AttentionRank: Unsupervised keyphrase Extraction using Self and Cross Attentions

Haoran Ding and **Xiao Luo**
Purdue School of Enigneering and Technology, IUPUI, USA
hd10@iu.edu, luo25@iupui.edu

Abstract

Keyword or keyphrase extraction is to identify words or phrases presenting the main topics of a document. This paper proposes the AttentionRank, a hybrid attention model, to identify keyphrases from a document in an unsupervised manner. AttentionRank calculates self-attention and cross-attention using a pre-trained language model. The self-attention is designed to determine the importance of a candidate within the context of a sentence. The cross-attention is calculated to identify the semantic relevance between a candidate and sentences within a document. We evaluate the AttentionRank on three publicly available datasets against seven baselines. The results show that the AttentionRank is an effective and robust unsupervised keyphrase extraction model on both long and short documents. Source code is available on Github [1].

1 Introduction

A vast amount of scientific or non-scientific articles are published online every year. Although some of them have associated keyphrases specified, which made them easy to index and search, a considerable amount of them has no keyphrase defined, making indexing and information retrieval challenging. Given so many articles available, it is not feasible to manually extract keyphrases from each. Automate the keyphrases extraction becomes crucial. Keyphrase extraction has immense value to the downstream text mining tasks, such as text segmentation, text summarization, query expansion, indexing, and so on.

Keyphrase extraction methods can be supervised or unsupervised. Traditional supervision methods use decision tree (Turney, 2000) or naive Bayes (Witten et al., 2005) to identify whether the input word/phrase is a keyphrase. With the advancing of neural networks, various supervised deep learning models (Meng et al., 2017; Alzaidy et al., 2019; Sun et al., 2019) are developed to extract keyphrases. The supervised method requires a large labeled training dataset and is often domain-specific, whereas unsupervised methods do not need labeled datasets. The traditional unsupervised methods use statistical and graph-based approaches. The statistical-based methods (Beliga et al., 2016; Rose et al., 2010; Campos et al., 2018) utilize the candidate position, frequency, length, and capitalization to determine the importance of a word. The graph-based approaches (Wan and Xiao, 2008; Gollapalli and Caragea, 2014) construct a graph with the candidates as nodes. The edges indicate the similarity or co-occurrences of the candidates. Graph-based algorithms can be applied to calculate the importance of the nodes (candidates) on the graph. Driven by recent deep language models for natural language processing and text analysis, text embedding-based or mixed statistical and embedding-based unsupervised keyphrase extraction methods have emerged, such as EmbedRank (Bennani-Smires et al., 2018), SIFRank (Sun et al., 2020), and KeyGames (Saxena et al., 2020).

In this research, we propose an attention-based unsupervised model – AttentionRank for keyphrase extraction. AttentionRank is motivated by the self-attention mechanism of the BERT model (Devlin et al., 2019), and hierarchical attention retrieval (HAR) mechanism (Zhu et al., 2019). AttentionRank model calculates the accumulated self-attention and cross-attention of a candidate to rank the importance. The accumulated self-attention value for each candidate is extracted from a pre-trained BERT model. It is calculated as adding the received attention from other words within a sentence then summing up these self-attention values on all sentences within the document. The accumulated self-attention addresses the importance of a candidate within the sentences where it locates. The cross-attention identifies the semantic

[1] https://github.com/hd10-iupui/AttentionRank

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1919–1928
November 7–11, 2021.

relevance between a candidate and the document. It calculates the word-level bidirectional attention between the embeddings of a candidate and sentences within a document then generate an enhanced document embedding for the candidate. The final ranking of a candidate is determined by the linear integration of the accumulated self-attention value and the cross-attention relevance value. A post-processing step based on the document frequency is used to remove the generic terms of a specific corpus.

AttentionRank is the first to use the attention values extracted from a pre-trained BERT model for keyphrase extraction. No additional information or domain knowledge is needed. It is generalizable to documents of any domain. This research focused on investigating whether the pre-trained none domain-specific language model can be used for keyphrase extraction.

AttentionRank is compared against seven state-of-the-art unsupervised keyphrase extraction methods on three benchmark datasets. Two datasets contain short documents, and one contains long documents. The results show that AttentionRank performs better than or as competitive as the baselines.

The main contributions of this paper are summarized as follows:

- We proposed a novel attention-based unsupervised keyphrase extraction model - AttentionRank;
- We demonstrated pre-trained language model could be utilized to identify keyphrases via self-attentions and cross-attentions.
- We showed that the AttentionRank model outperforms the compared baselines and is robust to identify keyphrases from both short and long documents of different domains.

2 Methodology

In this section, we introduce the AttentionRank model in detail. The overview architecture of our model is shown in Figure 1. AttentionRank integrates the accumulated self-attention component with the cross-attention component to calculate the final score of a candidate. The proposed model has four main steps: (1) Generate a candidate set C from a document; (2) Calculate the accumulated self-attention value a_c for each candidate c, $c \in C$; (3) Calculate the cross-attention relevancy value (r_c) between a candidate c and the document d; (4) Calculate the final score s_c for each candidate through a linear combination of a_c and r_c.

