2023 4th IEEE Global Conference for Advancement in Technology (GCAT)
Bangalore, India. Oct 6-8, 2023

Comparative Analysis of Embedding Models for Keyphrase Extraction: A KeyBERT-Based Approach

Nimisha MR
Department of Electronics and Communication Engineering
BMS College of Engineering
Bangalore,India
nimisha.ec20@bmsce.ac.in

Shamitha T
Department of Electronics and Communication Engineering
BMS College of Engineering
Bangalore,India
shamitha.ec20@bmsce.ac.in

Dr .Geetishree Mishra
Department of Electronics and Communication Engineering
BMS College of Engineering
Bangalore,India
geetishreemishra.ece@bmsce.ac.in

2023 4th IEEE Global Conference for Advancement in Technology (GCAT) | 979-8-3503-0525-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/GCAT59970.2023.10353319

***Abstract*—Key phrase extraction is a fundamental task in information management, which is often used as a preliminary step in various information retrieval and natural language processing tasks. Embedding models achieve higher F-scores than graph-based models in key phrase extraction for short documents compared to longer documents. Therefore, they are suitable for real-time semantic processing of short textual data. With a vast set of embedding models dedicated to key phrase extraction, it becomes a tedious task to overview and compare the performance of each model. Hence, we have carried out an extensive comparative study of four prominent pre-trained embedding models, namely Sentence Transformers, Flair, spaCy, and Universal Sentence Encoder, that have been used in conjunction with Key-BERT. The results of the experiments conducted on this corpus show that the Sentence Transformer "*all-MiniLM-L6-v2*" version outperforms the other embedding models listed above, with respect to BERT score evaluation and adaptability to transfer learning. Notably, spaCy exhibits fast processing while maintaining satisfactory accuracy. An extensive comparison of Key-BERT-based embedding models suited to texts of different sizes and a detailed qualitative analysis makes the case for our proposed methodology.**

Keywords— Key-BERT, Embedding models, Key phrase.

I. INTRODUCTION

In a digitalisation driven world, we are witnessing a huge growth in unstructured data. Text data such as social media opinions, tweets, digital documents, and blogs are growing over the internet very fast. To leverage and reap the benefits of the growing text data, automating the task of capturing the importance of text and representing them in a succinct way is a popular area of research in Natural Language Processing. One method of representing a large text in a succinct way is representing them by keywords and key phrases. Key phrase extraction is a crucial component when gleaning real-time insights from large amounts of Web and social media data. In this case, the extraction must be fast, and the key phrases must be disjoint. Most existing systems are slow and plagued by overgeneration, i.e., extracting redundant key phrases. While it is easy to extract keywords and key phrases from long corpus, it is difficult to extract the same from a shorter sentence [4]. There are several proposed algorithms which successfully extract keywords from long sentence corpuses, however, their performance is comparatively unsatisfactory for short sentences. Key BERT is a minimal and easy-to-use keyword extraction library that leverages embeddings from BERT-like models to extract key phrases from short sentences as well. Unlike other methods available for keyword generation like Rake and YAKE, KeyBERT uses BERT-embeddings and simple cosine similarity to find the sub-phrases in a document that are most like the document itself. KeyBERT supports many embedding models that can be used as an underlying architecture. Some of them are Sentence-Transformers, Flair, spaCy and Universal Sentence Encoder. All the above-mentioned models are pre-trained on more than 1 billion training pairs and are designed as general-purpose models. With a vast set of such embedding models, each using different types of architectures, it becomes a tedious task to overview the pros and cons of each model and choose an appropriate one for our need. In this paper we have done an extensive comparison of all the above-mentioned embedding models in terms of BERT score, speed or inference time and adaptability to transfer learning scenarios. The above-mentioned models have been specifically chosen due to their high accuracy in extracting embeddings for text classification tasks. Our work leverages the Key-BERT library and opens a way to further curtail the research on a comparative outlook towards different embedding models.

This paper is organised as follows. Related work on key phrase extraction and sentence embedding models is presented in Section II. In Section III, we present the methodology adopted. The comparative analysis of the four embedding models with results is then described in Section IV. Section V concludes the work carried out and highlights the future scope.

