Another way of utilizing BERT for keyphrase extraction using a feature-based technique is using it in ranking candidate phrases (Mu et al., 2020). (Ding et al., 2021) have incorporated different types of features with BERT extracted features for the Chinese medical keyphrase extraction. The task is considered as a character-level labeling task. They have incorporated POS feature and lexicon feature using two techniques: concatenation and feature embedding. They have used the model without feature as a baseline and have tested the effect of features (POS only, lexicon only, and their combination) and the effect of different feature incorporation techniques (concatenation, embedding, and their combination). Their results show that incorporating the lexicon feature has a more positive impact than the POS feature, regardless of the incorporation techniques. Furthermore, the best incorporation technique is the embedding technique. (Sahrawat et al., 2020) have utilized contextualized word embeddings and compared them to static word embedding in sequence labeling keyphrase extraction. They have used Bi-LSTM-CRF and Bi-LSTM architectures with several embeddings; BERT, SciBERT, ELMo, TransformerXL (Dai et al., 2019), OpenAI-GPT (Radford and Narasimhan, 2018), OpenAI-GPT2 (Radford et al., 2019), RoBERTa (Liu et al., 2019), Glove, FastText, and Word2Vec. They have found that contextualized embeddings are better than static embedding, and BERT is the best among them since it uses bi-directional training.

Studies on Arabic keyphrase extraction followed several approaches. Rule-based approaches (El-Beltagy and Rafea, 2009) (Rammal et al., 2015)(Najadat et al., 2016)(Loukam et al., 2019) (Alotaibi and Ahmad, 2019) (Musleh et al., 2019), ranking approaches (Basaldella et al., 2017) (Amer and Foad, 2017), using a graph-based model as a base for ranking (Halabi and Awajan, 2019) (Al Hadidi et al., 2019), utilizing bag-of-concept (Awajan, 2015) (Suleiman and Awajan, 2017) (Suleiman et al., 2019b), machine learning approaches (Ali and Omar, 2015) (Armouty and Tedmori, 2019) (Al Etaiwi et al., 2019) and deep learning approaches (Helmy et al., 2018). (Ali and Omar, 2015) have combined statistical and machine learning methods and have formulated the keyphrase extraction task as a classification task. They have used term frequency, first occurrence, sentence count, c-value for multi-word nested terms, and TF-IDF statistical features to construct a feature vector. They have trained linear logistic regression, linear discriminant analysis, and support vector machine (SVM) classifiers. (Armouty and Tedmori, 2019) have used TF-IDF and the first occurrence weight of the term with Support Vector Machine (SVM), Naïve Bayes, and Random Forest classifiers. (Al Etaiwi et al., 2019) have used graph centrality measures along with term frequency and POS tags as input features to multi-layer perceptron, Naïve Bayes, Random Forest, and OneR algorithms. (Helmy et al., 2018) have proposed a deep learning-based model and a large-scale dataset for keyphrase extraction task. They have used AraVec (Soliman et al., 2017) to represent each token and a Bi-LSTM model.

3 Methodology

3.1 Data Preprocessing

The text is tokenized using Stanford Stanza neural pipeline for Arabic[2] and processed to remove punctuation, normalize all forms of Alef [إآأا] into plain Alef [ا] and decorated kaf [گ] to kaf [ك], and replace numbers' digits with a token to represent numbers which is [رقم] (number). Moreover, three types of features are extracted to be incorporated with the embeddings; linguistic feature (part-of-speech for each token), positional features (first occurrence and the sentence order of the first occurrence), and statistical feature (Term Frequency-Inverse Document Frequency-TFIDF). The part-of-speech tags are extracted using the MLE disambiguator of Camel tool (Obeid et al., 2020) and the TFIDF using TFIDF vectorizer from the scikit-learn library[3]. The actual value of the positional features and statistical feature and the one-hot encoding vector of the linguistic feature are concatenated to the end of the word embedding. Finally, the data is processed to be suitable for the sequence labeling task by converting the document into a sequence of tokens labeled with 1 if it is part of a keyphrase and with 0 if it is out of the keyphrase. Moreover, the maximum document length considered is 512 tokens for all models.