2.1 Candidates Generation

We use the candidate extraction module implemented in EmbedRank (Bennani-Smires et al., 2018). This module first use Part-of-Speech (PoS) to identify words tagged as NN, NNS, NNP, NNPS, JJ, etc. Then, the python package NLTK [2] is used to generate the noun phrases, which are candidates. Given a sentence 'Most parameters of the program are controllable by experimenter-edited text files or simple switches in the program code, thereby minimizing the need for programming to create new experiments', the extracted candidates are 'program code', 'experimenter-edited text files', 'need', 'parameters', 'simple switches', 'program', and 'new experiments'.

2.2 Accumulated Self-Attention Calculation

We use the method introduced by Clark et al. (Clark et al., 2019) to extract self-attention weights of the words from the pre-trained BERT. We sum the attentions ($a_{w'w}$) that a word (w) received from other words (w') within the same sentence (s) to obtain the attention value (a_w) of the word within a sentence, shown as Equation 1. This attention value (a_w) represents the importance of the word within the context of a sentence.

$$a_w = \sum_{w' \in s} a_{w'w} \tag{1}$$

As shown in Figure 2, all highlighted are noun chunks. Intuitively, the darker the noun chunk is, the higher self-attention it receives. They have a higher probability of being selected as keyphrases.

To calculate the self-attentions of a candidate (c) in sentence i, we add up the attention of the words in c, shown as Equation 2.

$$a_i^c = \sum_{w \in c} a_w \tag{2}$$

The document level self-attention value of candidate c is computed as the sum of all self-attention values of c in each sentence of document d:

$$a_c = \sum_{i \in d} a_i^c \tag{3}$$

[2] https://github.com/nltk

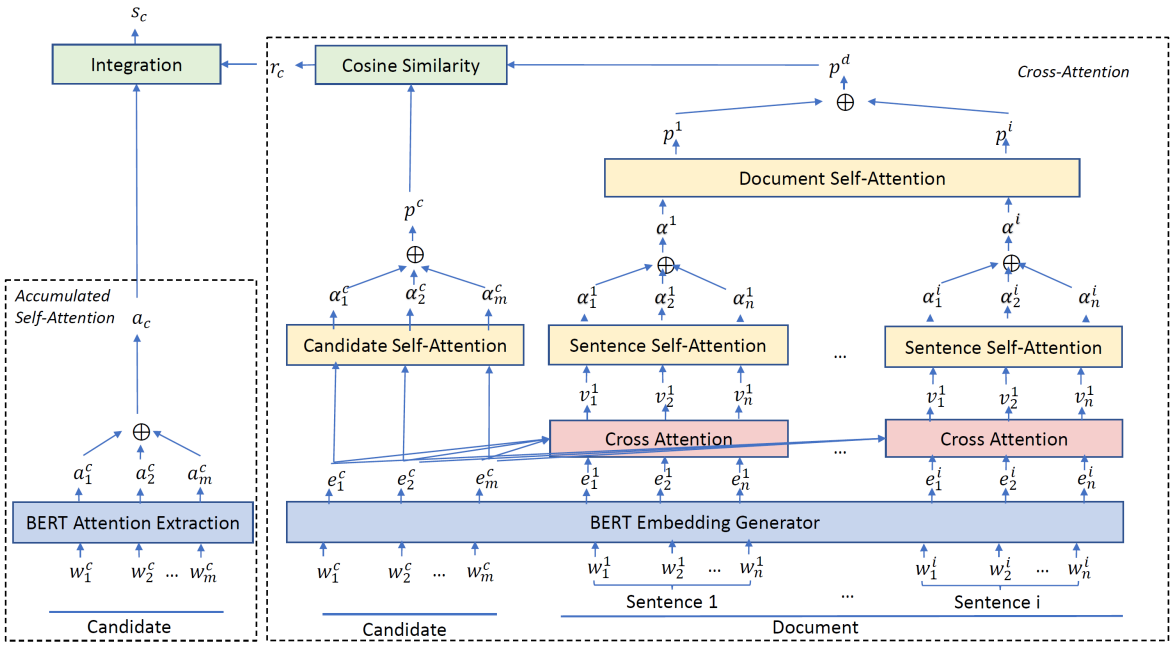


Figure 1: Architecture of the AttentionRank model

Waiting for the wave to crest [wavelength services]
Wavelength services have been hyped ad nauseam for years. But despite their quick turn-up time and impressive margins, such services have yet to live up to the industry's expectations. The reasons for this lukewarm reception are many, not the least of which is the confusion that still surrounds the technology, but most industry observers are still convinced that wavelength services with ultimately flourish.

Figure 2: Visualization of the Self-Attention Weights

2.3 Cross-Attention Calculation

The cross-attention model is inspired by the hierarchical attention retrieval (HAR) model (Zhu et al., 2019) and the bi-direction attentional model (Seo et al., 2016). Based on their network architectures, we develop the cross-attention component to measure the correlation between a candidate and the document based on the context.