II. LITERATURE SURVEY

Here, a literature survey on unsupervised key phrase extraction methods, word and sentence embeddings and previous work pertaining to comparison of different key phrase extraction techniques has been mentioned and our improvements have been highlighted.

A. Unsupervised Keyword Extraction

Unsupervised key phrase extraction comes in two variants: corpus-dependent [1] and corpus-independent [3]. Corpus-independent methods require no other inputs than the one document from which key phrases must be extracted. Most such existing methods are graph-based, with the notable exceptions of KeyCluster[2] and TopicRank[2]. In graph-based key phrase extraction, first introduced with TextRank[5] , the target document is a graph, in which nodes represent words and edges represent the co-occurrence of the two endpoints inside a window. Recently,

WordAttractionRank[7] followed an approach similar to SingleRank, with the difference of using a new weighting scheme for edges between two words, to incorporate the distance between their word embedding representation. Scoring a candidate phrase as the aggregation of its word score has been used in [4] and [7]. However, it leads to over-generation errors. In addition, focusing on individual words hurts the diversity of the results. Therefore, all the pre-trained models that we have selected for our use case are not trained to focus on individual words, and hence avoid this error.

B. Word and Sentence Embeddings

Word embeddings mark a very impactful advancement in representing words as vectors in a continuous vector space. Representing words with vectors in moderate dimensions solves several major drawbacks of the classic bag-of-words representation, including the lack of semantic relatedness between words and the very high dimensionality (size of the vocabulary). Different methods are needed for representing entire sentences or documents. Sent2Vec [3] uses word n-gram features to produce sentence embeddings. It produces word and n-gram vectors specifically trained to be additively combined into a sentence vector, as opposed to general word vectors. Sent2Vec features much faster inference than Paragraph Vector [2]. Similarly, Sentence Transformer model reflects semantic relatedness between phrases when using standard similarity measures on the corresponding vectors. This property is at the core of our method, as we show it outperforms competing embedding methods for key phrase extraction.

C. Key-BERT and Embedding models

Several studies have compared and evaluated different keyword extraction techniques, including KeyBERT, against benchmark datasets. Text summarization and keyword extraction becomes slightly more difficult to accomplish with sentences of very short length as in our corpus. The authors of [6] introduced PKDE (Pre-trained Key phrase Detection Embedding), a graph-based embedding model for key phrase extraction. These recent research papers highlight the exploration and development of embedding-based approaches, however none of them draw comparisons between embedding models based on performance. The authors of [4] compare automatic keyword extraction techniques with human-generated tags, providing insights into the effectiveness and limitations of different approaches. However, they do not report precision and recall values for their system, but our implementation has yielded Precision, Recall and F1 scores. The work in [6] investigates the relationship between keyword extraction and terminology extraction and presents a comparative analysis of various key phrase extraction methods. However, it is a very broad comparison of graph based, statistical, deep learning-based methods. We have improvised on this by dwelling deeper into each embedding model and hence our results are more insightful specifically about the deep learning-based framework which is KeyBERT. Our approach to seamlessly integrating frameworks such as Sentence Transformers, spaCy, Flair and USE with Key-BERT.

III. PROPOSED METHEDOLOGY

In this work, three different embedding models have been experimented upon, as underlying architectures to Key-BERT, namely, Sentence Transformers, Flair, spaCy and Universal Sentence Encoder, to identify the best performing model for our particular use case of key phrase extraction from short sentences. We chose these models due to their high accuracy on extracting embeddings for text classification tasks. Then the Key-BERT library leverages these embeddings to extract key phrases from the input text. To offer a fair comparison, we ensure that all the models are tested with the same hyperparameters while fine tuning and diversifying the results. Finally, we show a comparison of how each model works based on 3 criteria which are BERT score, Speed or inference time and adaptability to transfer learning. As we change the model, which is extracting the embeddings, the final performance of the key phrase extraction task also varies. The description of the working of the different embedding models that have be chosen is described. The end-to-end pipeline of our approach is shown in Figure 1.