3.2 Models' specifications

A Bi-LSTM token classifier is built with one bidirectional LSTM layer that accepts input from the embedding layer and has one dense layer to generate the output label. There are two settings for the model input; the first is word embedding only, and the other is word embedding concatenated with different individual features or combined features. Figure 1 shows the model architecture. Two types of pre-trained word embeddings are used; static word embedding and contextualized word embedding, which are AraVec pre-trained embedding (Soliman et al., 2017) and AraBERT v2 (Antoun

[2]https://stanfordnlp.github.io/stanza/
[3]https://scikit-learn.org/stable/

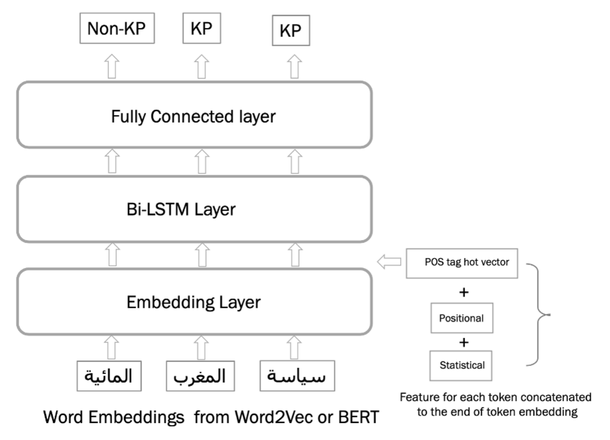

Figure 1: Proposed Model Architecture

Word2vec Embedding dimension	100
BERT Embedding dimension	768
Bi-LSTM hidden unit	150
POS tag embedding dimension	26
Learning rate	1.00E-05
Loss function	Cross entropy loss
Batch Size	1
Optimizer	SGD
Epoch for BERT	5
Epoch for Word2Vec	10
AraBERT pretrained Model	bert-base-arabertv02
Word2vec pretrained model	full_uni_sg_100_wiki
Maximum Document length	512

Table 1: Models Specifications

et al., 2020), respectively. We have used precision, recall, and F1-score metrics to evaluate the model performance on the level of extracted keywords and the extracted keyphrases. We have rewarded the model for each correctly extracted keyword at the keyword level, even if the model has generated part of the keyphrase. In contrast, at the keyphrase level, we have rewarded the model if it has generated the entire exact keyphrase. We have not stemmed the keywords before testing.

3.3 Experiments setup

The used dataset is ArabicKPE (Helmy et al., 2018) with the same splits provided by the authors; 4887 documents for training, 944 for model validation, and 941 for testing. In addition, Word2Vec and BERT have been used with Bi-LSTM and different features combinations. Further experiments with BERT include concatenating the last four hidden layers of BERT used as inputs to Bi-LSTM and fine-tuning BERT for keyphrase extraction on the used dataset. We have tested each feature independently and have combined two, three, and four features. Pytorch library[4] is used to build the models. The same hyper-parameters are used for all experiments as specified in Table 1. The Experiments are conducted using Google Colab Pro+ with GPU. Due to memory constraints and the large model size, the batch size is set to 1.

4 Results and Discussion

4.1 Using no features experiments

Table 2 and 3 present the results of using AraVec (Soliman et al., 2017) and AraBERT (Antoun et al., 2020) without features and our baseline of using no pre-trained embedding and no features. The results clearly show the benefit of using pre-trained word embedding compared to the baseline. Using pre-trained word embeddings enhances the model performance over the baseline in terms of F1-score for keyphrase level by 0.03 and 0.23 for Word2Vec and BERT, respectively. Moreover, Contextualized word embedding (BERT) has vastly enhanced the performance compared to static word embedding (Word2Vec) by 0.20.