A pre-trained BERT model can generate candidate c representation as $E^c = \{e_1^c, ..., e_m^c\}$, where $e_i \in \mathbb{R}^H$ is embedding of w_i, and there are m words in the candidate. Similarly, a pre-trained BERT model can also generate a representation $E^i = \{e_1^i, ..., e_n^i\}$ for a sentence i which contains n words.

Cross-attention calculates a new document embedding to better measure the contextual correlations between candidates and sentences within a document. Given a sentence i represented as $E^i \in \mathbb{R}^{n \times H}$, and a candidate c represented as $E^c \in \mathbb{R}^{m \times H}$, a similarity matrix between i and c $S \in \mathbb{R}^{n \times m}$ can be calculated as Equation 4. Then, the word based sentence to candidate and candidate to sentence similarity can be measured as Equation 5 and 6).

$$S = E^i \cdot E^{c\intercal} \tag{4}$$

$$\bar{S}_{i2c} = softmax_{row}(S) \tag{5}$$

$$\bar{S}_{c2i} = softmax_{column}(S) \tag{6}$$

The word-based cross-attention weights from sentence to candidate and from candidate to sentence are calculated as Equation 7 and 8. The new sentence representation V^i is built upon these cross-attention weights and computed by averaging the sum of the four items, shown as Equation 9. The E^i is the original context of the sentence, the A_{i2c}, $E^i \odot A_{i2c}$ and $E^i \odot A_{c2i}$ measure the context correlation between a sentence and a candidate. $\odot$ is element-wise multiplication.

$$A_{i2c} = \bar{S}_{i2c} \cdot E^c \tag{7}$$

$$A_{c2i} = \bar{S}_{i2c} \cdot \bar{S}_{c2i}^{\intercal} \cdot E^i \tag{8}$$

$$V^i = AVG(E^i, A_{i2c}, E^i \odot A_{i2c}, E^i \odot A_{c2i}) \tag{9}$$

The new sentence representation V^i is still a set of embeddings that comprises the word-based relations between the candidate and sentence. To generate a standardized sentence representation based

on V^i, a self-attention is performed on V^i to highlight the importance of the words after applying the cross attention. Given a new sentence representation $V^i = \{v_1^i, ..., v_n^i\}$ with n words, the self-attention of sentence i is calculated as Equation 10. Then, the column-wise average is calculated to gain the final representation of a sentence $\alpha^i \in \mathbb{R}^H$.

$$I = softmax_{row}(V^i \cdot V^{i\intercal}) \cdot V^i \quad (10)$$

$$\alpha^i = AVE(I[:, i]) \quad (11)$$

Once the sentence embeddings are generated, we perform a similar process on sentence embeddings to generate document embedding. Given a document d which include a set of sentences $E^d = \{\alpha^1, ..., \alpha^i\}$, to calculate the document embedding, we first generate self-attention of the document to emphasize sentences with higher correlation to the candidate (Equation 12), then take the column-wise average to get the final document embedding $p^d \in \mathbb{R}^H$.

$$P = softmax_{row}(E^d \cdot E^{d\intercal}) \cdot E^d \quad (12)$$

$$p^d = AVE(P[:, i]) \quad (13)$$

Since the candidate is also originally represented as a word embedding set $E^c = \{e_1^c, ..., e_m^c\}$, the self-attention calculation (Equation 14) is also applied, and the column-wise average is done afterwards to get the final candidate embedding $p^c \in R^H$ as Equation 15.

$$C = softmax_{row}(E^c \cdot E^{c\intercal}) \cdot E^c \quad (14)$$

$$p^c = AVE(C[:, i]) \quad (15)$$

Finally, the relevance between a candidate c and a document d is determined by the cosine similarity of p^c and p^d shown as Equation 16.

$$r_c = \frac{p^c \cdot p^d}{||p^c|| \cdot ||p^d||} \quad (16)$$

2.4 Final Score Calculation and Post-processing

For document d, the accumulated attention value a_c and cross-attention based relevance value r_c are calculated and normalized separately for each candidates. The final score of a candidate is the generated by linear integration of these two values using Equation 17, where $d \in [0, 1]$.

$$s_c = d * a_c + (1 - d) * r_c \quad (17)$$

A corpus is often domain-specific. It means some words with high document frequency might be generic words to this corpus. In this research, in order to limit the generic word or phrase becoming a keyphrase, we remove the candidates with a high document frequency than a threshold df_θ.

3 Experiments

3.1 Datasets and Evaluation Metrics

To fully evaluate the performance of AttentionRank, we evaluated it on three benchmark datasets. The percentages of the n-grams, the average number of words (AveWords), and the average number of sentences (AveSentences) per document are provided in Table 1. Over 50% of the keyphrases are either unigram or bigram. Datasets SemEval2017 (Augenstein et al., 2017) and Inspec (Hulth, 2003) contain short documents with an average of 6 to 7 sentences, whereas SemEval2010 (Kim et al., 2010) contains long documents with hundreds of sentences.

The performance of the keyphrase extraction is evaluated using Precision (P), Recall (R), and F1 measure (F1) on the top 5, 10, and 15 ranked keyphrases. The PR curve based on the top-ranked keyphrases is also generated for comparison.

