AGRank: Augmented Graph-based Unsupervised Keyphrase Extraction

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

Abstract

Keywords or keyphrases are often used to highlight a document's domains or main topics. Unsupervised keyphrase extraction (UKE) has always been highly anticipated because no labeled data is needed to train a model. This paper proposes an augmented graph-based unsupervised model to identify keyphrases from a document by integrating graph and deep learning methods. The proposed model utilizes mutual attention extracted from the pre-trained BERT model to build the candidate graph and augments the graph with global and local context nodes to improve the performance. The proposed model is evaluated on four publicly available datasets against thirteen UKE baselines. The results show that the proposed model is an effective and robust UKE model for long and short documents. Our source code is available on GitHub[1].

1 Introduction

The mainstream unsupervised keyphrase extraction (UKE) approaches fall into one of three types: statistical, graph-based, and deep learning approaches. The statistical methods include the TF-IDF-based approach and other recent works (Campos et al., 2020; Beliga et al., 2016), which utilize term frequency, document frequency, word offsets and the number of n-grams to calculate the importance of the candidates. The graph-based methods treat the candidates as the nodes in a graph (Gollapalli and Caragea, 2014; Wan and Xiao, 2008). The edges are calculated based on candidates' co-occurrences, semantic similarity, or other relations. Graph-based algorithms then determine the importance of candidates. Several recent studies have shown that embedding-based methods can achieve excellent performance on unsupervised keyphrase extraction, such as JointModeling (Liang et al., 2021), AttentionRank (Ding and Luo, 2021), SIFRank (Sun et al., 2020), KeyGames (Saxena et al., 2020) and EmbedRank (Bennani-Smires et al., 2018). These approaches base candidates' importance on the distance or similarity of candidate embeddings, and some consider the global or local context.

We propose an augmented graph-based unsupervised model to identify keyphrases from documents. The model extracts attention from the pre-trained BERT model to generate a candidate keyphrase graph, then augments the attention graph with nodes that present the global and local context. Similar to the baseline approaches, noun phrases are extracted as candidates representing the nodes on the graph. The co-occurrence of candidates determines graph edges within the sentential context. Edge weights between the candidates are calculated based on the mutual attention extracted from the pre-trained BERT model and the indexes of the sentences where the candidates are located. The candidate graph adds the global and local contexts as document and sentence nodes. The edge weights between the document node and candidates and the edge weights between sentence nodes and candidates are calculated based on the cosine similarity between their embeddings. The graph is then adjusted by removing nodes and edges based on the document frequency and edge weights. Finally, the ranking of each candidate is calculated using the weighted PageRank algorithm.

We summarize our contributions as follows:

- A novel augmented graph-based unsupervised keyphrase extraction (UKE) model considering global and local context is proposed and evaluated using four benchmark datasets.
- The mutual attention extracted from the pre-trained language model is utilized to build a weighted graph.
- The proposed model works better than or is competitive with the state-of-the-art UKE baselines.

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

2 Methodology

Our model has three main parts: (1) Candidate Graph Generation, in which we convert each document into a weighted graph with candidates as nodes and attention between candidates in a sentential context as weighted edges; (2) Graph Augmenting, in which we add a document node and sentence nodes to emphasize the global and local context and their relations to the candidates; (3) PageRank Scoring, in which we apply the weighted PageRank algorithm on the graph to rank candidates to identify keyphrases.

2.1 Candidate Graph Generation

To build the candidate graph, we first extract candidates from a document, then add weighted edges between each pair based on the sentence level self-attention mechanism. Furthermore, the edge weights also are influenced by the importance of the sentences containing the candidates' pairs.

Candidates Generation. The candidates are extracted using the module implemented in the previous approach (Bennani-Smires et al., 2018). The module first uses part of speech (PoS) to tag the nouns, verbs, pronouns, and adjectives. Then, the noun phrases are extracted using the NLTK[2] package as candidates. In our research, the punctuation are removed from the candidates, except '-'. The stemming is applied to candidates and ground truth keyphrases for model building and performance evaluation. The effectiveness of stemming is investigated in the ablation study section.

Edge Weight Generation. The generation of edge weight is based on the mutual attention between candidates extracted from the pre-trained BERT model (Devlin et al., 2018). Clark et al. (2019) have shown that important syntactic and semantic information is captured in attention maps of the pre-trained BERT model. To compute the mutual attention between candidates, we utilize the methods introduced by Ding and Luo (2021) and Clark et al. (2019) to extract attention between words. The attention between words is then aggregated to attention between phrases.

For a sentence with n words, the mutual attention mapping between words can be presented as a matrix (A).