4.2 Results of different BERT settings

In Table 3, we present the results of different settings of using BERT. The results show that using Bi-LSTM with embedding extracted from BERT has a slightly better F1-score than fine-tuning the BERT model for keyphrase extraction. This might be due to the ability of the model to learn more context utilizing Bi-LSTM. Moreover, unlike (Devlin et al., 2019) suggestion, using the output of the last encoder layer has a slightly better effect on performance than concatenating the output of the last four layers. This might be attributed to the difference in the language used to train the BERT model; different languages might have different behavior regarding choosing the best layer from the twelve encoder layers. Another possible reason is the difference in the tested downstream task as they test for the Named Entity Recognition (NER) task. Hence, for our task and language choice, it is better to use the output of the last encoder layer only to reduce the dimensionality of the input vector.

[4]https://pytorch.org/

Word2Vec Model Name	Keyword-Wise			Keyphrase-Wise		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Bi-LSTM-Baseline	0.50	0.38	0.43	0.20	0.21	0.20
No added feature	0.64	0.33	0.44	0.28	0.19	0.23
POS	0.63	0.33	0.43	0.29	0.19	0.23
TFIDF	0.65	0.38	0.48	0.24	0.20	0.22
First Occurrence	0.57	0.41	0.47	0.30	0.25	0.27
Sentence Order	0.64	0.42	0.51	0.28	0.25	0.27
POS + TFIDF	**0.69**	0.34	0.45	0.28	0.18	0.22
POS + First occurrence	0.68	0.39	0.50	**0.31**	0.23	0.27
POS + Sentence order	0.60	**0.49**	**0.54**	0.28	**0.30**	**0.29**
TFIDF+ First occurrence	0.63	0.43	0.51	0.26	0.25	0.26
TFIDF+ sentence order	0.66	0.37	0.47	0.24	0.19	0.21
First occurrence + sentence order	0.68	0.40	0.50	0.29	0.24	0.26
TFIDF+ First occurrence + First sentence order	0.65	0.42	0.51	0.26	0.23	0.25
POS+ First occurrence + First sentence order	0.65	0.43	0.51	0.29	0.26	0.27
POS+ TFIDF+ First occurrence	0.67	0.38	0.48	0.26	0.21	0.23
POS+ TFIDF+ Sentence order	0.65	0.44	*0.52*	0.30	0.26	*0.28*
POS+ TFIDF+ First occurrence + First sentence order	0.64	0.41	0.50	0.27	0.24	0.25

Table 2: Word2Vec Experiments' results

4.3 Results of adding features:

We have tested the effect of incorporating the raw positional and statistical features' values to the embedding of each token and the one-hot 26 dimensions vector of the POS feature to the end of each word embedding.Table 2 and Table 3 show the results of adding each feature to Word2Vec and BERT embeddings respectively.

4.3.1 Independent features

In Word2Vec experiments, the results show that the positional features have the most impact on the performance in terms of the F1-score for the keyphrase level. Moreover, TFIDF has decreased the performance by 0.01 for the keyphrase level. Meanwhile, TFIDF has increased the performance over the no-feature model at the keyword level by 0.04. Adding POS tag features unexpectedly has no effect on the keyphrase level's performance and has decreased the keyword level's performance. In contrast, in BERT experiments, all features have a slightly positive impact on performance over the no-features model in terms of F1-score and recall of keyphrase level and recall of keyword level. Meanwhile, all features have not impacted performance regarding the F1-score of the keyword level. This slight improvement or no improvement in performance might be attributed to the fact that BERT already learned that information during the pr-training phase.

4.3.2 Combination of features

- **Combing two features**: In Word2Vec experiments, the best features combined with POS are the sentence order and the first occurrence. Combining TFIDF with POS has decreased the performance in terms of F1-score and recall for keyphrase level evaluation. However, it has increased the performance in terms of F1-score and recall at the keyword level. Combining numerical features reveals that combining TFIDF with sentence order has decreased the performance of the F1-score at the keyphrase level but has increased it on the keyword level. The best combination of two features in BERT experiments is when combining the POS feature with the first occurrence. Like Word2Vec, adding TFIDF to POS features has decreased the F1-score performance for the keyphrase level. Combining numerical features does not improve the performance, unlike when each independent numerical feature is used. That might result from combining features without normalizing them to have the same mean leading to some noise. Moreover, we can notice that in the context of keyword level evaluation, all F1-score results can be rounded to 0.60.