Table 1: A Summary of Datasets

dataset	SemEval2017	Inspec	SemEval2010
#Doc	493	500	243
#unigram	26.34%	16.20%	22.94%
#bigram	28.60%	54.15%	44.15%
#trigram	17.88%	22.72%	23.59%
AveWords	168	134	8154
AveSentences	7	6	369

3.2 Baselines

We compared our model against seven different unsupervised models: SingleRank (Wan and Xiao, 2008), RAKE (Rose et al., 2010), TopicRank (Bougouin et al., 2013), PositionRank (Florescu and Caragea, 2017b), YAKE! (Campos et al., 2018), EmbedRank (Bennani-Smires et al., 2018),

and SIFRank (Sun et al., 2020). These baselines all generate candidates using noun phrases without any additional steps. Since KeyGames (Saxena et al., 2020) includes designed steps, such as stop word removal and threshold on token length, for candidate generation, it is not included. We used PKE[3] to run SingleRank, RAKE, and TopicRank. The published GitHub code of YAKE![4], EmbedRank[5], PositionRank[6], and SIFRank[7] were used to produce the results on the selected datasets. It is worth noting that the produced results of the baselines are slightly higher or lower than the results presented in the original papers.

3.3 HyperParameter Setting and Computational Cost

The BERT-Base (Devlin et al., 2019) is used. For different corpora, the document frequency threshold df_θ and integration ratio d are fine-tuned to achieve the best performance. The document frequency threshold df_θ for dataset Inspec, SemEval2017 and SemEval2010 are set to be 5, 5, and 44, respectively. For all datasets, we set the linear combination ratio d to be 0.8. For the baseline methods, the parameters published on the corresponding GitHub were used.

Based on our experiments, the computational cost to extract the attention between words is low since we can utilize a pre-trained BERT. It takes an average of 74 seconds to generate an attention matrix on a long document using our computer with an Intel i7 9700k CPU and 32G memory.

3.4 Results

Table 2 shows the results of recall, precision, and F1 @5, 10, and 15 using AttentionRank and baseline models on all datasets. A paired statistical significance test (used t-test when both are normal-distributed, otherwise used Kolmogorov-Smirnov test) is used to evaluate whether the F1 values of the AttentionRank model are statistically significantly better (p-value <0.01 ▼ or p-value<0.05 ▽). If p-value >0.05, there is no ▼ or ▽. It is worth noting that the difference is not statistically significant for those cases when AttentionRank performs slightly lower than the SIFRank model.

[3]https://github.com/boudinfl/pke

[4]https://github.com/LIAAD/yake

[5]https://github.com/swisscom/ai-research-keyphrase-extraction

[6]https://github.com/ymym3412/position-rank

[7]https://github.com/sunyilgdx/SIFRank

The results show that the embedding-based algorithms, including AttentionRank, perform better than the statistical-based (RAKE and YAKE!) and graph-based algorithms (SingleRank, TopicRank, and PositionRank) on short document sets (Inspec and SemEval 2017). SIFRank performs slightly better than AttentionRank on Inspec. AttentionRank works better than the other baselines on Inspec when K is set to 5 or 10. Nevertheless, AttentionRank has a slightly better F1 than SIFRank and other baselines when the top 15 candidates are used for evaluation. It shows that AttentionRank performs competitively on the dataset Inspec. AttentionRank outperforms all baselines on the other two datasets with statistical significance. AttentionRank shows advantage on long document set - SemEval2010. The F1 value is at least 3% better than the highest baseline.

Table 3 shows the comparison results with other unsupervised keyphrase extraction methods by referring to the reported in the original papers. These methods reported results on Inspec or SemEval2010. The comparison shows that AttentionRank works better than KeyGames (Saxena et al., 2020) on SemEval2010 by 0.5%, although KayGames used a designed candidate selection approach to remove noise. AttentionRank also works better than the MultipartiteRank (Boudin, 2018) and TeKET (Rabby et al., 2020) based on the original reported result on SemEval2010. Because the accumulated self-attention model considers the self-attention values accumulation of candidates over the document, AttentionRank works better on the long document set. The performance of AttentionRank on the Inspec is not as good as KeyGames, but better than the Salience Rank (Teneva and Cheng, 2017).

The PR curve (shown in Figure 3) is drawn using the top 60 ranked candidates generated by each method for overall comparison. The PR curves show consistent results that SIFRank works slightly better than AttentionRank on the Inspec dataset, whereas the AttentionRank outperforms all the baselines on both SemEval datasets. AttentionRank performs much better than YAKE! on SemEval2010, and both outperform the other baselines.