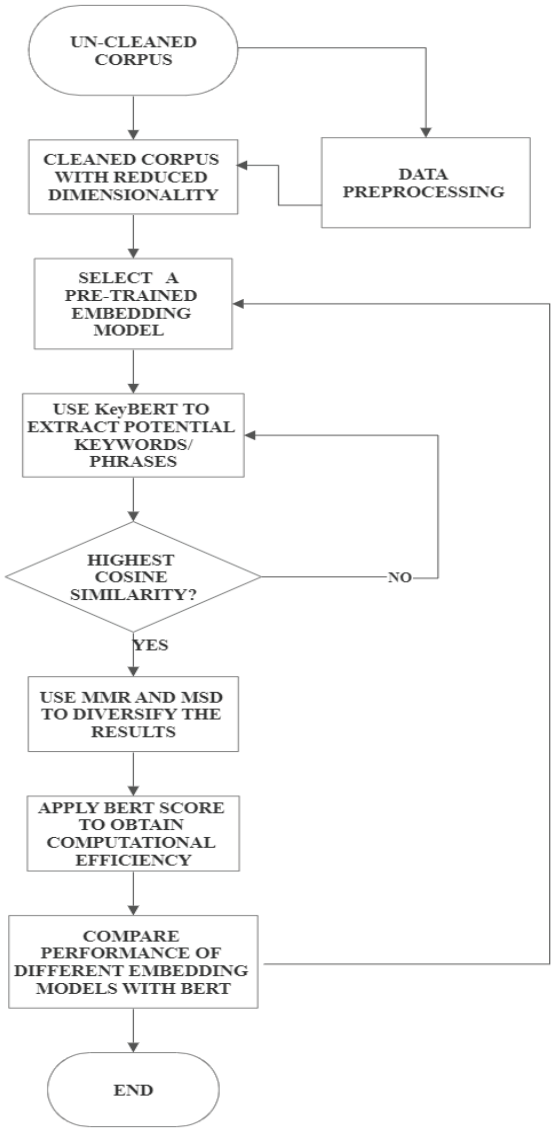

Fig.1. End to End Pipeline of the approach

A. Sentence Transformers

Sentence Transformers are based on transformer architectures, which are deep learning models designed for natural language processing tasks. The architecture of a Sentence Transformer model typically consists of several layers of self-attention and feed-forward neural networks.

The input text is tokenized into sub word units, such as WordPiece or Byte Pair Encoding (BPE).These sub word tokens are then converted into token embeddings, which are initially random vectors. The model's self-attention mechanism allows it to attend to different parts of the input text while considering the dependencies between words or sub word. This attention mechanism enables the model to capture contextual relationships and dependencies within the text. The self-attention layers generate attention scores that determine the importance of each token with respect to other tokens in the input sequence. The output of the self-attention layers is combined with the initial token embeddings using a residual connection and layer normalization. This combination preserves the original token information while incorporating the contextual information captured by the self-attention layers. The resulting embeddings contain rich semantic and contextual information about the input text.

B. Universal Sentence Encoder

The USE model is trained to generate high-quality embeddings that capture the semantic meaning of sentences. The architecture of the USE model combines both encoder and transformer components to extract embeddings from input sentences. The encoder component of the USE model utilizes a deep neural network, such as a multi-layer bidirectional LSTM or a deep averaging network (DAN). This encoder processes the input sentence word by word, capturing the syntactic and semantic information at each word position. The encoder's recurrent or averaging mechanisms allow it to consider the surrounding words and their relationships. In addition to the encoder, the USE model incorporates a transformer component. Transformers employ self-attention mechanisms. The self-attention mechanism attends to different words and assigns attention weights based on their relevance to other words in the sentence, allowing the model to effectively capture the overall context and meaning. The output of the encoder and transformer layers is combined using pooling operations, such as max pooling or mean pooling, to obtain a fixed-length sentence embedding.

C. Flair

Flair is a powerful natural language processing library. Flair creates embeddings by utilizing a combination of pre-trained word embeddings and contextual string embeddings. Flair utilizes a contextual string embedding approach to create embeddings for text. The process begins with tokenizing the input text into individual words or sub words using a word-level or character-level tokenizer. Each token is then mapped to a trainable embedding layer, which assigns a unique vector representation to each token. These embeddings serve as initial representations for the tokens. Next, the token embeddings are passed through a bi-directional recurrent neural network (RNN), such as a Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU). The bi-directional RNN processes the tokens in both forward and backward directions, capturing the contextual information from the surrounding words. The hidden states of the bi-directional RNN at each time step are concatenated to create a contextualized representation for each token. This contextualized representation incorporates the information from the entire input sequence, allowing the model to capture the influence of the surrounding words on the meaning of each token. The embeddings are trainable.