BERT Model Name	Keyword-Wise			Keyphrase-Wise		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Bi-LSTM-Baseline	0.50	0.38	0.43	0.20	0.21	0.20
No added feature	**0.60**	0.59	0.59	**0.43**	0.43	0.43
Fine-Tuned	0.51	0.67	0.58	0.37	0.49	0.42
No-feature -4layer	0.56	0.59	0.58	0.41	0.44	0.42
POS	0.56	0.63	0.59	0.42	0.46	0.44
TFIDF	0.55	0.64	0.59	0.41	0.48	0.44
First Occurrence	0.53	0.67	0.59	0.40	0.51	**0.45**
Sentence Order	0.51	**0.69**	0.59	0.39	**0.52**	**0.45**
POS + TFIDF	0.56	0.60	0.58	0.42	0.43	0.42
POS + First occurrence	0.53	**0.69**	**0.60**	0.41	0.51	**0.45**
POS + Sentence order	0.51	**0.69**	0.59	0.38	**0.52**	0.44
TFIDF+ First occurrence	0.58	0.60	0.59	0.42	0.43	0.43
TFIDF+ sentence order	0.55	0.62	0.58	0.41	0.45	0.43
First occurrence + sentence order	0.55	0.64	0.59	0.40	0.46	0.43
TFIDF+ First occurrence + First sentence order	0.58	0.59	0.59	0.42	0.42	0.42
POS+ First occurrence + First sentence order	0.56	0.62	0.59	0.41	0.46	0.43
POS+ TFIDF+ First occurrence	0.56	0.62	0.59	0.42	0.45	0.44
POS+ TFIDF+ Sentence order	0.56	0.62	0.59	0.41	0.45	0.43
POS+ TFIDF+ First occurrence + First sentence order	0.54	0.65	0.59	0.39	0.47	0.43

Table 3: BERT Experiments' results

- **Combining three features**: In Word2Vec experiments, all three features combination has improved the performance in terms of F1-score for keyphrase level except for (POS+TFIDF+first occurrence) combination, which does not change the performance of keyphrase level but increases the performance of the keyword level. In contrast, BERT experiments show that the models have the same performance in terms of F1-score of the keyword level. While in keyphrase level performance, combining (POS+TFIDF+ First Occurrence) has slightly increased the performance, and combining (TFIDF+First occurrence+First sentence order) has slightly decreased the performance in terms of F1-score.

- **Combining four features**: BERT model has the same results as using no features on both keyphrase and keyword levels. On the other hand, Word2Vec has benefited by 0.02 and 0.06 F1-score for keyphrase level and keyword level, respectively, compared to using no features experiments.

4.4 Comparison with others' work

Table 4 shows our results compared to (Helmy et al., 2018). They have used deep learning with AraVec pre-trained word embedding (Soliman et al., 2017). Additionally, they have reported their results on the same dataset at the top 5, 10, and 15 retrieved keyphrases. They have compared the lemmatized version of the gold keyphrase with the lemmatized version of the predicted keyphrase. We have chosen to compare our results to their top 15 results since the maximum number of keyphrases available on the test set is 13 keyphrases for the document, and all of them will be included in our and their results. Moreover, they have not mentioned the used stemmer, and we have used ISRIStemmer from the NLTK library[5] to stem keywords. The results show that using no feature on Word2Vec has a similar F1-score to their model and that the best performing model on our Word2Vec experiments has outperformed their model due to incorporating features to Word2Vec embedding. Moreover, using BERT embedding without features and BERT's best performing model has vastly outperformed their model by 0.21 and 0.24, respectively.

