LLMRank: Enhancing Large Language Models for Unsupervised Keyphrase Extraction with a Candidate Graph Approach

Abstract

Keyphrase extraction is a crucial NLP task that extracts essential information from extensive texts, aiding in content summarization and browsing. This paper introduces LLMRank, a novel unsupervised keyphrase extraction method that augments Large Language Models (LLMs) with a graph-based approach. LLMs are first used to generate a wide array of candidate keyphrases, which are then represented as nodes in a custom graph. Edges between these nodes are established based on the co-occurrence of candidates within the content, enhancing keyphrase ranking through structured contextual information. We evaluated LLMRank using three state-of-the-art LLMs across four publicly available datasets, comparing its performance against seventeen baseline models. The results demonstrate that LLMRank effectively extracts keyphrases from long and complex documents in an unsupervised manner. Source code is available on GitHub[1].

1 Introduction

In the field of natural language processing (NLP), Keyphrase Extraction (KPE) is essential for various applications, including information retrieval, text summarization, and document clustering. Keyphrases capture the core meaning of texts, improving metadata for indexing and retrieval while providing concise summaries of lengthy documents. Traditionally, KPE methods fall into two categories: supervised KPE, which requires extensive labeled data for model training, and unsupervised KPE, which does not need labeled data for model training.

Unsupervised KPE removes the need for costly annotations but often relies on statistical metrics or researcher-developed unsupervised learning models that may not fully capture the semantic nuances inherent in natural language. The emergence of Large Language Models (LLMs) such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) has transformed this field. GPT operates as a unidirectional, autoregressive model specializing in text generation through its decoder-only architecture. Conversely, BERT is engineered to process text bidirectionally, which enhances its understanding. It is pre-trained on two interconnected NLP tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). These advanced transformer models effectively capture contextual relationships and long-range dependencies, providing a deep understanding of language nuances.

Recently, generative LLMs have gained popularity across various fields due to their impressive text generation capabilities, cross-domain versatility and ease of access and use. These models have shown remarkable effectiveness in capturing contextual meanings, significantly improving performance across language tasks. However, they struggle with a comprehensive understanding of entire texts, especially in tasks like KPE where understanding the full context of lengthy documents is crucial. While GPT-3 and GPT-4 excel in generating text and dialogue systems, they often miss the broader structure of extensive documents, limiting their effectiveness in tasks requiring a deep grasp of content. To overcome these limitations, some approaches have integrated enhancements such as knowledge graphs [9][28] and long-term memory networks [30][34] to improve the long-text processing capabilities of generative LLMs.

We propose a method called LLMRank that enhances keyphrase extraction by augmenting generative Large Language Models (LLMs) with a graph-based approach. Initially, the generative LLM identifies and ranks candidate keyphrases. These candidates then undergo an improved ranking process using a graph-based method that leverages contextual information on a global scale within the

[1]https://github.com/emnlp2024code/LLMRank

document. In this graph-based approach, the candidate keyphrases identified by the LLM serve as nodes, while edges are weighted based on both the distance and co-occurrence of these candidates within the content. The nodes are subsequently ranked using a PageRank algorithm to determine their importance in the document.

We outline our contributions as follows:

- We introduce LLMRank, a method that enhances keyphrase extraction from Large Language Models (LLMs) by incorporating a graph-based approach.

- We demonstrate that the proposed LLMRank effectively captures the content structure and phrase co-occurrence relationships within a document, thereby enhancing the performance of LLMs for Unsupervised Keyphrase Extraction (uKE).

- We evaluate our enhanced model against standard baselines across four benchmark datasets, demonstrating that the proposed LLMRank outperforms in handling complex and long texts.

2 Methodology

Our research introduces a novel method that leverages the capabilities of an LLM for keyphrase extraction. Given an input text, a selected LLM extracts the top candidate keyphrases. These candidate keyphrases are then organized into a weighted graph based on their relevance and co-occurrences within the input text. A dedicated PageRank algorithm is applied to produce a score reflecting their importance from a different perspective. The scores generated by the LLM and the PageRank algorithm are then combined to rank the candidate keyphrases. This approach integrates the semantic analysis strengths of LLMs with the importance-ranking capability of PageRank—an algorithm traditionally used to rank web pages in web search, but here adapted for the textual domain. Figure 1 provides a comprehensive overview of the system framework.

2.1 Candidate Generation and Ranking using LLM

First, we generate and rank the candidate keyphrases using an LLM. Prompt, shown in the Table 1, is designed and optimized to guide the LLM to output the candidate keyphrases and importance scores.