D. spaCy

SpaCy is a popular library for natural language processing that offers a variety of pretrained models, including those that provide word and sentence embeddings. At its core, spaCy utilizes convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to process text and extract embeddings. The text is tokenized into individual words, and each word is represented by a unique word vector. These word vectors are typically initialized randomly or using pre-trained word embeddings like Word2Vec or GloVe. CNN units employ filters to scan over the word vectors and capture local contextual information. RNN units, such as long short-term memory (LSTM) or gated recurrent units (GRU), process the word vectors sequentially, capturing sequential dependencies and long-range contextual information. The output of the hidden layers is transformed into a fixed-size representation, often using pooling operations like max pooling or mean pooling. These pooling operations aggregate the information from the hidden layers to obtain a fixed-length vector, which serves as the embedding for the sentence or text.

E. Key-BERT

KeyBERT employs an embedding-based approach to generate key phrase candidates, which can encompass individual words, phrases, or multi-word expressions. Each key phrase candidate is associated with its own embedding. By adjusting the "n_gram_range" parameter, the size of the resulting candidates can be modified, allowing for the extraction of phrases with a specific number of keywords. To assess the similarity and relevance of each key phrase candidate to the input sentences, cosine similarity scores are computed. This involves comparing the embedding of each candidate with the embeddings of the input sentences. The cosine similarity measure indicates the degree of similarity between the vectors. Given two vectors of attributes 'a' and 'b', the cosine similarity θ is represented using a dot product and magnitude as in (1)

$$\cos\theta = \frac{\vec{a} * \vec{b}}{\|a\|\|b\|} = \frac{\sum_i^n a_i b_i}{\sqrt{\sum_i^n a_i^2}\sqrt{\sum_i^n b_i^2}} \quad (1)$$

$$\sum_i^n a_i\, b_i = a_1 b_1 + a_2 b_2 + \cdots\ a_n b_n\, i \quad (2)$$

The resulting similarity ranges from 0 indicating 'different', to 1 indicating 'same' and in between these values indicating intermediate similarity and dissimilarity. For the text matching, the attribute vectors a and b are the term frequency vectors of the documents. The ranking of key phrase candidates is determined based on their cosine similarity scores, with higher scores indicating greater similarity and relevance to the input sentences. Diversification of the results requires striking a delicate balance between the accuracy of keywords/key phrases and the diversity among them. Key-BERT incorporates two algorithms, Maximal Marginal Relevance (MMR) and Max Sum Similarity (MSS), to explicitly expand the coverage and diversity of the selected key phrases.

In our evaluation, we consider three criteria to compare the embedding models. These include the BERT score for performance evaluation, model speed and inference time, and the ability to adapt to transfer learning. These criteria provide a comprehensive assessment of the strengths and weaknesses of each embedding model.

IV. EXPERIMENTATION AND RESULT

This section exposes the empirical experiments that have been carried out to assess four pre-trained embedding models. Initially, we present an overview of the dataset and the preprocessing stage. The hyperparameter tuning and diversification strategies employed on the Key-BERT model ensuring uniformity across all tested embedding models has been highlighted. Additionally, we elaborate on the comparison criteria utilised to assess model performance, including relevant metrics and specific evaluation measures for keyword extraction. These four models have been selected for two main reasons. Firstly, to the best of our knowledge, these models are well-established among NLP practitioners and researchers. For example, a fine-tuned sentence transformer is usually the go-to baseline in many diverse NLP applications. The same applies to other models in different contexts. Secondly, the four models are structurally diverse, allowing us to cover relevant scenarios and use-cases. For instance, Sentence Transformer is based on transformer models, which consist of stacked self-attention and feed-forward layers. On the other hand, the Universal Sentence Encoder utilises a combination of recurrent and convolutional neural networks (RNNs and CNNs) to process words and sentences. Flair adopts a character-level and word-level hybrid architecture while spaCY uses a pipeline-based architecture that consists of several components, such as tokenization, part-of-speech tagging, and dependency parsing.