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

a_{ij} is the attention value that word w_i projects to word w_j within the same sentence s. If a candidate is a phrase with multiple words, we sum the word attention into phrase attention. Given candidate $c_1 = \{w : w_i \in c_1\}$ with n words and candidate $c_2 = \{w : w_j \in c_2\}$ with m words, the attention between c_1 and c_2 is the sum of the attention that the words in c_1 project to the words in c_2, shown as Equation 1.

$$a(c_1, c_2) = \sum_{i}^{n} \sum_{j}^{m} a_{ij} \qquad (1)$$

Fig. 1 shows a visual example of the mutual attention values between phrases. Given a document's title – "Standards for service discovery and delivery", the colored rows in the heatmap represent the attention project from words/phrases labeled on the y-axis to the words/phrases labeled on the x-axis.

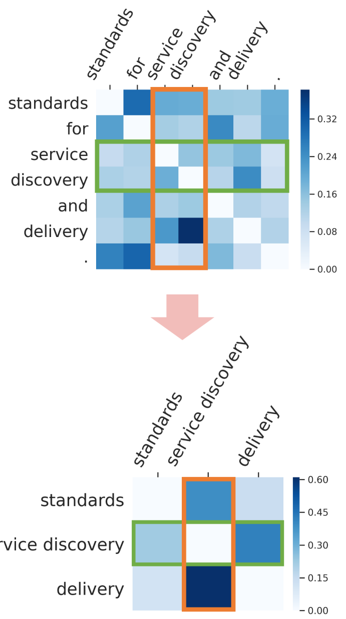


Figure 1: Attention aggregation from words to phrases (The attention values between identical words or phrases are set to zeros.)

$a(c_1, c_2)$ indeed represents the weight of the directed edge from c_1 to c_2 within sentence s, shown in Equation 2.

$$v_s < c_1, c_2 >= a(c_1, c_2) \qquad (2)$$

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

To generate undirected weighted edges, we sum edge weights from c_1 to c_2 and from c_2 to c_1 in all sentences containing c_1 and c_2, shown as Eq. 3.

$$v(c_1, c_2) = \sum_{s \in doc} (v_s < c_1, c_2 > + v_s < c_2, c_1 >) \tag{3}$$

Edge Weight Adjustment. Campos et al. (2020) has shown that the first few sentences of an article often summarize the main topic and emphasize the domain of the work. Therefore, we adjust the edge weights (v) according to the positions of the sentences (i_s) containing the edges (Eq. 4).

$$v(c_1, c_2) = v(c_1, c_2) \times [1 + (k - i_s)/10]^2, \; if \; i_s < k \tag{4}$$

The weight of edge (c_1, c_2) increases proportionally according to the index (i_s) of the first sentence containing candidates c_1 and c_2. k is the threshold for sentence position. When the sentence index exceeds k, the edges contained in the sentence have no weight adjustment. k can be fine-tuned in terms of the number of sentences based on the length of the document. For a long article, the threshold k can be set to the number of sentences in the abstract or introduction. Articles in different fields will have different k. In the following ablation study, we explored the effect of different k. k is designed to be a multiple of 10. For short documents containing less than ten sentences, k is set to 10.

2.2 Graph Augmenting

The candidate graph does not consider the relations between each candidate and the document's global context and the relations between the candidates and each sentence's local context. Hence, we add document and sentence nodes to augment the candidate graph with the global and local context.

Document Node. The candidates ($\{c_1, ..., c_r\}$) extracted from the document are concatenated as the document node representation. The edge weight between the document node d and a candidate node c is their embeddings' cosine similarity, shown in Equation 5.

The document node embedding (e_d) and the candidate node embedding (e_c) are generated by feeding the text representations of the document or candidate into a pre-trained BERT model. The self-attention mechanism of BERT generates a context-based embedding for each member word of a text. A document or candidate node's embedding is generated by summing up the member words' embeddings of the node. We use the bert-embedding[3] package to generate word-level embeddings.

The α_d is a coefficient value to adjust edge weights between the document and candidates. It can be set to the average number of sentences in a corpus.

$$v(d, c) = \frac{e_c \cdot e_d}{||e_c|| \cdot ||e_d||} \times \alpha_d \tag{5}$$

Sentence Nodes. A sentence node is represented using its original sentence content. The sentence node embedding (e_s) is generated using the same way as the document node embedding generation. The edge weight between candidate c and sentence s equals the cosine similarity of their embeddings, shown in Equation 6.

$$v(s, c) = \frac{e_c \cdot e_s}{||e_c|| \cdot ||e_s||} \tag{6}$$

Figure 2 shows a visualization example of an augmented graph of a document randomly selected from the dataset Inspec. The blue-colored nodes represent the stemmed candidates. The document node and the sentence nodes are pink-colored and green-colored, respectively. The edge weights between pairs of candidates and between candidates to document or sentence nodes are shown. For demonstration purposes, the edge weights are multiplied by ten and rounded. The original document content is shown in Fig. 3, and the ground truth keyphrases are highlighted. In this example, 'Service Location Protocol' is a labeled keyphrase. In the augmented graph, the edge weight between nodes '$servic\ locat\ protocol$' and '$race$' is high as calculated using BERT mutual attention. 'Service discovery' is another labeled keyphrase and occurs in four different sentences. Hence, in our augmented graph, the node '$servic\ discoveri$' has connections with many candidates. This example reveals that our augmented graph has the mechanism to emphasize the importance of the edges and the nodes based on the document content.