The prompt design takes into account the following three factors:

Top K Keyphrases: Depending on the length of the given text, the LLM is directed to extract only the top K keyphrases. For long documents, we increase the K value to capture various topics embedded in the content. Conversely, for short documents with a limited number of topics, setting a high K value could lead to redundancy in the keyphrases. Therefore, a smaller K value is chosen. In the experimental section, we demonstrate the impact of the K value.

Importance Score: The prompt is designed for the LLM to assign an importance score to each keyphrase, reflecting its relevance to the main topics in the text and its contextual significance within the document. Based on the importance scores, the list of candidate keyphrases can be ranked for performance calculation.

Original or Generative Text: In the prompt, we either instruct the LLM to extract keyphrases that must occur in the original text to avoid generating content that might not accurately represent the content, or we allow the LLM to produce generative content. This setup enables us to assess whether the LLM can generate keyphrases based on a deep understanding of the content. If we instruct the LLM to use the original text, we include the last sentence in the prompt; otherwise, the last sentence is omitted.

Role	Content
System	Extract top K keyphrases from the input text. Each keyphrase contains 4 or less grams. For each keyphrase, include a numerical value representing its importance. For example: 1. keyphrase - 9.1 2. keyphrase - 3.9 3. keyphrase - 5.2 The keyphrases must be the exact phrases that occur in the input text.
User	Input Text

Table 1: Designed Prompt for Keyphrase Extraction

As a result, LLM generates a set of candidate keyphrases $C = \{c_1, c_2, ..., c_n\}$ for each input text T. These candidates are further graphed in the next stage. Their rankings $R_{LLM} = \{l_{c_1}, l_{c_2}, ..., l_{c_n}\}$ based on their importance scores are used again in the overall integration stage.

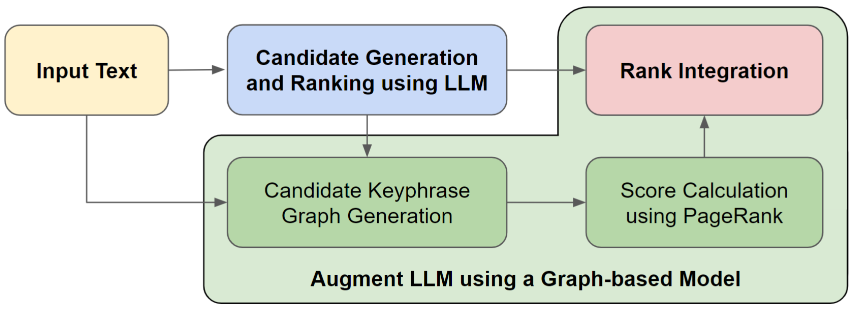
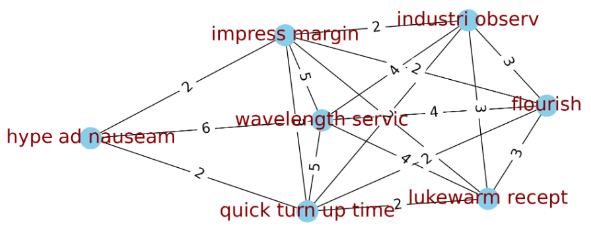

Input Text
Candidate Generation and Ranking using LLM
Rank Integration
Candidate Keyphrase Graph Generation
Score Calculation using PageRank
Augment LLM using a Graph-based Model

Figure 1: System Overview



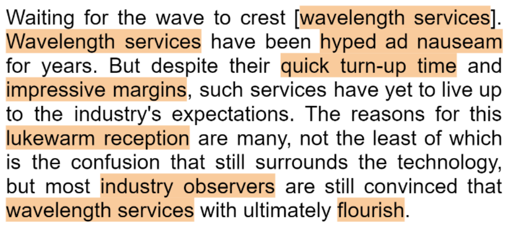

Waiting for the wave to crest [wavelength services]. Wavelength services have been hyped ad nauseam for years. But despite their quick turn-up time and impressive margins, such services have yet to live up to the industry's expectations. The reasons for this lukewarm reception are many, not the least 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: An example of a candidate phrases graph.

2.2 Augment LLM using a Graph-based Model

To enhance the LLM using a graph-based model, we first construct a graph from the candidate keyphrases generated by the LLM and then apply the PageRank algorithm to produce a score. Finally, we combine the score generated by the LLM with the score generated by the PageRank algorithm in a linear fashion.