A. Dataset and Preprocessing

The dataset we have exploited in this paper is an unlabelled corpus of short sentences. These short sentences serve as instructions to a computer system to automatically execute file management related instructions. The structure of our dataset is as follows: There are 5000 documents consisting of short sentences. To reduce model complexity and collinearity we reduced the dimensionality by removing highly correlated attributes. NLTK library supports tasks like classification, tagging, stemming, parsing, and semantic reasoning. We have converted our text to lowercase and remove non-alphanumeric special characters and punctuations which carry no weight in the textual data for our use case. Maintaining grammatical correctness is not relevant to our data and hence stop words have been removed. It is important to reduce infected words to their root form as the base meaning is the one that holds contextual importance for the data. For this lemmatization with POS tagging has been used. Stemming is not performed at preprocessing stage, firstly because words with the same stem may represent different meanings and, secondly, the word embeddings do not contain word stems. After these preprocessing steps, we feed the cleaned data to the different pretrained models to extract relevant key phrases and compare their performances.

With each embedding model we also explicitly expand coverage and diversity among the selected key phrases by introducing an embedding-based maximal marginal relevance (MMR) for new phrases. The maximum sum distance between pairs of data is defined as the pairs of data for which the distance between them is maximised. In our case, we have maximised the candidate's similarity to the document whilst minimising the similarity between candidates. To do this, the top 20 keywords key phrases have been selected, and out of the 20, we select the 5 which are least like each other."nr_candidates" refers to the parameter that determines the number of candidate phrases considered during the extraction of key phrases. For a low value of "nr_candidates", our results seem to be around 6% more accurate than when with a higher value of "nr_candidates". There is a trade-off between accuracy and diversity that has been observed. It is advised to keep "nr_candidates" less than around 20% of the total number of unique words in the document. There are multiple versions of embedding models available under each of the 4 varieties considered. The versions selected for our experimentation are as follows:

- In Sentence Transformers, the' all-mpnet-base-v2' model provides the best quality, but since '*all-MiniLM-L6-v2*' is 5 times faster and still offers good quality and therefore we are using the same.
- The "flair-embeddings" model in Flair can be a suitable choice.
- Since our corpus contains slightly domain specific text, we use the "en_core_web_lg" in spaCy .
- "Universal-sentence-encoder-large"under Universal Sentence Encoder has been used for our use case.

B. Comparison of Performance

The comparison has been done based on 3 factors namely, BERT scores, Speed or inference Time and Adaptability to transfer learning. The results of the same have been demonstrated below.

a) BERT Score: BERT Score is an evaluation metric for Natural Language Processing (NLP) tasks that aims to measure the quality of generated or predicted sentences by comparing them to reference sentences. This model has generated keywords which are compared with reference keywords provided as a separate text file with our corpus for validation purpose. BERT Score calculates precision, recall, and F1 score based on the obtained similarities between the predicted and reference keywords. The precision here measures the proportion of retrieved instances that are like the input phrases, while recall measures the proportion of relevant instances that are successfully retrieved. The F1 score combines these two metrics into a single value, providing a balanced assessment of the system's effectiveness. A higher F1 score indicates a better balance between precision and recall, implying that the keyword search system is accurately retrieving relevant instances while minimising false positives and false negatives. It addresses some limitations of n-gram-based measures by considering contextual information, which aligns better with human judgement.We applied the pre-trained transformer models as described in Section III. The results computed are averaged over 5 different instances of inferencing and are presented in Table I. A graphical representation of the same has been provided in Figure 2.