Graph Pruning. To reduce the computational cost and improve the performance, we prune the graph by removing some nodes based on their NLP features and some edges based on the edge weights distribution (Faralli et al., 2018). The following steps are applied:

[3]https://pypi.org/project/bert-embedding/

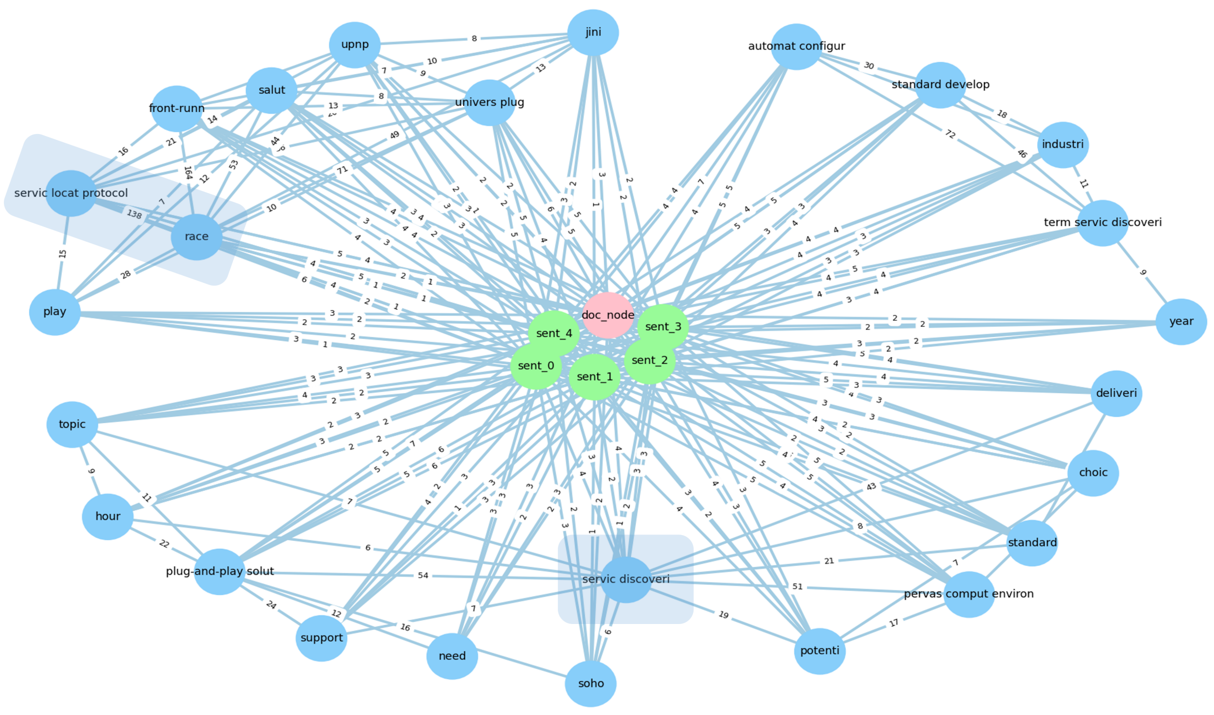


Figure 2: An example of an augmented graph. (All candidates are stemmed.)

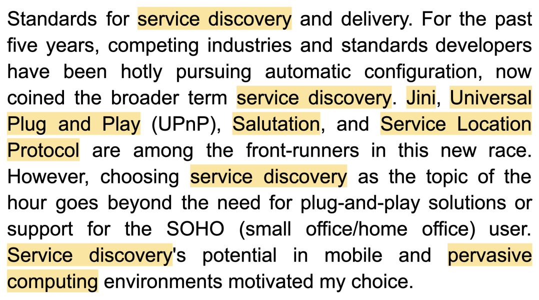

Standards for service discovery and delivery. For the past five years, competing industries and standards developers have been hotly pursuing automatic configuration, now coined the broader term service discovery. Jini, Universal Plug and Play (UPnP), Salutation, and Service Location Protocol are among the front-runners in this new race. However, choosing service discovery as the topic of the hour goes beyond the need for plug-and-play solutions or support for the SOHO (small office/home office) user. Service discovery's potential in mobile and pervasive computing environments motivated my choice.

Figure 3: Example document with ground truth keyphrases highlighted.