2.2.1 Candidate Keyphrase Graph Generation

Our method employs a unique graph-based model designed to effectively evaluate the relevancy and co-occurrences of keyphrases. In this model, the top candidate keyphrases generated by LLMs are stemmed and treated as nodes in the graph. The edges between these nodes are determined based on the co-occurrences of the candidate keyphrases within a defined window in terms of number of sentences in the input text.

The weight on the edge is calculated as follows:

Within an input text T which has q sentences $S = \{s_1, s_2, ..., s_q\}$, each candidate keyphrase c occurs in a list of r sentences, $O_c = \{o_1, o_2, o_3..., o_r\}$, $0 \leq r \leq q$, where o_k is the index of sentence s_k.

A window of co-occurrence is defined as distances between the two sentences containing the two candidate keyphrases c_i and c_j are within a window of w sentences. If two candidate keyphrases c_i and c_j occur in multiple sentences O_{c_i} and O_{c_j}, respectively, and within the defined w of window of co-occurrences, the weight on the edge ($e_{i,j}$) between c_i and c_j are cumulatively summed, as shown in Eq. 1, where $o_i \in O_{c_i}, o_j \in O_{c_j}$.

$$e_{i,j} = \sum(w - |o_i - o_j|),\ if\, |o_i - o_j| \leq w \quad (1)$$

Figure 2 illustrates a stemmed candidate keyphrase graph (left side) generated from a selected document (right side) in the Inspec dataset. In this graph, the blue nodes represent stemmed candidate keyphrases produced by an LLM. If the co-occurrence window is set to 3, it means that the candidate keyphrases must co-occur within a distance of three sentences or fewer.

The candidate keyphrases are highlighted in yellow in the selected document. For example, the keyphrase "flourish" appears once in the 4th sentence, and "impressive margins" appears once in the 3rd sentence, resulting in a weighted edge of 2 linking these two nodes. The closer the candidate keyphrases are to each other, the higher the weight of the edge. The keyphrase "wavelength services" appears three times, in the 1st, 2nd, and 4th sentences, thereby connecting with the most nodes. It also has higher weights on the edges, reflecting its multiple co-occurrences with other candidate keyphrases within the window of co-occurrence.

2.2.2 Keyphrase Ranking using PageRank

The stemmed graph is evaluated through the application of the weighted PageRank algorithm [31], which computes the significance scores for each candidate. These scores, designated as $PR(c)$ for each candidate c, are determined in accordance with Equation 2:

$$PR(c) = (1-\delta) + \delta \times \sum_{c_n \in B_c} PR(c_n) \times e^2_{c,c_n} \quad (2)$$

In this equation, δ represents the dampening factor which is set to be 0.85 in this research; c_n indicates a neighboring node of c; and B_c comprises all neighboring nodes of c. The function e_{c,c_n} refers to the edge weight between c and c_n. The algorithm accounts for both incoming and outgoing edge weights, which are considered equally due to the undirected nature of the graph. PageRank scores, $\{PR(c_i), c_i \in C\}$, is used to generate the rankings $R_{PageRank} = \{p_{c_1}, p_{c_2}, ..., p_{c_m}\}$ of the candidate keyphrases. This rankings will be used to integrate with the LLM-generated rankings.

2.2.3 Ranking Integration

We consider the rankings obtained from the LLMs ($R_{LLM} = \{l_{c_1}, l_{c_2}, ..., l_{c_n}\}$) and those derived from PageRank ($R_{PageRank} = \{p_{c_1}, p_{c_2}, ..., p_{c_m}\}$) as separate ranking scores, which are subsequently integrated linearly as outlined in Equation 3. d indicates an integration ratio, ranging from 0 to 1, which indicates the contribution from the LLM. Thus, a lower f_c suggests higher ranking of the candidate. It is important to note LLM can generate candidates that do not appear in the input text T. These candidates do not have a corresponding ranking calculated using the PageRank. Hence, the only LLM-generated ranking is used in calculation f_c.

$$f_c = \begin{cases} d * l_c + (1-d) * p_c, & if\ c \in T \\ l_c, & if\ c \notin T \end{cases} \quad (3)$$

After calculating f_c for each candidate keyphrase, they are ranked in ascending order for performance evaluation.

3 Experiments

3.1 Datasets and Evaluation Metrics

The performance of the LLMRank is evaluated using four established benchmark datasets [2]. The datasets Inspec [14] and SemEval2017 [1] feature short documents, whereas SemEval2010 [16] and Nguyen2007 [20] are comprised of longer documents. Table 2 provides a summary of the basic statistics for these datasets. F1 scores calculated based on the top 5, 10, and 15 are used for performance comparison. For During the evaluation process, both extracted and labeled keyphrases, are stemmed before performance calculation.