TABLE I. BERT SCORE VALUES

Embedding Model Name	Net F1 Scores	Net Precision	Net Recall
Sentence Transformer	0.850	0.837	0.864
Universal Sentence Encoder	0.743	0.735	0.735
Flair	0.645	0.690	0.655
spaCy	0.628	0.673	0.639

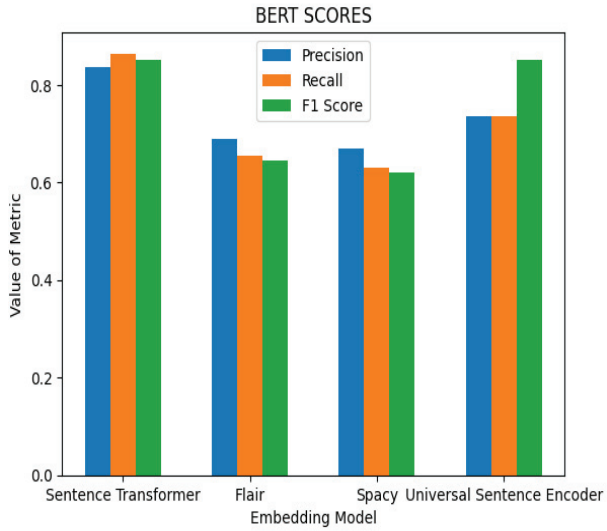

Fig.2. BERT scores graph for different embedding models

b) Speed or Inference Time: In real-time applications like search engines and content recommendation systems, the utilisation of models with superior execution times enables swift extraction of key phrases.Transformer-based models, with their attention mechanisms and larger parameter sizes, may have relatively longer inference times. Flair models are designed to provide fast inference times. They utilise lightweight architectures and efficient algorithms. USE models are known for their fast inference times. They have been optimised to deliver efficient computations, allowing for speedy predictions. spaCy offers fast inference times as well. An important observation that has been made here is that a technique called quantization achieves improved model latency by utilising hardware optimizations and taking advantage of specialised instructions for low-bit computations on modern processors. The Python 'Time' module has been used in our work and the inference times have also been averaged over 5 instances. It has been observed from the experiment that Flair performs the fastest with an execution time of 25.6 seconds and the latency has been accelerated from 25.6 seconds to 16.41seconds or (1.56x).Table II. summarises our findings. A graphical representation has also been shown in Figure 3.

TABLE II. INFERENCE TIMES

Embedding Model Name	**Execution time before Quantization (Seconds)**	**Execution time after Quantization (Seconds)**
Sentence Transformer	41.1	26.34
Universal Sentence Encoder	35.8	22.94
Flair	25.60	16.41
spaCy	28.00	17.94

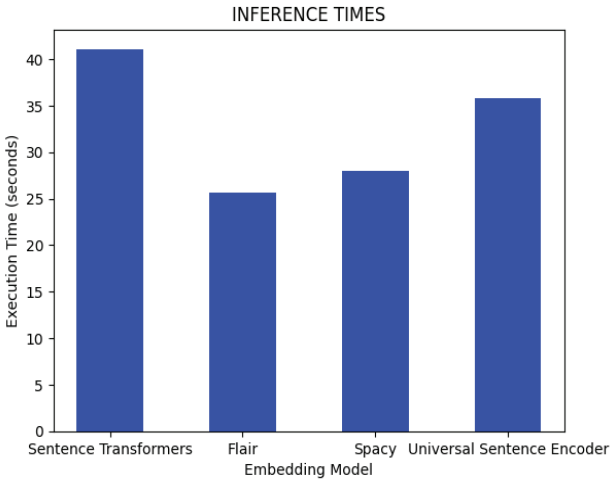

Fig.3. Inference time scores graph for different embedding models

Adaptability to Transfer Learning: The adaptability of a model to transfer learning is a critical aspect due to its ability to leverage learned representations from a source domain or task and apply them to a target domain or task with limited labelled data.Transfer learning facilitates domain adaptation by fine-tuning pre-trained models to capture domain-specific nuances and optimise performance in the target domain.It fosters generalisation by enabling models to acquire robust and transferable representations, leading to improved performance on unseen data. The adaptability to transfer learning thus plays a vital role in enhancing the efficiency and efficacy of NLP models. Sentence Transformers are specifically designed for transfer learning tasks, such as sentence-level embeddings, semantic textual similarity, or text classification. This model excels in transfer learning scenarios by providing adaptable embeddings.Flair provides contextual string embeddings and is well adaptable to transfer learning scenarios. It allows fine-tuning of embeddings on specific downstream tasks, enabling domain adaptation and improved performance. USE is not specifically designed for transfer learning; its embeddings can be fine-tuned for specific tasks. However, the fine-tuning capabilities are more limited compared to Sentence Transformers or Flair. While spaCy does not have built-in support for transfer learning with embeddings, it provides a framework for training models on specific tasks. Overall, we deduce that Sentence Transformers and Flair demonstrate stronger adaptability to transfer learning. Universal Sentence Encoder can be adapted to transfer learning to some extent, while spaCy's strength lies more in its efficient processing pipeline rather than extensive transfer learning capabilities.