4.5 Discussion

The best performing model on both level keyword level and keyphrase level for Word2Vec is when combining POS with the sentence order feature fol-

[5]https://www.nltk.org/

Model Name	Keyword-wise			Keyphrase-wise		
	Precision	Recall	F1-score	Precision	Recall	F1-score
(Helmy et al., 2018)@15KP	-	-	-	0.16	**0.67**	0.26
Word2Vec-no feature	**0.68**	0.35	0.46	0.30	0.21	0.25
BERT-no feature	0.63	0.62	0.63	**0.48**	0.47	0.47
Word2Vec -Best	0.63	0.52	0.57	0.32	0.34	0.33
BERT-Best	0.57	**0.71**	**0.64**	0.45	0.56	**0.50**

Table 4: Comparing the results with previous work

lowed by combining POS with TFIDF and sentence order. Conversely, the model with the least performance at the keyphrase level is when combining TFIDF with sentence order features. In general, it seems that using the TFIDF feature or combining it with other features degraded the model learning. In contrast, the model that has the least performance on the keyword level is when using the POS feature. In General, all BERT experiments have similar performance in both keyphrase and keyword levels. The keyphrase level scores differ by 0.03 only and range between 0.42 to 0.45, and all keyword level scores can be rounded to 0.60. The best performing models on BERT are when using first occurrence features alone, sentence order features alone, and when combining POS feature with first occurrence. While the least performing models are when combining TFIDF, first occurrence, and sentence order features and combining POS and TFIDF features. We can notice that the performance on the keyphrase level is not affected on 6 features combinations experiments out of 11, i.e., it is the same as no feature model. This might prove that BERT can encode linguistic and statistical features during pre-training. Further investigation and model propping are needed to confirm this finding. Generally, the improvements for both evaluation levels are aligned in BERT experiments. On the other hand, some Word2Vec experiments, which are TFIDF, POS+TFIDF, and TFIDF+Sentence order, have different behavior; the increased performance at the keyword level might be accompanied by a decreased performance at the keyphrase level. We can notice that TFIDF feature is available in all these experiments, which suggests that this feature might be beneficial to identifying the keyword more than the keyphrase. Comparing the gap between the scores of keyword level and keyphrase level on both Word2Vec and BERT, we notice that the difference between the two levels on BERT is smaller than the difference between the two levels on Word2Vec. This can be attributed to BERT's ability to generate the correct entire keyphrase due to more contextual information considered when giving context-dependent embedding compared to Word2Vec, which gives the same embedding for the word in different contexts. It seems that using Word2Vec enables models to recognize that the word is part of the keyphrase but could not present these words in the correct order. Additionally, we have noticed that different features combinations have different effects on performance depending on the embedding type. For example, BERT embedding based models are positively affected by the combination that includes the first occurrence feature more than the sentence order feature. In contrast, Word2Vec embedding based models are positively affected by the sentence order feature more than the first occurrence.

5 Conclusion

This study uses two types of pre-trained word embeddings for Arabic keyphrase extraction task: static and contextualized word embedding (Word2Vec and BERT). Several features are incorporated into the models to test their effect on performance. We have found that contextualized word embedding has vastly enhanced the performance of Arabic keyphrase extraction. Moreover, incorporating features with static embedding has more effect than incorporating features with contextualized embedding. Different features and features combinations affect the performance differently depending on the used embeddings. For future work, we consider trying the effect of adding more features to the models. Moreover, investigate the best combination of layers to select from BERT for keyphrase extraction.

Limitations

First, we have adopted a strict evaluation metric at the keyphrase level, which only rewards the correct keyphrase with the same keyword order and keyword numbers, i.e., we do not reward the model if it over generates a word in the middle of the keyphrase. This might affect the reported performance. Therefore, less strict metrics that consider stemming or word similarity might be helpful. Second, we broadcast the value of the POS for the unknown words that BERT decides to segment into sub-words which might introduce some noise to the training that might affect the performance. Nevertheless, trying not to broadcast the value does not affect the performance. Third, the randomness introduced on PyTorch run time execution with GPU setting might affect the ability to reproduce the same results when repeating the experiments. The model size and the time needed to model training have been challenging. Although we are using a GPU subscription with google colab, the run has taken a long time, and we have run out of drive space.