Table 2: Table of Basic Statistics for the 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

Our model was benchmarked against 14 baseline unsupervised keyphrase extraction models, divided into four distinct groups: (1) Statistical models[3]: TF-IDF, YAKE! [7]; (2) Graph-based models[4]: TextRank [19], SingleRank [27], PositionRank [13], MultipartiteRank [4]; (3) Deep learning-based or mixed models: EmbedRank[5] [3], SIFRank[6] [26], KeyGames[7] [22], JointModeling[8] [18], AttentionRank[9] [10], MDERank[10] [33], HyperRank [25]; (4) Generative LLM-based models: PromptRank[11] [17].

3.3 Hyperparameter Setting

In our experiments, we evaluated three different LLMs, including GPT-3.5 (gpt-3.5-turbo-0125[12]) and GPT-4o (gpt-4o-2024-05-13[13]), both under the OpenAI API license for non-profits with a maximum context length of 16385 tokens, and Llama3 (meta-llama-3-70b-instruct[14], maximum context length 8096 tokens) under the Meta Llama 3 community license. To ensure robust generation of candidate keyphrases, the 'Top K' number of candidate keyphrases was set at 50 for

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

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

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

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

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

[7]https://github.com/mangalm96/keygames-pke

[8]https://github.com/xnliang98/uke_ccrank

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

[10]https://github.com/linhanz/mderank

[11]https://github.com/NKU-HLT/PromptRank

[12]https://platform.openai.com/docs/models/gpt-3-5-turbo

[13]https://platform.openai.com/docs/models/gpt-4o

[14]https://replicate.com/meta/meta-llama-3-70b-instruct

longer document datasets. For shorter document datasets, the model is configured to extract the top 20 keyphrases. For LLMs, the parameter for consistency $temperature$ is set to be 0, and max number of tokens max_tokens is set to be 576. All other LLM parameters are kept at their default values.

The window size of co-occurrence (w) and the weight for the linear integration factor (d) for each dataset are shown in the Table 3. It is worth noting that these parameters are used for all three LLMs evaluated in this research to demonstrate the generalizability of the proposed model. The window size is not applicable to Inspec. Our experiments show that augmenting LLM with the graph-based model does not improve the overall performance.

Table 3: A Summary of Parameters

Data Set	Inspec	SemEval2017	SemEval2010	Nguyen2007
Top K	20	20	50	50
w (in Eq. (1))	-	5	30	13
d (in Eq. (3))	1.0	0.9	0.5	0.5

3.4 Results

Table 4 presents a comparison of the LLMRank model's performance with UKE baselines across four benchmark datasets. The performance of the baseline models is reported based on results published in the original publications or from recent papers that cited those publications. For datasets that were not evaluated using some of the baselines, we ran the published code to obtain the results for comparison. The results generated from running the published code are marked with an asterisk (*).

The table presents the performance of keyphrase extraction using three state-of-the-art LLMs alone and LLMRank models using "original text only" and "generative text allowed" settings. We highlight the best results each dataset under F1@5, 10 and 15, respectively. For the long document datasets (SemEval2010 and Nguyen2007), the state-of-the-art LLMs alone outperformed the baseline models in the literature, particularly in F1@5. This indicates that LLMs are superior at capturing the semantic richness of long texts, which provide ample information for analysis and extraction. For the short document datasets (Inspec and SemEval2017), the state-of-the-art LLMs achieved performance comparable to or slightly below the best-performing baselines. LLMs consistently show strong performance in identifying the top 5 keyphrases (reflected by the values of F1@5), indicating their effectiveness in identifying a few, but crucial, phrases in the input text. Comparing the three state-of-the-art LLMs alone models, GPT models are better at analyzing the short text documents, whereas Llama 3 is better at analyzing the long text documents.

Although LLMs alone shows superior performance on long document datasets, the LLMRank that augments the LLMs using the graph-based model enhance the LLMs performance further. The LLMRank achieves the highest F1 scores across all metrics on the SemEval2010 and Nguyen2007 datasets and improved the performances of the corresponding LLMs alone approach. This underscores the efficacy of capturing the sentential relations using graph-based approach for long texts. However, the results also show that the augmenting the LLMs using graph-based approach does not have a positive impact on the performances gained from the shorter documents. Through experiments, we found that the graph-based approach can even lower the performance of the LLMs on the Inspec dataset. Whereas for SemEval2017 dataset, the positive impact of the graph-based augmenting is very minimal or none. In general, the state-of-the-art LLMs show no significant advantage over other deep learning models in generating candidate keyphrases for short documents.