(1) Remove the candidate node when its document frequency exceeds some threshold. High document frequency often indicates that the term is a generic one in a corpus. For each corpus, we calculate the document frequency of all candidates and determine the threshold by the Elbow law[4].

(2) Remove the edge between a pair of candidates when the edge weight is lower than a threshold, such as the 25^{th} percentile of the candidate-candidate edge weights distribution.

(3) Remove the edge between a sentence and a candidate when the edge weight is lower than a threshold (p_s) determined by the sentence-candidate edge weights distribution.

[4]https://pypi.org/project/kneed/

2.3 PageRank Scoring

The pruned graph is fed into the weighted PageRank algorithm (Xing and Ghorbani, 2004) to calculate the importance score of each candidate. The score ($PR(c)$) of a candidate (c) is calculated as Equation 7.

$$PR(c) = (1-\delta) + \delta \times \sum_{c_n \in B_c} PR(c_n) \times v^2(c, c_n) \tag{7}$$

Where δ is the dampening factor, c_n is a neighbor node of c, and B_c is the set of all candidate c neighbors. $v(c, c_n)$ is the weight of the edge (c, c_n). The weighted PageRank algorithm considers in-edge and out-edge weights. Since we have an undirected graph, in-edge and out-edge weights are treated the same.

During the final ranking, the document and sentence nodes are excluded, and the candidates with a high document frequency, e.g., higher than a threshold df_θ, are also excluded.

3 Experiment

3.1 Datasets and Evaluation Metrics

The performance of our model is evaluated on four benchmark datasets[5]. Datasets Inspec

[5]https://github.com/LIAAD/KeywordExtractor-Datasets

(Hulth, 2003) and SemEval2017 (Augenstein et al., 2017) contain short documents, and datasets SemEval2010 (Kim et al., 2010) and Nguyen2007 (Nguyen and Kan, 2007) contain long documents. Table 1 summarizes the basic statistics of the datasets. The performance of keyphrase extraction is evaluated using F1 scores at the top 5, 10, and 15 ranked keyphrases.

To make an appropriate comparison with the baselines, we follow the common practice of using the uncontrolled annotated keyphrases of dataset Inspec and using the test set of SemEval2010 with 100 documents in our experiment. All extracted and labeled keyphrases are stemmed for evaluation.

Table 1: A Summary of Datasets

dataset	Document Number	Average Sentence Number	Average Word Number
Inspec	500	6	134
SemEval2017	493	7	168
SemEval2010	100	362	7845
Nguyen2007	209	235	5088

3.2 UKE Baselines

We compared our model against 13 baseline unsupervised keyphrase extraction models categorized into three categories: (1) Statistical models[6]: TF-IDF, YAKE! (Campos et al., 2020); (2) Graph-based models[7]: TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013), PositionRank (Florescu and Caragea, 2017b), MultipartiteRank (Boudin, 2018a); (3) Deep learning-based or mixed models: EmbedRank[8] (Bennani-Smires et al., 2018), SIFRank[9] (Sun et al., 2020), KeyGames[10] (Saxena et al., 2020), JointModeling[11] (Liang et al., 2021), AttentionRank[12] (Ding and Luo, 2021), MDERank[13] (Zhang et al., 2021).

3.3 Hyperparameter Setting

The BERT-Base is used for attention extraction (Clark et al., 2019) and node embedding generation[14]. The Hyperparameters for each dataset are fine-tuned and set as follows:

For all datasets, δ is set to 0.85, and α_d is set to the average sentence number of the corpus. For Inspec and SemEval2017, k is set to 10, df_θ is set to 5, and p_s is set to the 60^{th} and the 75^{th} percentile, respectively. For SemEval2010, k is set to 20, df_θ is set to 25. For Nguyen2007, k is set to 90, df_θ is set to 45. Sentence nodes are not added to the augmented graphs for SemEval2010 and Nguyen2007 due to the computational cost and the need.

On a computer with an Intel i7 9700k, 48G RAM and RTX 2060 graphics card, generating an augmented graph costs less than 10 seconds for a short document and about one minute for a long document.

3.4 Results

Table 2 compares AGRank and the baseline UKE models using F1@5, 10, and 15. The values for baseline models are those presented in the original papers or better results published in other papers recently. Since not all datasets are used in the original papers, we applied the baselines to the datasets using the published code. Those produced results are tagged with *.

In most cases, the deep learning-based or mixed models outperform the statistical and graph-based models on short document datasets (Inspec and SemEval2017). AGRank outperforms all UKE baselines on Inspec and performs better than most baselines except AttentionRank on SemEval2017.

Our proposed model has more apparent advantages on long document datasets. For the dataset SemEval2010, the F1@5 score is more than 3% higher than the best UKE baseline, and F1@10 and @15 are also about 2% higher than the best UKE baseline.