Table 2: Model Comparison with Precision (P), Recall (R), and F-score (F1) @5, @10, @15

k	Method	Inspec			SemEval2017			SemEval2010		
		P	R	F1	P	R	F1	P	R	F1
5	RAKE	27.40	11.94	16.63 ▼	26.94	8.78	13.24 ▼	1.4	0.44	0.67 ▼
	SingleRank	33.96	14.44	20.26 ▼	35.29	11.26	17.07 ▼	2.33	1.41	1.76 ▼
	TopicRank	29.80	12.15	17.26 ▼	33.43	10.45	15.92 ▼	10.37	3.52	5.26 ▼
	YAKE!	25.04	11.00	15.29 ▼	24.79	7.96	12.05 ▼	16.87	5.65	8.46 ▼
	PositionRank	34.80	14.89	20.85 ▼	40.57	12.60	19.23 ▼	5.16	1.62	2.46 ▼
	EmbedRank	40.92	17.50	24.52	47.10	14.59	22.28 ▼	3.29	1.10	1.64 ▼
	SIFRank	**44.00**	**18.40**	**25.95**	43.16	13.83	20.94 ▼	11.44	3.83	5.74 ▼
	AttentionRank	41.19	17.38	24.45	**49.17**	**15.52**	**23.59**	**22.27**	**7.66**	**11.39**
10	Rake	27.11	22.27	24.45 ▼	28.92	18.56	22.61 ▼	1.69	1.10	1.33 ▼
	SingleRank	29.97	24.25	26.81 ▼	32.90	20.52	25.28 ▼	2.23	2.53	2.37 ▼
	TopicRank	24.54	18.99	21.24 ▼	27.09	16.62	20.60 ▼	9.26	6.20	7.43 ▼
	YAKE!	19.48	16.67	17.96 ▼	23.33	14.86	18.16 ▼	14.94	10.00	11.98 ▼
	PositionRank	28.04	23.25	25.50 ▼	33.10	20.20	25.09 ▼	4.61	3.05	3.67 ▼
	EmbedRank	35.75	28.69	31.83	42.11	25.86	32.04 ▼	3.58	2.34	2.83 ▼
	SIFRank	**37.35**	**29.43**	**32.92**	40.24	24.94	30.80 ▼	7.82	5.18	6.23 ▼
	AttentionRank	37.17	28.32	32.15	**44.89**	**27.84**	**34.37**	**18.65**	**12.72**	**15.12**
15	Rake	24.59	28.53	26.41 ▼	27.69	26.10	26.87 ▼	1.81	1.75	1.78 ▼
	SingleRank	26.41	30.20	28.18 ▼	30.57	28.08	29.27 ▼	2.62	4.37	3.28 ▼
	TopicRank	21.74	23.37	22.52 ▼	23.60	21.27	22.37 ▼	7.98	8.06	8.02 ▼
	YAKE!	17.12	21.70	19.14 ▼	21.41	20.07	20.72 ▼	12.87	12.86	12.87 ▼
	PositionRank	24.09	29.04	26.33 ▼	29.00	26.50	27.69 ▼	4.15	4.02	4.08 ▼
	EmbedRank	32.07	35.63	33.76 ▽	37.85	34.10	35.88 ▼	3.65	3.63	3.64 ▼
	SIFRank	32.51	**35.73**	34.04	37.11	33.81	35.38 ▼	6.20	6.11	6.15 ▼
	AttentionRank	**34.58**	34.40	**34.49**	**40.43**	**36.22**	**38.21**	**16.51**	**16.82**	**16.66**