References

Wael Al Etaiwi, Arafat A. Awajan, and Dima Suleiman. 2019. Keywords extraction from arabic documents using centrality measures. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 237–241.

Meran M. Al Hadidi, Muath Alzghool, and Hasan Muaidi. 2019. Keyword extraction from arabic text using the page rank algorithm. *International Journal of Innovative Technology and Exploring Engineering*, 8(12):3495–3504.

Mohammed Al-Logmani and Husni Al-Muhtaseb. Arabic dataset for automatic keyphrase extraction. In *Second International Conference on Software Engineering (SOEN-2017)*, pages 217–222.

Nidaa Ghalib Ali and Nazlia Omar. 2015. A hybrid of statistical and machine learning methods for Arabic keyphrase extraction. *Asian Journal of Applied Sciences*, 8(4):269–276.

Fahad Mazaed Alotaibi and Shakeel Ahmad. 2019. Keywords Extraction from the Text of Holy Quran Using Linguistic and Heuristic Rules. *International Journal of Computer Science and Network Security*, 19(2):82–87.

Rabah Alzaidy, Cornelia Caragea, and C. Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The World Wide Web Conference*, WWW '19, page 2551–2557, New York, NY, USA. Association for Computing Machinery.

Eslam Amer and Khaled Foad. 2017. Akea: An arabic keyphrase extraction algorithm. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016*, pages 137–146, Cham. Springer International Publishing.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Batool Armouty and Sara Tedmori. 2019. Automated keyword extraction using support vector machine from arabic news documents. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 342–346.

Arafat Awajan. 2015. Keyword extraction from arabic documents using term equivalence classes. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 14(2).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Marco Basaldella, Elisa Antolli, Giuseppe Serra, and Carlo Tasso. 2018. Bidirectional lstm recurrent neural network for keyphrase extraction. In *Digital Libraries and Multimedia Archives*, pages 180–187, Cham. Springer International Publishing.

Marco Basaldella, Muhammad Helmy, Elisa Antolli, Mihai Horia Popescu, Giuseppe Serra, and Carlo Tasso. 2017. Exploiting and evaluating a supervised, multilanguage keyphrase extraction pipeline for under-resourced languages. *International Conference Recent Advances in Natural Language Processing, RANLP*, 2017-Septe(1998):78–85.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Kareem Darwish and Walid Magdy. 2014. Arabic Information Retrieval. *Foundations and Trends® in Information Retrieval*, 7(4):239–342.

Cristian Dascalu and Ştefan Trăuşan-Matu. 2021. Experiments with contextualized word embeddings for keyphrase extraction. In *2021 23rd International Conference on Control Systems and Computer Science (CSCS)*, pages 447–452.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liangping Ding, Zhixiong Zhang, and Yang Zhao. 2021. Bert-based chinese medical keyphrase extraction model enhanced with external features. In *Towards Open and Trustworthy Digital Societies: 23rd International Conference on Asia-Pacific Digital Libraries, ICADL 2021, Virtual Event, December 1–3, 2021, Proceedings*, page 167–176, Berlin, Heidelberg. Springer-Verlag.

Samhaa R. El-Beltagy and Ahmed Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, 34(1):132–144.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Nizar Y. Habash. 2010. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Dana Halabi and Arafat Awajan. 2019. Graph-Based Arabic Key-phrases Extraction. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, Amman, Jordan. IEEE.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273, Baltimore, Maryland. Association for Computational Linguistics.

Muhammad Helmy, Marco Basaldella, Eddy Maddalena, Stefano Mizzaro, and Gianluca Demartini. 2016. Towards building a standard dataset for arabic keyphrase extraction evaluation. In *2016 International Conference on Asian Language Processing (IALP)*, pages 26–29.