It is worth noting that the AGRank can often rank the keyphrases in the top 5. The results show that the F1@5 values gained by AGRank on all datasets are 1.5% - 3% higher than the best-performed UKE baseline model on Inspec, SemEval2010, and Nguyen2007. The F1@5 value gained by AGRank is also competitive with the best UKE baseline model - AttentionRank, on the SemEval2017 dataset.

[6] https://github.com/boudinfl/pke
[7] https://github.com/boudinfl/pke
[8] https://github.com/swisscom/ai-research-keyphrase-extraction
[9] https://github.com/sunyilgdx/SIFRank
[10] https://github.com/mangalm96/keygames-pke
[11] https://github.com/xnliang98/uke_ccrank
[12] https://github.com/hd10-iupui/AttentionRank
[13] https://github.com/linhanz/mderank
[14] https://pypi.org/project/bert-embedding/

Table 2: Model Comparison based on F1@5, @10, @15

Method	Inspec			SemEval2017			SemEval2010			Nguyen2007		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Statistical Models												
TF-IDF	11.28	13.88	13.83	12.70	16.26	16.73	2.81	3.48	3.91	8.66*	11.03*	12.42*
YAKE!	18.08	19.62	20.11	11.84	18.14	20.55	11.76	14.40	15.19	15.63*	17.46*	17.63*
Graph-based Models												
TextRank	27.04	25.08	36.65	16.43	25.83	30.50	3.80	5.38	7.65	1.07*	2.35*	2.95*
SingleRank	27.79	34.46	36.05	18.23	27.73	31.73	5.90	9.02	10.58	1.86*	3.55*	4.56*
TopicRank	25.38	28.46	29.49	17.10	22.62	24.87	12.12	12.90	13.54	11.23*	13.36*	13.18*
PositionRank	28.12	32.87	33.32	18.23	26.30	30.55	9.84	13.34	14.33	6.35*	9.89*	10.25*
MultipartiteRank	25.96	29.57	30.85	17.39	23.73	26.87	12.13	13.79	14.92	13.49*	15.63*	16.50*
Deep Learning-based or Mixed Models												
EmbedRank d2v	31.51	37.94	37.96	20.21	29.59	33.94	3.02	5.08	7.23	4.47*	6.39*	7.18*
SIFRank	29.11	38.80	39.59	22.59	32.85	38.10	8.32*	8.69*	8.78*	9.40*	9.55*	8.88*
KeyGames	32.12	40.48	40.94	16.04*	24.86*	29.48*	11.93	14.35	14.62	15.02*	15.68*	14.30*
JointModeling	32.61	40.17	41.09	19.17*	29.59*	35.68*	13.02	19.35	21.72	11.52*	15.93*	17.71*
AttentionRank	31.55	39.16	40.65	**24.45**	**35.24**	**39.06**	12.72	17.21	19.15	17.22*	20.63*	**22.01***
MDERank(BERT)	26.17	33.81	36.17	22.81	32.51	37.18	12.95	17.07	20.09	14.47*	17.45*	17.44*
AGRank	**34.59**	**40.70**	**41.15**	24.13	33.46	37.21	**15.37**	**21.22**	**23.72**	**18.76**	**22.16**	21.74

Table 3: Ablation Study

Method	Inspec			SemEval2017			SemEval2010			Nguyen2007		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Stemming Ablation												
AGRank	34.59	40.70	**41.15**	**24.13**	**33.46**	**37.21**	15.37	21.22	**23.72**	18.76	22.16	**21.74**
w/o Stemming	**35.32**	**40.98**	40.57	22.84	32.59	36.62	14.79	19.95	21.34	13.76	16.65	16.37
Graph Augmenting Ablation												
w/o Doc. Node	33.86	40.31	41.08	23.78	33.32	36.85	**15.38**	**22.10**	23.38	**19.13**	**22.33**	21.66
w/o Sent. Nodes	34.15	40.21	40.78	23.67	33.03	36.83	-	-	-	-	-	-
Edge Weight Adjustment based on Sentence Position												
w/o Sent. Weight Adjust.	34.13	40.56	40.98	23.96	33.10	36.91	13.74	17.22	18.48	18.37	20.14	19.81

4 Ablation Study

4.1 Analysis of Stemming

Candidate stemming causes nodes to merge and change the graph's structure. Table 3 compares the performance of AGRank with and without stemming. The results show that stemming improves the model performance on SemEval2017, SemEval2010 and Nguyen2007. However, the improvements on the Inspec are not significant.

4.2 Analysis of Graph Augmenting and Edge Weight Adjustment

The proposed model augments the graph by adding document and sentence nodes to provide global and local context. We present the impact of the context nodes in Table 3. The model takes better advantage of document node addition on Inspec. In contrast, the sentence node addition contributes more to the model performances on SemEval2017.