Our results also highlight the impact of using the "original text only" versus "generative text allowed" settings. For instance, in the Inspec dataset, all three LLMs and our LLMRank models perform better in the "generative text allowed" setting. However, this improvement is not observed for the long-text document dataset Nguyen2007. Results on the SemEval datasets are mixed, with different LLMs showing varying performances. We believe this variability may be related to the training data of these LLMs and their sensitivity to the instructions in the prompts. Since LLMs are highly sensitive to the instructions in the prompt, some prompts may be particularly effective. Our approach includes an intuitive evaluation of keyphrase extraction performance by setting the temperature parameter to zero to minimize result variations. Achieving a good reproduciblility of LLM outputs remains challenging.

Overall, our results emphasize the potential of enhancing keyphrase extraction methods with LLMs and graph-based techniques, especially for challenging datasets with long or complex texts.

Table 4: Comparative F1 Scores at 5, 10, 15 Across Keyphrase Extraction Models

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 [15]	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! [8]	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 [19]	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]	27.79	34.46	36.05	18.23	27.73	31.73	5.90	9.02	10.58	1.86⋆	3.55⋆	4.56⋆
PositionRank [13]	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 [4]	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 [2]	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 [26]	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 [22]	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 [18]	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 [10]	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) [33]	26.17	33.81	36.17	22.81	32.51	37.18	12.95	17.07	20.09	14.47⋆	17.45⋆	17.44⋆
HyperRank [25]	33.35	**40.79**	**42.12**	-	-	-	14.79	21.33	24.20	-	-	-
PromptRank(T5) [17]	31.73	37.88	38.17	**27.14**	**37.76**	**41.57**	17.24	20.66	21.35	18.07⋆	21.14⋆	21.48⋆
Models in this Study (Original Text Only)												
GPT-3.5	34.61	39.40	38.22	24.65	33.16	36.15	19.44	21.65	21.49	24.29	24.74	22.84
GPT-4o	34.26	38.16	37.86	23.29	33.01	37.26	20.40	23.49	24.25	23.32	22.22	20.53
Llama3	25.09	28.09	26.70	18.22	24.02	25.52	21.94	24.64	23.27	29.06	28.43	25.57
LLMRank (GPT-3.5)	34.61	39.40	38.22	24.73	33.17	36.10	19.66	22.77	22.17	25.00	26.08	23.47
LLMRank (GPT-4o)	34.26	38.16	37.86	23.34	33.03	37.25	21.88	25.23	25.51	24.89	24.68	22.01
LLMRank (Llama3)	25.09	28.09	26.70	18.19	24.01	25.55	**22.22**	24.87	23.76	27.86	28.42	**26.63**
Models in this Study (Generative Text Allowed)												
GPT-3.5	**35.93**	39.90	38.52	24.35	32.67	35.93	18.56	21.61	21.20	23.42	23.49	21.76
GPT-4o	35.86	39.72	38.57	23.28	32.62	36.64	20.55	23.72	24.39	22.10	22.03	20.36
Llama3	25.73	29.60	28.75	18.23	24.50	26.12	21.37	25.02	23.98	28.84	28.69	25.77
LLMRank (GPT- 3.5)	35.93	39.90	38.52	24.48	32.66	35.88	18.66	22.26	22.04	24.43	25.21	22.66
LLMRank (GPT-4o)	35.86	39.72	38.57	23.38	32.63	36.71	21.86	**26.31**	**26.03**	24.13	23.39	21.15
LLMRank (Llama3)	25.73	29.60	28.75	18.26	24.50	26.10	22.03	25.19	24.65	**29.56**	**29.69**	26.31

The superior performance of the proposed LLMRank validates our novel augmentation of LLMs, effectively addressing the inherent limitations of existing approaches.

3.4.1 Computational cost

The experiments were performed on a machine with an i7 9700k processor and 48GB of RAM. For all three LLMs, the average processing time is under 0.054 seconds for both short and long documents. The proposed LLMRank model adds less than 0.015 seconds for short documents and less than 0.034 seconds for long documents to the base LLMs' processing time.

3.5 Ablation Study

3.5.1 Effects of Window Size

The proposed LLMRank model utilizes the structure of the article, employing a graph model to improve the ranking of keyphrases generated by LLMs. This section details ablation studies conducted on the co-occurrence window size between the candidate keyphrases using LLMRank built on GPT-4o in the "Generative Text Allowed" setting. Since short text document sets have no more than seven sentences, this ablation study is only applied to the long text document sets.