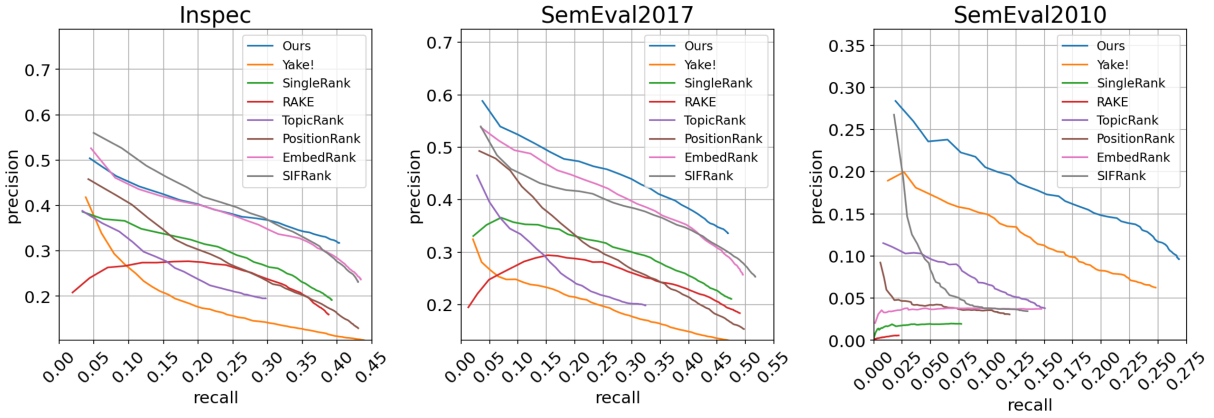


Figure 3: PR-curve Evaluation of Models Performance

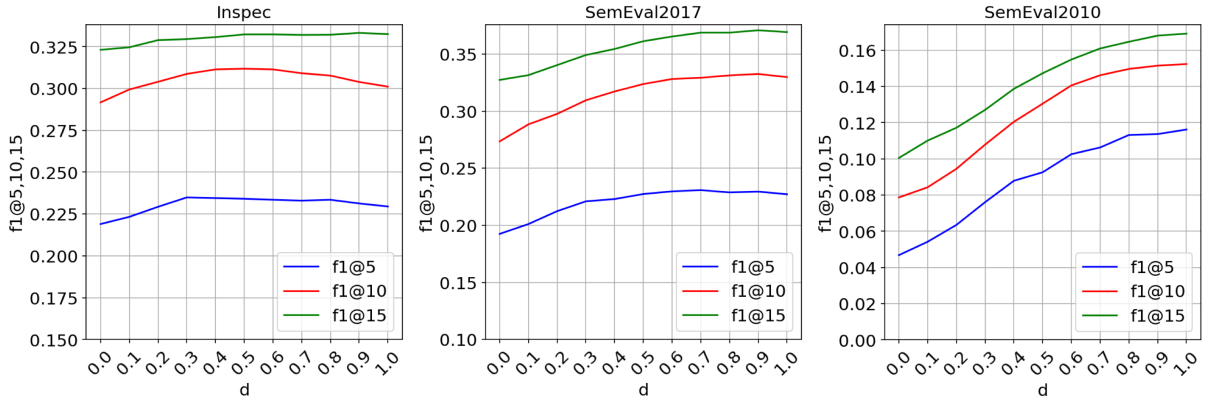


Figure 4: Evaluation of the Integration Ratio Impact on Performance

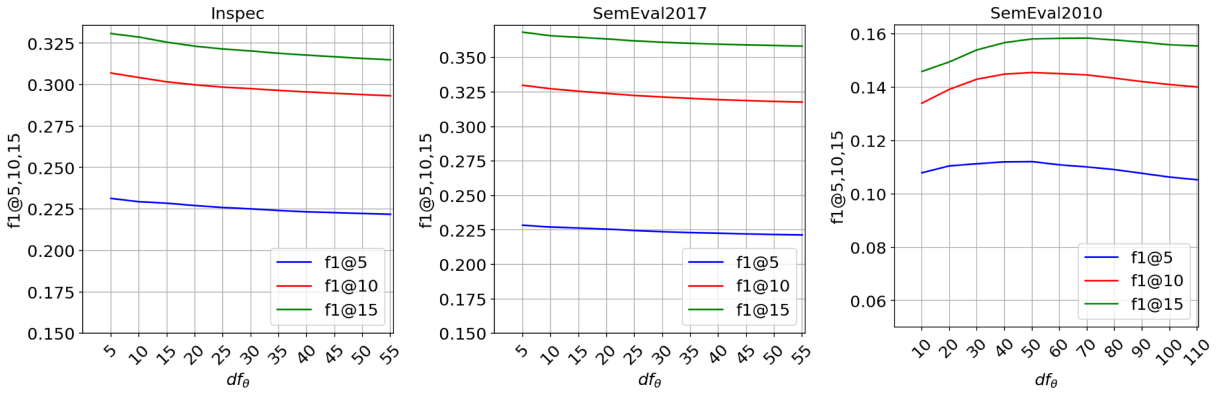


Figure 5: Evaluation of the Document Frequency Impact on Performance

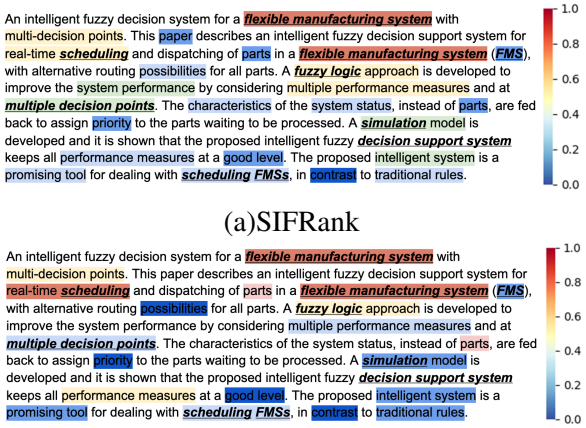
Table 3: Comparison of Others on Reported F1@10

Method	Inspec	SemEval2010
Salience Rank	26.60	-
MultipartiteRank	-	14.50
KeyGames	40.48	14.35
TeKET	-	14.75
AttentionRank	32.15	15.12

3.5 Ablation Study

3.5.1 Integration Ratio

The AttentionRank model linearly integrates the accumulated self-attention with the cross-attention relevance to determine the importance of a candidate. We study the influence of accumulated self-attention and cross-attention by adjusting d (in Equation 17) from 0 to 1, with a step size of 0.1. Figure 4 shows that the contribution of accumulated self-attention value is higher than cross-attention relevance in general. However, the best ratio is different for each dataset. With dataset Inspec, F1 value is highest when d is round 0.3, 0.5, and 0.9 for K equals 5, 10, and 15, respectively. For SemEval2017, the best performance can be achieved when d is set to 0.7, 0.9, and 0.9. For the long document set - SemEval2010, the performance is optimized when d is 1, which means only accumulated self-attention value is needed to find the keyphrases. We think that the accumulated self-attention model considers the repetition of the keyphrases implicitly through the self-attention weights accumulation over the document. However, for short document sets, such as Inspec, the cross-attention relevance has more impact. Since there are only a few sentences in a document, the repetition of the word is low. However, the context-wise relevancy between keyphrases and sentences and the document needs to be emphasized.