Muhammad Helmy, R.M. Vigneshram, Giuseppe Serra, and Carlo Tasso. 2018. Applying deep learning for arabic keyphrase extraction. *Procedia Computer Science*, 142:254–261. Arabic Computational Linguistics.

Wei Zhang Jiqing Yao Yuquan Le Kehua Yang, Yaodong Wang. 2019. Keyphrase generation based on self-attention mechanism. *Computers, Materials & Continua*, 61(2):569–581.

Yeonsoo Lim, Deokjin Seo, and Yuchul Jung. 2020. Fine-tuning bert models for keyphrase extraction in scientific articles. *JOURNAL OF ADVANCED INFORMATION TECHNOLOGY AND CONVERGENCE*, 10:45–56.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Mourad Loukam, Djamila Hammouche, Freha Mezzoudj, and Fatma Zohra Belkredim. 2019. Keyphrase extraction from modern standard arabic texts based on association rules. In *Arabic Language Processing: From Theory to Practice*, pages 209–220, Cham. Springer International Publishing.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

Zakariae Alami Merrouni, Bouchra Frikh, and Brahim Ouhbi. 2020. Automatic keyphrase extraction: a survey and trends. *Journal of Intelligent Information Systems*, 54(2):391–424.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Funan Mu, Zhenting Yu, LiFeng Wang, Yequan Wang, Qingyu Yin, Yibo Sun, Liqun Liu, Teng Ma, Jing Tang, and Xing Zhou. 2020. Keyphrase extraction with span-based feature representations.

Dhiaa Musleh, Rashad Ahmed, Atta Rahman, and Fahd Al-Haidari. 2019. A novel approach to arabic keyphrase extraction. *ICIC Express Letters*, 10:875–884.

Hassan Najadat, Ismail Hmeidi, Mohammed N. Al-Kabi, and Maysa Mahmoud Bany Issa. 2016. Automatic keyphrase extractor from arabic documents. *International Journal of Advanced Computer Science and Applications*, 7.

Narjes Nikzad-Khasmakhi, Mohammad-Reza Feizi-Derakhshi, Meysam Asgari-Chenaghlu, Mohammad-Ali Balafar, Ali-Reza Feizi-Derakhshi, Taymaz Rahkar-Farshi, Majid Ramezani, Zoleikha Jahanbakhsh-Nagadeh, Elnaz Zafarani-Moattar, and Mehrdad Ranjbar-Khadivi. 2021. Phraseformer: Multimodal key-phrase extraction using transformer and graph embedding.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 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, Marseille, France. European Language Resources Association.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2020. A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):1–45.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. *Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning*. Springer Cham.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Mahmoud Rammal, Zeinab Bahsoun, and Mona Jabbour. 2015. Keyword extraction from arabic legal texts. *Interactive Technology and Smart Education*, 12:62–71.

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In *Advances in Information Retrieval*, pages 328–335, Cham. Springer International Publishing.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R. El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265. Arabic Computational Linguistics.

Dima Suleiman and Arafat Awajan. 2017. Bag-of-concept based keyword extraction from Arabic documents. *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, pages 863–869.

Dima Suleiman, Arafat A. Awajan, and Wael al Etaiwi. 2019a. Arabic text keywords extraction using word2vec. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pages 1–7.

Dima Suleiman, Arafat A. Awajan, and Wael Al Etaiwi. 2019b. Arabic Text Keywords Extraction using Word2vec. *2019 2nd International Conference on New Trends in Computing Sciences, ICTCS 2019 - Proceedings*.

Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert. *CoRR 2020*, abs/2004.13639.

Yuxiang Zhang, Huan Liu, Suge Wang, W.H. Ip, Fan Wei, and Chunjing Xiao. 2020. Automatic keyphrase extraction using word embeddings. *Soft Computing*, 24:1–16.

Jing Zhao and Yuxiang Zhang. 2019. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5224–5233, Florence, Italy. Association for Computational Linguistics.

Xian Zu, Fei Xie, and Xiaojian Liu. 2020. Graph-based keyphrase extraction using word and document em beddings. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pages 70–76.