Context-Based Keyphrase Extraction using BERT as a Sentence Embedder

Amritesh Amrit
Department of Computer Science
PES University
Bangalore, India
amriteshc101@icloud.com

Ashwath Krishnan
Department of Computer Science
PES University
Bangalore, India
ashwathkrishnan45@gmail.com

Sudhanva Rajesh
Department of Computer Science
PES University
Bangalore, India
sudhanva0001@gmail.com

Dr. Shylaja SS
Department of Computer Science
PES University
Bangalore, India
shylaja.sharath@pes.edu

***Abstract*—The extraction of a set of words that are most relevant to a given text document is called keyphrase extraction. Supervised keyphrase extraction requires a labelled dataset of documents and keyphrases to train on. They also tend to perform poorly outside the domain represented by the training data. On the other hand, unsupervised methods do not need to be trained, but have poor accuracy. We overcome this constraint by using BERT, short for 'Bidirectional Encoder Representations from Transformers', as an embedder. The keyphrases extracted through this method are more relevant to the document than other methods since BERT provides better embeddings of a word, by taking into consideration both the left and the right context of a given word. Our approach achieved an accuracy of up to 80% when compared against the manual selection of keyphrases.**

***Keywords*—Keyphrase extraction, BERT, sentence embedding, cosine similarity.**

I. Introduction

By definition, the purpose of keyphrase extraction is to select a set of phrases that best describes a given document, regardless of its domain. Keyphrases are best described by two important properties, which have been talked about extensively throughout the corresponding literature. Firstly, keyphrases should be well-formed phrases, that is, sequences of words which are grammatically valid and semantically complete. Secondly, keyphrases should be informative, they should be relevant to the main topic in a given document. A third crucial property, coverage, requires the set of keyphrases to cover the majority of the main topics in the document.

Keyphrases help in faster and more accurate searching of required documents, from a corpus of documents, and have also been used to improve various tasks such as text categorization and text summarization. The Internet is a huge space and has a vast amount of information. But as the size of the internet increases day by day, information becomes easy to get lost. Thus, comes the idea of classification of information on the internet. Classifying information on the internet is done using tags, keywords and key-phrases as they play an important role in ranking information on the internet.

With the launch of the first set of search engines, users used single words and simple algorithms for searches. With time search engines have grown and now accept key-phrases with more than one keyword. The problem arises when the wrong key-phrase is chosen. Having one extra word might affect our reach on the internet. Therefore, selection of the right keyphrase is the key to an accurate search result. We address this problem with a new unsupervised approach to keyphrase extraction. This calls for a precise model that produces relevant keyphrases that best describe the given text document.

Unsupervised keyphrase extraction works better than supervised extraction methods for a number of reasons. It eliminates the requirement of a labelled corpus of documents and their manually keyphrases for training. Supervised methods also tend to perform poorly outside the domain of their training data. On the other hand, unsupervised keyphrase extraction addresses these limitations by only using information extracted from the given input document.This method assumes that a candidate phrase is more important if it is related to a large number of other candidates. This results in supervised models being slow and highly accurate and unsupervised models being faster with lower accuracy.

Natural Language Processing achieved a breakthrough with the introduction of BERT (Bidirectional Encoder Representations from Transformers). It is pre-trained using bidirectional representations from unlabeled text by jointly considering the left and the right context. Hence, the pre-trained BERT model can be fine-tuned for a required task using an additional output layer. BERT, based on the transformer architecture, is pre-trained on a large collection of unlabelled data including the entire Wikipedia.

For truly understanding the meaning of a sentence, BERT has an advantage of being bidirectional. . The original BERT model was developed for the english-language, which has two types: (1) the $BERT_{BASE}$ model, which is a 12-layer neural network architecture, and (2) the $BERT_{LARGE}$ model, which is a 24-layer neural network architecture.

The root of BERT lies in clever ideas initiated by the NLP community. It started with a common idea of word embedding provided by Word2Vec which helps in establishing a relationship between two words. Following Word2Vec was ElMo, which provided context-based id's that were generated for each token using a Bi-directional LSTM. This was simply achievable by a concept called Language Modeling. ULM-Fit utilizes a lot of information that it gathers in the pre-training process. This finally provides for NLP a way to perform transfer learning. These advancements in the field of NLP resulted in BERT.