Given that a long text document can contain hundreds of sentences, we experimented with the window size from 2 to 20 with the incremental step of 1, then from 25 up to 40 with an incremental step of 5. The performance across co-occurrence windows ranging from 2 to 40 is shown in Figure 3. For the SemEval2010 dataset, the peak F1@5 score occurs at a window size of 4, with performance stabilizing after a window size of 12. The F1@10 scores reach their peak at a window size of 30, while F1@15 scores peak at 25, maintaining high levels thereafter. For the Nguyen2007 dataset, F1@5 remains high between window sizes of 9 to 14 and 20 to 25; F1@10 is sustained between 9 and 15 and after 25; and F1@15 is consistently high between 13 and 18.

Although optimal performance for both datasets might be achieved with larger co-occurrence windows, considering computational costs, a window size below 10 can still yield comparable performance.

3.5.2 Effects of Integration Ratio

Augmenting the LLMs is significantly influenced by the integration ratio d defined in Equation 3. To explore the effects of this ratio, an ablation study on the proposed LLMRank model built on GPT-4o in the "Generative Text Allowed" setting was conducted. Figure 4 illustrates how we systematically varied d from 0 to 1 in increments of 0.1.

For the Inspec dataset, the F1 scores incremen-

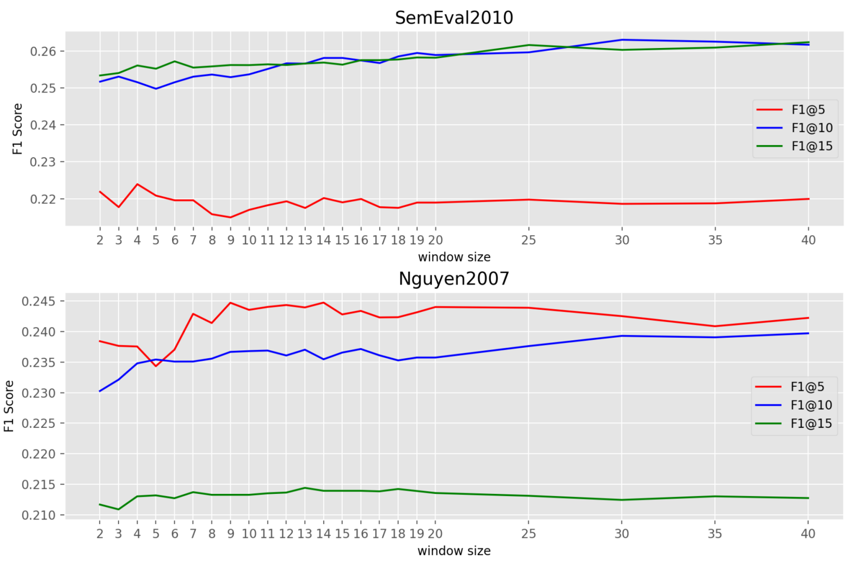


Figure 3: Evaluation of the Co-occurrence Window Size Impact on Performance

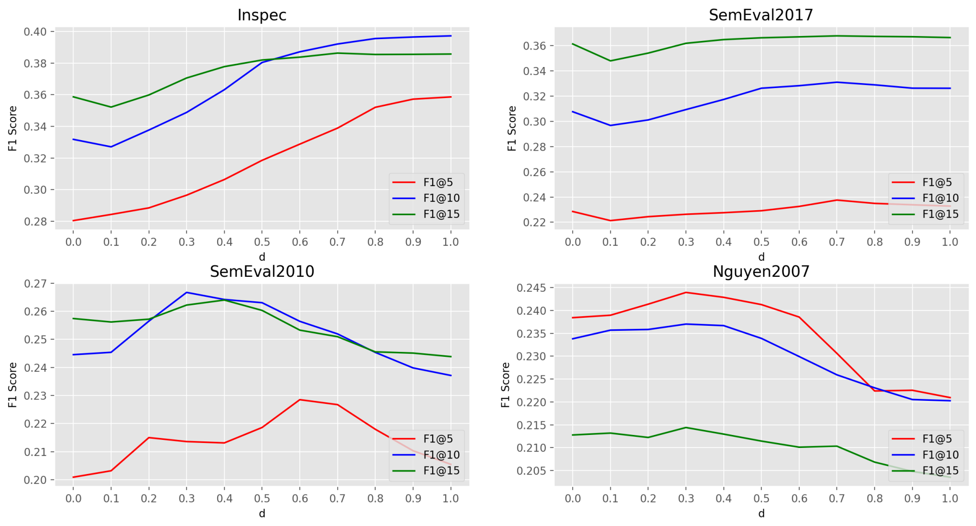


Figure 4: Evaluation of the Integration Ratio Impact on Performance

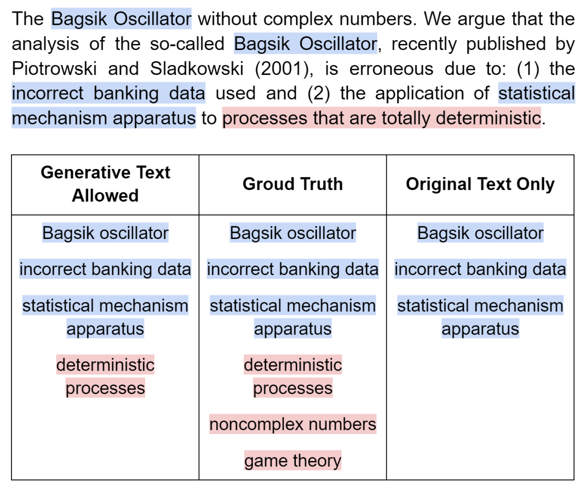
tally improve as d increases, with the highest scores observed at $d = 1.0$ across F1@5, 10, and 15. This implies that the model relies entirely on the ranking power of GPT-4o, without incorporating the graph model's ranking. Similarly, the SemEval2017 dataset experiences peak performances at high d values, specifically $d = 0.7$ for F1@5, 10 and 15. This pattern suggests that LLM alone can perform well on short document datasets.

In the case of the SemEval2010 dataset, which contains long text documents, the optimal performance occurs at a medium integration ratio; a peak at $d = 0.6$ for F1@5 is noted, with F1@5 and F1@10 peaking at $d = 0.3$ and $d = 0.4$, respectively. Thus, we adopt the integration ratio of 0.5, which is generalizable towards the calculation of F1@5, 10, and 15. For the Nguyen2007 dataset, the optimal integration ratio is identified at $d = 0.3$ across F1@5, 10, and 15. The LLMRank model achieves optimal performance when d is set to be 0.3, which means GPT-4o contributes 30% and the PageRank model provides 70% to the final ranking. This indicates that GPT-4o might not fully comprehend the overall context of these long documents. The LLMRank model is capable to addressing this limitation and effectively identify keyphrases.