Interestingly, the model performance on SemEval2010 and Nguyen2007 are marginally better without document node addition. We think the document node generated for a long document cannot sufficiently capture the overall context by generating one single embedding.

In our model, the weights of edges between candidates are also adjusted according to the sentence position. From Table 3, the edge weight adjustment based on sentence position has a higher impact on SemEval2010 and Nguyen2007. Without using it, the performance could drop up to 2%.

4.3 Analysis of Hyperparameters

We evaluated the impact of the hyperparameters of our model. Fig. 4 shows the hyperparameter tuning of k - the parameter to adjust edge weights by sentence position, p_s - the parameter to remove the edges between the sentences and candidates based on weight distribution, and df_θ - the parameter to exclude the candidates based on document frequency. Note that the tuning study of parameter k only applies to long documents. For short documents with less than ten sentences, k is set to 10. The parameter p_s is only applicable to short documents since sentence nodes are not added to the augmented graphs for long documents due to the computational cost.

We investigated the impact of threshold k from 10 to 130 with a step size of 10. Fig. 4 shows that for the long document dataset SemEval2010, the best F1@15 is gained when the first 20 sentences are considered. Whereas for the long document dataset Nguyen2007, the highest F1@15 is achieved when the first 90 sentences are considered. These results show that adjusting candidates' mu-

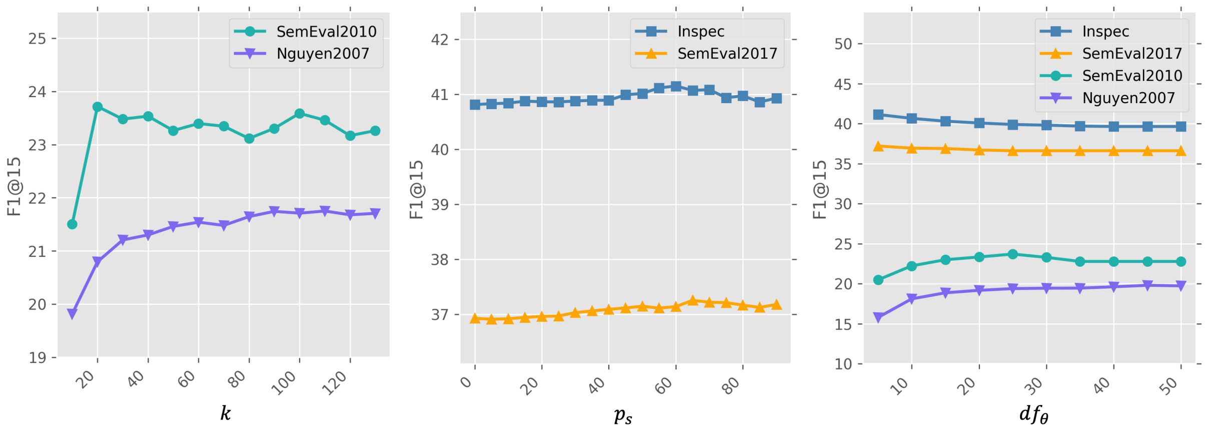


Figure 4: Evaluation of the Hyperparameters on Model Performance

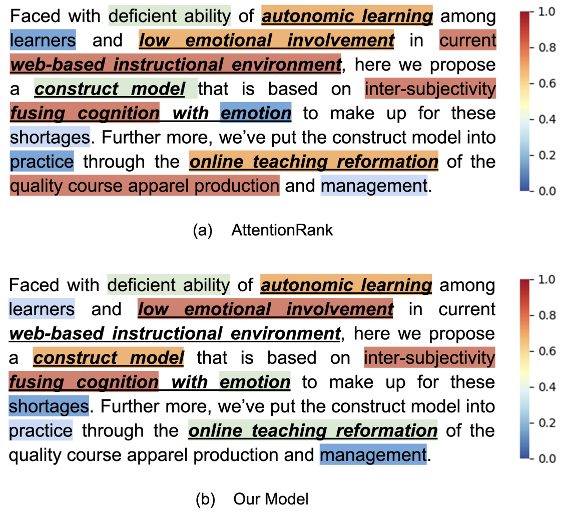
tual edge weights by sentence position improves model performance, although k needs to be tuned for different datasets.

We showed that adding sentence nodes can slightly improve the performance on short document sets, but the sentence nodes might not have strong relationships with all the candidates. Tuning the number of edges between sentence nodes to candidates can reduce the computational cost and optimize the model performance. We adjusted the p_s from 0 to the 90th percentile based on the weight distribution. Fig. 4 shows that for datasets Inspec and SemEval2017, optimal p_s are between the 60th and the 80th percentile.