BERT uses the idea of predicting the next word in the sentence and uses encoders as well as decoders which helps it to mask words and hence provide a better accuracy during its training on the dataset. With its vast pre-trained dataset, BERT has quickly become popular and effective. It can be used for both supervised as well as unsupervised learning and promises to provide good accuracy in both the scenarios. It's because of these advantages that BERT is being used in the google search engine.

II. Related Work

Significant work has been done in the past where keyphrase extraction was considered as a classification problem and models were trained on huge collections of data. Since supervised methods require manually labelled training datasets, it takes a considerable amount of time. Our approach being unsupervised, we will be mainly focusing on unsupervised methods throughout this paper.

In the article titled "Intro to Automatic Keyphrase Extraction"[1], there are two main steps in the process of keyphrase extraction: extracting the candidate keyphrases and their ranking. The article suggests an approach where all the phrases in a document are considered to be candidates. However, this leads to increased computation. Hence, the suggested methodology was the use of filters using expression matching(regex). The regex patterns are constructed using POS patterns. The article uses the NLTK RegexParser model for POS pattern matching. The TF*IDF method is then used to find the most relevant key phrases. This can be done using one of two different models: A (1)Supervised and a (2)Unsupervised model. Supervised models are slow and are highly accurate, whereas unsupervised models are fast but have lower accuracy. Hence, there is a choice between accuracy or speed. PageRank, which is a graph-based system, was used to rank the keyphrases.

As mentioned in the paper titled "Simple Unsupervised Keyphrase Extraction using Sentence Embeddings"[2], EmbedRank is an unsupervised method to extract relevant keyphrases from the given document using sentence embeddings. It consists of three main steps as follows: (1) extracting candidate phrases from the text, based on part-of-speech sequences. (2) Using sentence embeddings to represent both the document and the cand idate phrases in the same embedding space. (3) Ranking the candidate phrases to select the output keyphrases. The candidate phrases are ranked based on their respective cosine distance to the document embedding.

By the use of sentence embeddings, EmbedRank embeds both the document and candidate phrases into the same embedding space. Hence, EmbedRank achieves higher F-scores than graph-based systems on standard datasets. However, the above-stated methods only take into consideration the context in a single direction (left/right). We overcome this constraint by using BERT, which takes into consideration the context in both directions (left and right) of a given word/token.

III. Methodology

The methodology consists of three main parts: (1) Extracting candidate keyphrases based on RegEx patterns, (2) Embedding the generated candidate keyphrases and the document using BERT, (3) Ranking the keyphrases based on their relevance to the text document using cosine similarity. The following subsections explain the same in detail.

A. Extracting candidate keyphrases based on RegEx patterns:

Part-of-speech(POS) tagging is the process of matching a given word in a text to a specific part of speech, based on its definition. For POS tagging, we have used Spacy which is an NLP language library. We utilise the English library provided by spacy. For the extraction of candidate keyphrases, different POS tag patterns were constructed. One of the major POS tag patterns include an adjective (zero or more times) followed by a noun (one or more times), which is then followed by a preposition, adjective or a noun. A simpler pattern could be an adjective followed by a noun. It was also observed that common names such as the names of countries and prominent organisations could also be potential keyphrases and hence, they were also included in the POS tag patterns. The below code snippet shows the same:

```
pattern1 = [{'POS':'ADJ', 'OP':'*'}, {'IS_PUNCT': True,
'OP':'*'}, {'POS':'NOUN', 'OP':'+'},{'IS_PUNCT': True,
'OP':'*'},{'POS':'ADP'},{'OP':'?'},{'POS':'ADJ'},{'IS_P
UNCT': True, 'OP':'*'},{'POS':'NOUN'}]
pattern2 = [{'POS':'ADJ'},{'IS_PUNCT': True,
'OP':'*'},{'POS':'NOUN'}]
pattern3 = [{'ENT_TYPE':'ORG', 'OP':'+'}]
pattern4 = [{'ENT_TYPE':'GPE', 'OP':'+'}]
matcher.add('Noun', None, pattern1, pattern2, pattern3,
pattern4)
```

Once the candidate keyphrases have been extracted, they now need to be cleaned. The keyphrases extracted contain redundant phrases which need to be removed. An example of redundant keyphrases would be the two phrases: “International Cricket” and “International Cricket Council”. It is quite evident that the “International Cricket Council” is the required candidate keyphrase. Hence, the candidate phrases were preprocessed to filter out redundant words/phrases. The filter functions on a general idea of checking if a particular phrase is contained within another keyphrase. If this is so, the phrases are discarded and if not, they are added to the list of candidate keyphrases, which is passed onto the BERT embedder.