(a)SIFRank

(b)AttentionRank

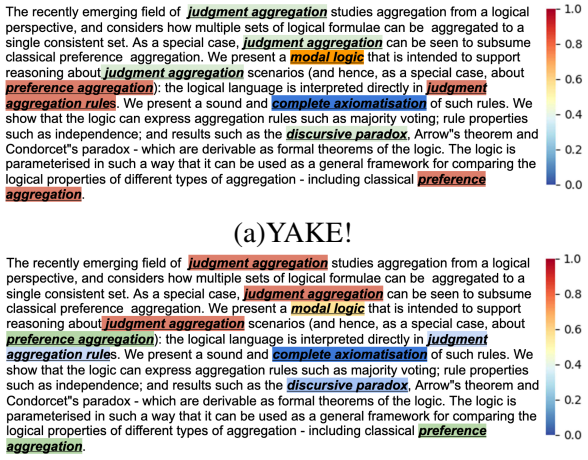
Figure 6: Comparison of SIFRank and AttentionRank on Short Document

3.5.2 Analysis on Document Frequency

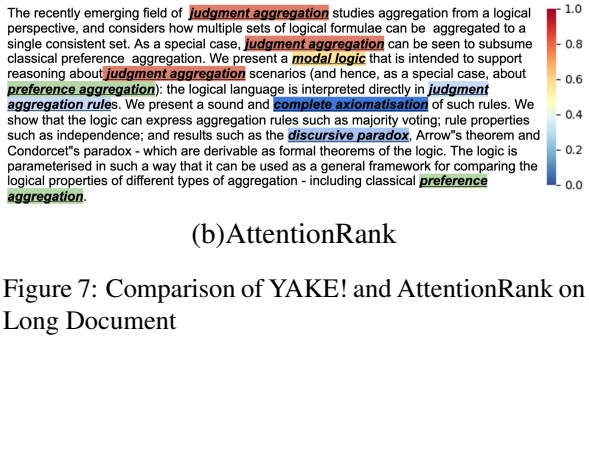
In the AttentionRank model, document frequency is used to remove generic terms for a specific corpus. Hence, we also study the impact of the document frequency threshold df_θ. For the short document set - SemEval2017 and Inspec, df_θ is adjusted from 5 or 10, whereas for the long document set - SemEval2010, df_θ is adjusted from 10 to 110. Figure 5 shows that different datasets can have different optimal performances based on the df_θ values. For short document datasets, the best df_θ is often small. The df_θ value for long document datasets is relatively larger. After df_θ is larger than a certain value, the performance drops with the increase of the df_θ, which means term with larger df_θ can be a keyphrase for a document within a specific corpus.

3.6 Case Study

AttentionRank performs closely with the SIFRank on short documents. To observe the difference between AttentionRank and SIFRank, we randomly



(a)YAKE!



(b)AttentionRank

Figure 7: Comparison of YAKE! and AttentionRank on Long Document

select a document in the Inspec. The heatmap in Figure 6 presents the importance scores of the candidates calculated by the two models. We normalized their original scores to draw the heatmap. The warmer the color is, the higher the score is. The phrases with bold italics and underline in the text are the labeled keyphrases. The candidate scores generated by AttentionRank fluctuate more than those generated by SIFRank. Using accumulated self-attention, AttentionRank generates slightly lower scores than SIFRank on some candidates, such as 'multiple decision points' which shows only once. The labeled keyphrase 'decision support system' is not identified by both models.

Although AttentionRank performs better than the other baselines on long documents, YAKE! also achieves good performance. Figure 7 shows a snippet of a long document with extracted keyphrases using these two models. The document is selected from the dataset SemEval2010.

Based on the observed heatmap, the scores are assigned differently by the models. The accumulated self-attention ranks the bigram candidate 'judgment aggregation' high because of its high frequency. Whereas, YAKE! counts the influence of frequency differently. Nevertheless, YAKE! emphasizes on a candidate using the length of the candidate. It generates a higher score on the trigram candidate 'judgment aggregation rules' higher than 'judgment aggregation'.