In summary, the optimal d values diverge across different datasets, suggesting a nuanced dependency on the specific characteristics of each dataset.

3.6 Case Study

In most scenarios, the system performs better in the "Generative Text Allowed" setting. In this sec-



The Bagsik Oscillator without complex numbers. We argue that the analysis of the so-called Bagsik Oscillator, recently published by Piotrowski and Sladkowski (2001), is erroneous due to: (1) the incorrect banking data used and (2) the application of statistical mechanism apparatus to processes that are totally deterministic.

Generative Text Allowed	Groud Truth	Original Text Only
Bagsik oscillator	Bagsik oscillator	Bagsik oscillator
incorrect banking data	incorrect banking data	incorrect banking data
statistical mechanism apparatus	statistical mechanism apparatus	statistical mechanism apparatus
deterministic processes	deterministic processes	
	noncomplex numbers	
	game theory	

Figure 5: Comparison of keyphrase extraction capabilities between "Original Text Only" and "Generative Text Allowed" settings

tion, we use a case study to investigate the possible reasons.

On the Inspec dataset, GPT-3.5 in "Generative Text Allowed" setting performs significantly better than in "Original Text Only" setting. Figure 5 shows a document from the Inspec dataset, the text contains six ground-truth keyphrases; three are embedded in the original text (highlighted in blue), and three are not (highlighted in pink). The system can successfully identify the three keyphrases embedded in the text in the "Original Text Only" setting. However, the ones that do not occurred in the original text might need either external knowledge or summarization to derive. For example, 'game theory' might not be derived without correct instruction and relevant domain knowledge. Other two keyphrases 'noncomplex numbers' and 'deterministic processes' could be potentially be derived from the text if proper instruction is given in the prompt. Even with current instruction in the prompt, the "Generative Text Allowed" setting enables GPT-3.5 to identify 'deterministic processes'. This implies that LLMs have the potential to generate keyphrases based on relevant domain knowledge and proper instruction.

4 Related Works

keyphrase Extraction (KPE)techniques are categorized into supervised and unsupervised methods. Supervised Keyphrase Extraction: Supervised KPE utilizes machine learning to treat extraction as a binary classification issue, employing models that learn from extensively labeled datasets to recognize patterns and relationships. Unsupervised Keyphrase Extraction utilizes three main approaches: statistical, graph-based, and hybrid methods. Statistical models, like Term Frequency-Inverse Document Frequency (TF-IDF) [15], analyze contextual data to highlight distinctive terms and assess keyphrase candidates based on word frequency and text positioning [21, 7].