V. CONCLUSION AND FUTURE WORK

By classifying a broad range of state-of-the-art approaches and analyzing their benefits and drawbacks we provide a clearer picture of them. However, it's important to note that the choice of evaluation criteria may vary depending on the specific application or research context. Our work dwells deeper by performing a set of controlled experiments on a real-world dataset to find the BERT score and inference times. The obtained results align with the previous literature in terms of absolute performance. We found that Sentence transformer version "all-MiniLM-L6-v2" outperforms other embedding models listed above for our corpus, with respect to BERT score evaluation and Adaptability to transfer learning However spaCY performs

the fastest whilst retaining the accuracy on an average of multiple runs. Our research can be extended by considering alternative versions and variations of the Sentence Transformers model, Universal Sentence Encoder (USE), Flair, and spaCy models beyond the ones we have initially explored. This expansion will allow for a more comprehensive analysis of the models' architectural variances, pre-training strategies, and their impact on key phrase extraction or other relevant tasks. Furthermore, this could be extended by conducting experiments using diverse and multilingual datasets to enhance the generalizability of our findings. By incorporating such datasets, we can evaluate the models' adaptability to different languages, domains, and text genres, thereby assessing their capabilities in handling a wider range of textual data. Overall, the work highlights the need for further extended research on comparative analysis of embedding models for a vast variety of data sets in the NLP community, which should define and encourage good practices in terms of reproducibility and replicability of results.

REFERENCES

[1] Weikang Rui, Jinwen Liu and Yawei Jia, "Unsupervised feature selection for text classification via word embedding", IEEE International Conference on Big Data Analysis (ICBDA) ,12-14th March Hangzhou,China,2016.

[2] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes and Adam Jatowt,"YAKE! Keyword extraction from single documents using multiple local features",Information Sciences,Volume 509,2020,Pages 257-289

[3] Tobias Schnabel et al., "Evaluation methods for unsupervised word embeddings", Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

[4] Debasis Ganguly et al., "Word embedding based generalised language model for information retrieval", *Proceedings of the 38th* International ACM SIGIR Conference on Research and Development in Information Retrieval, 2018.

[5] Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina,"Bert: Pre-training of deep bidirectional transformers for language understanding",arXiv preprint arXiv:1810.04805,2018

[6] Hasan, Kazi Saidul, and Vincent Ng. "Automatic keyphrase extraction: A survey of the state of the art." Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. 2014

[7] Adrien Bougouin, Florian Boudin, and Beatrice Daille. ´ "TopicRank : Graph-Based Topic Ranking for Keyphrase Extraction",Proc. IJCNLP 2013, (October):543–551

[8] Adrien Bougouin, Florian Boudin, and Beatrice Daille. ´ "TopicRank : Graph-Based Topic Ranking for Keyphrase Extraction",Proc. IJCNLP 2013, (October):543–551

[9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks", Proc. Conf. Empir. Methods Natural Lang. Process. Assoc. Compute. Linguist., pp. 3982-3992, 2019.

[10] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", Proc. Conf. North Amer. Chapter Assoc. Compute. Linguist., pp. 4171-4186, 2019.

[11] Li Wengen and Jiabao Zhao, "TextRank algorithm by exploiting Wikipedia for short text keywords extraction", Information Science and Control Engineering (ICISCE) 2016 3rd International Conference on. IEEE, 2016

[12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks", Proc. Conf. Empir. Methods Natural Lang. Process. Assoc. Compute. Linguist., pp. 3982-3992, 2019.

[13] Lee Sungjick and Han-joon Kim, "News keyword extraction for topic tracking", 2008 fourth international conference on networked computing and advanced information management, vol. 2, 2018.