4 Related Work

There are supervised and unsupervised keyphrase extraction approaches. The unsupervised approaches can be grouped into two categories: traditional unsupervised methods and deep learning-based methods. Rose et al. (Rose et al., 2010) identified the keywords based on the word frequency, the number of co-occurring neighbors, and the ratio between the co-occurrence and the frequency. Campos et al. (Campos et al., 2018) calculated the importance of each candidate using frequency, offsets, and co-occurrence. Alrehamy et al. (Alrehamy and Walker, 2018) first clustered the candidates based on the semantic similarity. Candidates with similar semantic relevance with the centroids were selected as keywords. Rabby et al. (Rabby et al., 2020) took a binary tree approach. Its statistical relevance determined the importance of a subtree with its root. Then, they extracted all the paths from the roots to the leaves to find the keyphrases. Mihalcea et al. (Mihalcea and Tarau, 2004) converted the candidates into nodes on a graph, then used the PageRank algorithm to calculate the importance of the candidates Wan and Xiao et al. (Wan and Xiao, 2008) enriched the graph of candidates by collecting information from k-Nearest-Neighbor documents. Bougouin et al. presented the TopicRank (Bougouin et al., 2013) model, which first assigned a score to each topic by candidate keyphrases clustering. The topics were scored using the TextRank ranking model, and keyphrases were extracted using the most representative candidate from the top-ranked topics. Boudin et al. (Boudin, 2018) proposed a Multipartite graph model to encode the topic information within a multipartite graph, which utilized the keyphrases mutual relationship to improve ranking. Florescu et al. (Florescu and Caragea, 2017a) proposed the PositionRank to use the position information of word occurrences into TextRank to improve the TextRank on a document. Sung et al. (yeon Sung and Kim, 2020) considered the candidate hierarchy based on conditional probability (general or specific) on a directed weighted graph.

Wang et al. (Wang et al., 2014) made use of the pre-trained word embedding and the frequency of each word to generate weighted edges between words in a document. A weighted PageRank algorithm was used to compute the final scores of words. Mahata et al. (Mahata et al., 2018) used a similar approach using the phrase embeddings

for representing the candidates and ranking the importance of the phrases by calculating the semantic similarity and co-occurrences of the phrases. Papagiannopoulou et al. (Papagiannopoulou and Tsoumakas, 2018) also used word embeddings for unsupervised keyphrase extraction. Different from the previous two methods, the embeddings were trained from a single document called local embedding. Bennani-Smires et al. (Bennani-Smires et al., 2018) used a document embedding model, EmbedRank, to measure the similarity between documents and candidates to select more representative keyphrases. Sun et al. (Sun et al., 2020) proposed SIFRank, integration of statistical model and pre-trained language model, to calculate the relevance between candidates and document topic. Saxena et al. (Saxena et al., 2020) employed the concept of evolutionary game theory and utilized the embeddings, position, and frequency of the candidate to calculate the confidence score to determine whether a candidate is a keyphrase.

Different from previous methods, this study focuses on integrating self-attention weights extracted from the pre-trained deep language model with the calculated cross-attention relevancy value to identify the keyphrases that are important to local sentence context and also have strong relevancy to all sentences within the whole document.

5 Conclusions

This research investigated the accumulated self-attention mechanism integrated with a cross-attention model for unsupervised keyphrase extraction. A pre-trained BERT model is utilized to calculate the self-attention and cross-attention values. A candidate is processed through a self-attention calculation and a cross-attention relevancy calculation to gain a final score towards ranking. We compared the proposed AttentionRank model with seven different baselines on three benchmark datasets, including two short document datasets and one long document dataset. AttentionRank gains a better or competitive F1@5, 10, and 15 on all datasets. The ablation study shows that accumulated self-attention has a higher contribution than the cross-attention relevancy score on the long document set. For a short document set, the linear integration of both attention mechanisms shows good performance. The future work includes fine-tuning the BERT on a target domain and comparing against more baselines on domain-specific datasets.

References

Hassan Alrehamy and Coral Walker. 2018. Exploiting extensible background knowledge for clustering-based automatic keyphrase extraction. *Soft Computing*, 22(21):7041–7057.

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*, pages 2551–2557.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. 2016. Selectivity-based keyword extraction method. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 12(3):1–26.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, Nagoya, Japan. Asian Federation of Natural Language Processing.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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.

Corina Florescu and Cornelia Caragea. 2017a. A position-biased pagerank algorithm for keyphrase extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Corina Florescu and Cornelia Caragea. 2017b. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26.

Debanjan Mahata, John Kuriakose, Rajiv Shah, and Roger Zimmermann. 2018. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 634–639.

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.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Eirini Papagiannopoulou and Grigorios Tsoumakas. 2018. Local word vectors guiding keyphrase extraction. *Information Processing & Management*, 54(6):888–902.

Gollam Rabby, Saiful Azad, Mufti Mahmud, Kamal Z Zamli, and Mohammed Mostafizur Rahman. 2020. Teket: a tree-based unsupervised keyphrase extraction technique. *Cognitive Computation*, pages 1–23.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Arnav Saxena, Mudit Mangal, and Goonjan Jain. 2020. KeyGames: A game theoretic approach to automatic keyphrase extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2037–2048, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 755–764.

Nedelina Teneva and Weiwei Cheng. 2017. Salience rank: Efficient keyphrase extraction with topic modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 530–535.

Peter D Turney. 2000. Learning algorithms for keyphrase extraction. *Information retrieval*, 2(4):303–336.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, volume 39, pages 1–8.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI global.

Yoo yeon Sung and Seoung Bum Kim. 2020. Topical keyphrase extraction with hierarchical semantic networks. *Decision Support Systems*, 128:113163.

Ming Zhu, Aman Ahuja, Wei Wei, and Chandan K Reddy. 2019. A hierarchical attention retrieval model for healthcare question answering. In *The World Wide Web Conference*, pages 2472–2482.