Graph-based models represent keyphrases as nodes, with edges reflecting semantic relationships. Algorithms like PageRank score these nodes, with enhancements from clustering to form document-specific graphs or integrate word embeddings for more accurate rankings [19, 27, 6, 29, 12, 5, 32].

Hybrid methods merge deep learning precision with traditional techniques, leveraging pre-trained models for enhanced context understanding. These methods employ embedding similarities and document attributes to effectively rank keyphrases, incorporating advanced algorithmic strategies like attention mapping and evolutionary game theory models [3, 26, 22, 10, 18, 33, 11, 25].

Generative models such as GPT and BERT have transformed KPE, enabling more nuanced analysis and ranking of candidates. These models use prompt-based learning and integrate knowledge graphs to improve extraction accuracy [17, 24, 23].

5 Conclusion

In conclusion, our research introduces an innovative method, LLMRank, to augment the state-of-the-art LLMs for unsupervised keyphrase extraction using a graph-based approach. Proposed model utilizes the generative LLMs to identify a set of candidate keyphrases. Subsequently, these candidates are subjected to a re-ranking process through a graph-based model that leverages structured contextual information on a global scale within a given text document. Our approach not only capitalizes on the inherent strengths of generative LLMs but also effectively mitigates their limitations by incorporating a sophisticated analysis of phrase co-occurrence relationships.

6 Limitations

The limitations of this research include: (1) The proposed model only works well on long text documents, which limits the generalization of the proposed approach. (2) Although we experimented with three state-of-the-art LLMs for unsupervised keyphrase extraction, other LLMs such as PaLM

2 can also be explored. (3) Only four benchmark datasets for keyphrase extraction are included in this study. More can be included in the future research; (4) This research focused on unsupervised keyphrase extraction, the capability of LLMs towards supervised keyphrase extraction can be further explored.

7 Ethical Impact

Generally speaking, this work does not involve ethical issues. However, if the proposed method is used to analyze data involving privacy and biases concerns, ethical impacts should be addressed before using generative AI.

References

[1] 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.

[2] Kamil Bennani-Smires, Claudiu Musat, Andreaa Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. *CoNLL 2018*, page 221.

[3] 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.

[4] Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

[5] 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.

[6] Adrien Bougouin, Florian Boudin, and B'eatrice 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.

[7] Ricardo Campos, V'ıtor Mangaravite, Arian Pasquali, Al'ıpio Jorge, C'elia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

[8] Ricardo Campos, V'ıtor Mangaravite, Arian Pasquali, Al'ıpio M'ario Jorge, C'elia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer.

[9] Tong Chen, Xuewei Wang, Tianwei Yue, Xiaoyu Bai, Cindy X Le, and Wenping Wang. 2023. Enhancing abstractive summarization with extracted knowledge graphs and multi-source transformers. *Applied Sciences*, 13(13):7753.

[10] 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.

[11] Haoran Ding and Xiao Luo. 2022. Agrank: Augmented graph-based unsupervised keyphrase extraction. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 230–239.

[12] Corina Florescu and Cornelia Caragea. 2017. A position-biased pagerank algorithm for keyphrase extraction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

[13] Corina Florescu and Cornelia Caragea. 2017. 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.

[14] 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.

[15] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

[16] 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.

[17] Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, and Xiaoyan Bai. 2023. Promptrank: Unsupervised keyphrase extraction using prompt. *arXiv preprint arXiv:2305.04490*.

[18] 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*.

[19] 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.

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

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

[22] 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.

[23] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*.

[24] Mingyang Song, Haiyun Jiang, Shuming Shi, Songfang Yao, Shilong Lu, Yi Feng, Huafeng Liu, and Liping Jing. 2023. Is chatgpt a good keyphrase generator? a preliminary study. *arXiv preprint arXiv:2303.13001*.

[25] Mingyang Song, Huafeng Liu, and Liping Jing. 2023. Hyperrank: Hyperbolic ranking model for unsupervised keyphrase extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16070–16080.

[26] 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.

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

[28] Fali Wang, Runxue Bao, Suhang Wang, Wenchao Yu, Yanchi Liu, Wei Cheng, and Haifeng Chen. 2024. Infuserki: Enhancing large language models with knowledge graphs via infuser-guided knowledge integration. *arXiv preprint arXiv:2402.11441*.

[29] 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.

[30] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2024. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36.

[31] 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.

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

[33] 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*.

[34] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.