B. Embedding candidate phrases and the given text document using BERT:

Sentence embeddings can be defined as the representation of a given sentence as an n-dimensional vector. The sentence embeddings in this project have been performed using BERT, which stands for “Bidirectional Encoder Representations from Transformers”. The main idea of embedding is to embed the whole document as a single tensor and embed each of our candidate phrases as individual tensors. We then proceed to compare each of these phrase tensors with our document tensor. We utilise the base-uncased-model of BERT that ignores casing and is of a smaller size and efficient for our purpose.

The BERT model being pretrained, expects its input to be in a specific format, where special tokens have to be appended to mark the beginning and end of each sentence, where the sentences themselves must have each word mapped to their token ids. Furthermore, all the sentences passed into the model have to be of the same length, which brings up the need for padding. The ‘encode_plus’ function of the BertTokenizer module takes care of all the above steps. We apply this function to all the sentences in our document, and all of our candidate phrases and convert them to the required format. We also store a list of attention masks for all our sentences and phrases, which indicates which of our tokens are used for padding.

We retrieve the embeddings of each word in our sentence, and take its mean as the embedding for the sentence. We then proceed to take the mean of all our sentence embeddings to get the document embedding. The below code snippet explains the same:

```
hidden_states=[]
with torch.no_grad():
    for i in range(len(input_ids)):
      outputs = model(input_ids[i],attention_mask=attention_masks[i])
      hidden_states.append(outputs[2])
sentence_embeddings=[]
for i in range(len(hidden_states)):
  token_vecs = hidden_states[i][-2][0]
  sentence_embedding = torch.mean(token_vecs, dim=0)
  sentence_embeddings.append(sentence_embedding)
sentence_embeddings = torch.stack(sentence_embeddings,dim=0)
document_embedding=torch.mean(sentence_embeddings, dim=0)
```

Similarly, we take the mean embedding of all the words in a particular candidate phrase, as the embedding for that phrase.

```
hidden_states=[]
with torch.no_grad():
    for i in range(len(input_ids)):
      outputs =
model(input_ids[i],attention_mask=attention_masks[i])
      hidden_states.append(outputs[2])
phrase_embeddings=[]
for i in range(len(hidden_states)):
  token_vecs = hidden_states[i][-2][0]
  phrase_embedding = torch.mean(token_vecs, dim=0)
  phrase_embeddings.append(phrase_embedding)
```

C. Ranking the keyphrases based on cosine similarity:

Now that we have the embedding for the whole document and the embeddings of all our keyphrases, we calculate the cosine similarity between the document tensor and each phrase tensor. Cosine similarity is the cosine value of the angle between two vectors in an *n*-dimensional space. It is the dot product of the two vectors divided by the product of their magnitude. This value ranges between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar. The more relevant key phrases have a higher similarity value, and hence we sort out our candidate phrases based on the similarity in descending order.

```
from scipy.spatial.distance import cosine
for i in range(len(phrase_embeddings)):
  sim = 1 - cosine(document_embedding, phrase_embeddings[i])
  similarity.append(sim)
  keyword.append(phrases[i])
for i in range(len(similarity)):
  for j in range(0, len(similarity)-i-1):
    if similarity[j] < similarity[j+1] :
      similarity[j], similarity[j+1] = similarity[j+1], similarity[j]
      keyword[j], keyword[j+1] = keyword[j+1], keyword[j]
selected_keyphrases=keyword[0:10]
```

A snippet of the input file, containing a short description of cricket is presented in Listing 1 [3].

Cricket is a bat-and-ball game played between two teams of eleven players on a field at the centre of which is a 22-yard (20-metre) pitch with a wicket at each end, each comprising two bails balanced on three stumps. The batting side scores runs by striking the ball bowled at the wicket with the bat (and running between the wickets), while the bowling and fielding side tries to prevent this and dismiss each batter. When ten batters have been dismissed, the innings ends and the teams swap roles. The game is adjudicated by two umpires, aided by a third umpire and match referee in international matches. Forms of cricket range from Twenty20, with each team batting for a single innings of 20 overs, to Test matches played over five days.

Listing 1[3]: A snippet of the input text file.

The selected keyphrases and their corresponding cosine similarities have been shown in TABLE 1. As it can be seen, keyphrases with the highest cosine similarity are ranked first.