We also tuned df_θ to see its impact on the performance. Fig. 4 shows that df_θ has less impact on short document sets – Inspec and SemEval2017. Performance of F1@15 can improve about 2% after tuning the df_θ on long document sets – SemEval2010 and Nguyen2007.

4.4 Case Study

AGRank performs closely with the AttentionRank on short documents. To observe the difference between AGRank and AttentionRank, we randomly select a document in SemEval2017. The heatmap in Fig. 5 presents the importance scores of the candidates calculated by the two models. We normalized the original scores to highlight the candidates with a heatmap. The labeled keyphrases are bold, italic, and underlined. AGRank scores higher for keyphrases 'construct model' and 'low emotional involvement', whereas the AttentionRank ranks 'online teaching reformation' higher. Since AttentionRank uses accumulated self-attention, long candidates with multiple words obtain higher scores.


Faced with deficient ability of ***autonomic learning*** among learners and ***low emotional involvement*** in current ***web-based instructional environment***, here we propose a ***construct model*** that is based on inter-subjectivity ***fusing cognition with emotion*** to make up for these shortages. Further more, we've put the construct model into practice through the ***online teaching reformation*** of the quality course apparel production and management.

(a) AttentionRank

Faced with deficient ability of ***autonomic learning*** among learners and ***low emotional involvement*** in current ***web-based instructional environment***, here we propose a ***construct model*** that is based on inter-subjectivity ***fusing cognition with emotion*** to make up for these shortages. Further more, we've put the construct model into practice through the ***online teaching reformation*** of the quality course apparel production and management.

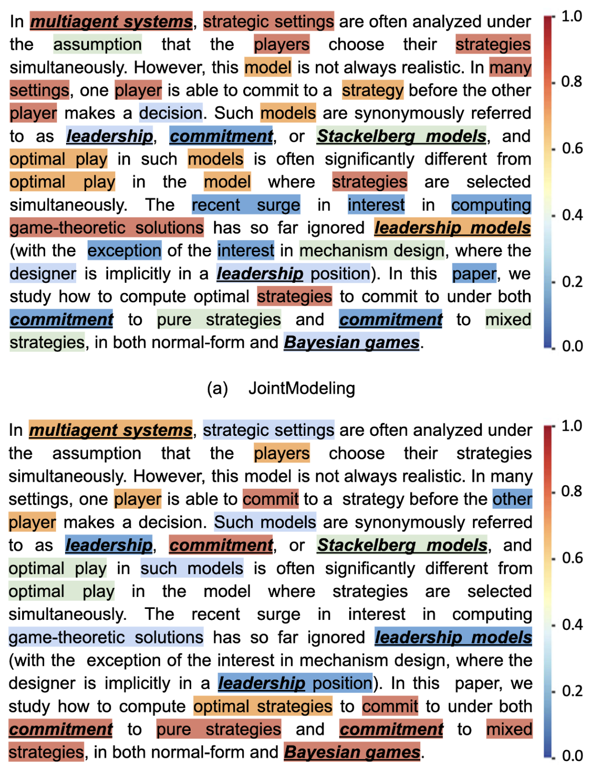
(b) Our Model

Figure 5: Comparison on Short Document

JointModeling performs well on the long document set SemEval2010. Fig. 6 shows the performances of JointModeling and AGRank on a selected paragraph taken from an article in SemEval2010. The heatmap shows the difference in the strategies of the two models. AGRank has fewer candidates than JointModeling, which attribute to our graph pruning step. The candidates with high document frequency and small neighbor edge weights are removed. Since the edge weights of the augmented graph are generated based on the extracted attention of the pre-trained BERT model, AGRank assigns high scores to 'commitment' and 'Bayesian games'.

5 Related Works

The unsupervised keyphrase extraction approaches can be categorized into statistical, graph-based, and deep learning-based or mixed methods. The mod-



In ***multiagent systems***, strategic settings are often analyzed under the assumption that the players choose their strategies simultaneously. However, this model is not always realistic. In many settings, one player is able to commit to a strategy before the other player makes a decision. Such models are synonymously referred to as ***leadership***, ***commitment***, or ***Stackelberg models***, and optimal play in such models is often significantly different from optimal play in the model where strategies are selected simultaneously. The recent surge in interest in computing game-theoretic solutions has so far ignored ***leadership models*** (with the exception of the interest in mechanism design, where the designer is implicitly in a ***leadership*** position). In this paper, we study how to compute optimal strategies to commit to under both ***commitment*** to pure strategies and ***commitment*** to mixed strategies, in both normal-form and ***Bayesian games***.