IV. Results

The model was tested on a variety of documents and the result was a list of keyphrases that were relevant to each given document. To check this relevancy, we manually (through means of a survey) selected keyphrases from the document and compared the same with the keyphrases generated by our model. We take the keyphrases extracted manually as the true keyphrases and calculate the accuracy of our model accordingly.

The accuracy is calculated as the ratio of number of keyphrases common to both the model output and manual selection to the number of manually selected keyphrases. We calculate this accuracy for keyphrases generated from four documents:

- [14] (Wikipedia page on Cricket)
- [7] (Wikipedia page on the Olympic games)
- [8] (Wikipedia page on Android)
- [8] (Wikipedia page on World war I)

The resultant accuracy is depicted by the graph as shown in Fig I.

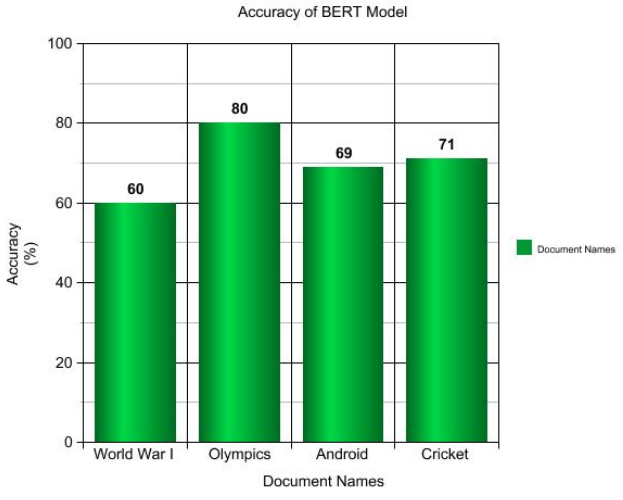

Fig I. Accuracy of Unsupervised Keyphrase extraction using BERT Model

The difference in accuracy between different documents is due to the length of each document. As the size of the input text document increases, the number of keyphrases generated also increases. This lowers the accuracy and hence, a large document cannot be represented by a small number of keyphrases.

TABLE I. KEYPHRASES GENERATED FOR ARTICLE ON CRICKET

Key-Phrases Generated	Similarity(%)
international cricket	76.1108
international standard	75.0624
the British Empire	75.0468
the International Cricket Council ICC	74.9033
Indian Subcontinent	74.1158
Australia	72.9060
other country	72.4605
first international matches in the second half	72.3130
Twenty20	71.8584
Australasia	71.4265

V. Conclusions

This paper proposed an unsupervised approach to extract keyphrases by taking into consideration the context of a given word in both directions in a sentence. We first extract candidate keyphrases using specific RegEx patterns. We then presented BERT as an embedder, where we embed the candidate phrases and the given text document as a whole. Finally, we used cosine similarity to find the similarity/relevance between the candidate phrases and the document and rank the phrases accordingly in order of their relevance to the document. This approach achieved an accuracy of up to 80% against the manual selection of keyphrases.

VI. Future Directions

In this paper, we have used cosine similarity to rank the keyphrases. However, the ranking could be improved if an industry standard ranking system is used. Additional POS patterns could be included to increase the diversity in the extracted keyphrases.

References

[1] Burton DeWilde, "Intro to Automatic Keyphrase Extraction", September 2014.

[2] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, Martin Jaggi. "Simple Unsupervised Keyphrase Extraction using Sentence Embeddings". In Proceedings of the 22nd Conference on Computational Natural Language Learning, October 2018.

[3] Adrien Bougouin, Florian Boudin, and Beatrice Daille. "TopicRank : Graph-Based Topic Ranking for Keyphrase Extraction". Proc. IJCNLP October, 2013.

[4] Corina Florescu and Cornelia Caragea. “A position-biased pagerank algorithm for keyphrase extraction”. 2017.

[5] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. “Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features”. 2017.

[6] Chris McDonald Rui Wang, Wei Liu. “Corpus Independent Generic Keyphrase Extraction Using Word Embedding Vectors”, 2015.

[7] Wikipedia page on Olympic Games

[8] Wikipedia page on Android.

[9] Wikipedia page on World War 1.

[10] Jay Alammar. ”The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)”, December 2018.

[11] Rani Horev. BERT Explained: State of the art language model for NLP, November 2018.

[12] Shay Palachy. Document Embedding Techniques, September 2019.[Online]

[13] Xu Liang. Understand TextRank for Keyword Extraction by Python, February 2019.

[14] Wikipedia page on Cricket

[15[Han xiao. BERT as a service