(a) JointModeling

In ***multiagent systems***, strategic settings are often analyzed under the assumption that the players choose their strategies simultaneously. However, this model is not always realistic. In many settings, one player is able to commit to a strategy before the other player makes a decision. Such models are synonymously referred to as ***leadership***, ***commitment***, or ***Stackelberg models***, and optimal play in such models is often significantly different from optimal play in the model where strategies are selected simultaneously. The recent surge in interest in computing game-theoretic solutions has so far ignored ***leadership models*** (with the exception of the interest in mechanism design, where the designer is implicitly in a ***leadership*** position). In this paper, we study how to compute optimal strategies to commit to under both ***commitment*** to pure strategies and ***commitment*** to mixed strategies, in both normal-form and ***Bayesian games***.

(b) Our Model

Figure 6: Comparison on Long Document

els based on statistical techniques convert contextual information into statistical features of candidates and then calculate candidate scores for ranking. Rose et al. (2010) utilized the ratio of word frequency and the number of co-occurring neighbors to evaluate the importance of the candidates. Besides term frequency and neighbor co-occurrence, Campos et al. (2020) also considered more contextual features to identify keyphrases, including the offsets of the candidates, the sentence positions of the candidates first shown, etc. Models based on graph methods treat candidates as nodes of the graph, convert certain relations between candidates into edges of the graph, then use a graph algorithm to calculate the candidates' scores (Mihalcea and Tarau, 2004). Wan and Xiao (2008) utilized a clustering method to select k-Nearest-Neighbor documents to create a graph for a single document and used a graph sorting algorithm to generate keyphrases. Bougouin et al. (2013) employed a clustering method to generate several topics of a document and assign the topics to candidates, then utilized the TextRank model to rank topics; the most representative candidates of the top-ranked topics are extracted as keyphrases. Wang et al. (2014) utilized the word embedding and word frequency to generate weighted edges between words, then used the weighted PageRank algorithm to compute candidate scores and rankings. Florescu and Caragea (2017a) proposed the Position-Biased PageRank algorithm, which incorporates the candidate positions in the document into the ranking calculation. Boudin (2018b) proposed the Multipartite graph model, which encodes the topic information within a multipartite graph to utilize candidate mutual relations. yeon Sung and Kim (2020) extracted hierarchical relationships to determine which edges and phrases should be used and evaluated the nodes according to their inflowing edges. Bennani-Smires et al. (2018) proposed the EmbedRank, which uses a pre-trained language model to generate the document and candidate embeddings and calculate the similarity between them to select more representative keyphrases. Sun et al. (2020) proposed SIFRank, which invokes both the similarity between candidate and document embeddings and the candidate position and frequency to calculate the correlation between candidates and the document. Saxena et al. (2020) investigated an evolutionary game theory model that uses candidate embeddings and statistics to calculate confidence scores to determine whether a candidate is a keyphrase. Ding and Luo (2021) extracted attention mapping weights and then integrated accumulated attention weights with the cross-attention similarity to rank the candidates. Liang et al. (2021) integrated bounded sentences and candidate local relations based on document-to-candidate global relations, then used both jointly to determine the importance of candidates. Zhang et al. (2021) proposed MDERank, which ranked candidates using the similarity between the BERT embeddings of the source document and the masked document.

6 Limitations, Conclusions, and Future Work

Although our augmented graph-based model performs better than the compared baselines, the graph augmentation process is designed with quite a few hyperparameters that need to be tuned for datasets of different domains to obtain optimal performance. We believe this can be further improved by automating the hyperparameter tuning process.

Our research investigated the integration of graph-based and deep learning-based models for unsupervised keyphrase extraction. The pre-trained BERT model is utilized to extract candidates' mutual attention to build the initial graph. Global and local context information are added through graph augmenting. PageRank algorithm is used to calculate the ranking scores. We compared the proposed model against 13 baseline unsupervised keyphrase extraction models on four benchmark datasets. The ablation study shows that the edge weight adjustment based on sentence position has a higher impact on the long document sets. Adding the document and sentence nodes improves the performance for short document sets.

Future work includes investigating possible solutions to reduce the number of parameters and improve efficiency. We also plan to compare our unsupervised model against supervised keyphrase extraction models to demonstrate the advantages and performances.

References

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. 2018a. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

Florian Boudin. 2018b. 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 Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

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. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Haoran Ding and Xiao Luo. 2021. Attentionrank: Unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1928.

Stefano Faralli, Irene Finocchi, Simone Paolo Ponzetto, and Paola Velardi. 2018. Efficient pruning of large knowledge graphs. In *IJCAI*, pages 4055–4063.

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.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. *arXiv preprint arXiv:2109.07293*.

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.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.

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.

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.

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.

Wenpu Xing and Ali Ghorbani. 2004. Weighted pagerank algorithm. In *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.*, pages 305–314. IEEE.

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

Linhan Zhang, Qian Chen, Wen Wang, Chong Deng, Shiliang Zhang, Bing Li, Wei Wang, and Xin Cao. 2021. Mderank: A masked document embedding rank approach for unsupervised keyphrase extraction. *arXiv preprint arXiv:2110.06651*.