Attention-Seeker: Dynamic Self-Attention Scoring for Unsupervised Keyphrase Extraction

Erwin D. López Z.[1], Cheng Tang[2], Atsushi Shimada[2],
Graduate School of Information Science and Electrical Engineering, Kyushu University,
[1]edlopez96s6@gmail.com
[2]{tang,atsushi}@ait.kyushu-u.ac.jp

Abstract

This paper proposes Attention-Seeker, an unsupervised keyphrase extraction method that leverages self-attention maps from a Large Language Model to estimate the importance of candidate phrases. Our approach identifies specific components – such as layers, heads, and attention vectors – where the model pays significant attention to the key topics of the text. The attention weights provided by these components are then used to score the candidate phrases. Unlike previous models that require manual tuning of parameters (e.g., selection of heads, prompts, hyperparameters), Attention-Seeker dynamically adapts to the input text without any manual adjustments, enhancing its practical applicability. We evaluate Attention-Seeker on four publicly available datasets: Inspec, SemEval2010, SemEval2017, and Krapivin. Our results demonstrate that, even without parameter tuning, Attention-Seeker outperforms most baseline models, achieving state-of-the-art performance on three out of four datasets, particularly excelling in extracting keyphrases from long documents. The source code of Attention-Seeker is available at `https://github.com/EruM16/Attention-Seeker`.

1 Introduction

Keyphrase extraction is a critical task in Natural Language Processing (NLP) where the core content of documents is summarized into a set of words or phrases. This process facilitates efficient and accurate information retrieval, which is valuable for several NLP applications, including Retrieval-Augmented Generation (RAG), document categorization, text segmentation, and topic modeling. Keyphrase extraction methods are generally classified into two categories: supervised and unsupervised. Supervised methods achieve high performance by harnessing the power of deep learning techniques, such as neural networks (Song et al., 2021), LSTM (Sahrawat et al., 2020), GNN (Sun et al., 2019), and transformers (Martinc et al., 2022). However, these methods require large amounts of labeled data and are often domain-specific, limiting their practical applicability. In contrast, unsupervised methods rely solely on information extracted from the document itself, making them adaptable across various domains.

Unsupervised keyphrase extraction methods can be divided into five categories based on the type of information they extract. The first two categories include traditional approaches: statistical methods (Spärck Jones, 2004; Campos et al., 2020), which use in-corpus statistical information, and graph-based methods (Mihalcea and Tarau, 2004; Wan and Xiao, 2008; Bougouin et al., 2013; Florescu and Caragea, 2017), which leverage words co-occurrence patterns. The remaining three categories consist of more recent works that use Pretrained Language Models (PLMs) to extract semantic information. Embedding-based methods (Bennani-Smires et al., 2018; Sun et al., 2020; Ding and Luo, 2021; Zhang et al., 2022; Liang et al., 2021) analyze similarities between documents and phrases within the PLM's embedding space. Prompt-based methods (Kong et al., 2023) use the PLM's decoder logits to estimate the probability of generating a candidate phrase when a prompt is included in the document (e.g., this book talks about [candidate]). Finally, self-attention-based methods (Kang and Shin, 2023) examine the PLMs' Self-Attention Map (SAM) to identify which candidate phrases receive the most attention across all document tokens.

As noted by Kong et al. (2023); Kang and Shin (2023), embedding-based approaches struggle to accurately estimate document similarities due to the anisotropic nature of PLM embedding spaces. Prompt and self-attention-based methods circumvent these limitations and achieve state-of-the-art (SOTA) results but introduce new complexities. For example, PromptRank (Kong et al., 2023), a prompt-based method, requires tuning two hyperparameters (alpha and beta) and exploring a wide range of potential prompts. Similarly, SAMRank (Kang and Shin, 2023), a self-attention-based model, requires selecting an optimal SAM from potentially thousands of layer and head combinations. Since documents typically lack labels to guide parameter selection, achieving optimal results without parameterization remains a significant challenge.

In this context, we propose a method that automatically adapts to the characteristics of the input text, eliminating the need for manual parameter tuning. Our approach extends SAMRank (Kang and Shin, 2023) by introducing a module that selects the most relevant SAMs to effectively score candidate phrases. The relevance of SAMs is estimated by defining the characteristics of an optimal SAM. Specifically, Attention-Seeker employs a binary vector that assigns ones to candidate phrases and zeros to all other tokens. It then computes the similarity between this vector and the attention vectors (rows) of the SAMs, using the average similarity score to determine each SAM's relevance. Furthermore, Attention-Seeker evaluates the relevance of different parts of the text: for short documents, it scores individual attention vectors, whereas for long documents, it scores document segments.

To the best of our knowledge, Attention-Seeker is the first method to automatically adapt its keyphrase extraction process to the specific characteristics of the input text. This novel approach allows Attention-Seeker to achieve SOTA-level performance on four benchmark datasets without requiring parameter tuning. Our ablation study shows that each of the Attention-Seeker's scoring modules positively contributes to the final performance. While our proposed relevance scoring mechanism is simple, its effectiveness highlights the potential for further exploration in this direction.

The main contributions of this paper are summarized as follows:

- We propose Attention-Seeker, a novel unsupervised keyphrase extraction method that automates parameter tuning of self-attention-based models.

- We propose a simple yet effective approach for selecting the most relevant attention vectors, self-attention maps, and document segments for keyphrase extraction.

- We demonstrate that non-parametric self-attention-based methods using LLAMA 3-8B achieve high performance on four benchmark datasets, with notable efficacy on long documents.

2 Related work

2.1 Unsupervised Keyphrase Extraction

Traditional unsupervised keyphrase extraction methods can be broadly divided into statistic-based and graph-based approaches. Statistic-based methods estimate word importance using statistical metrics derived from the text. For instance, the TF-IDF method (Spärck Jones, 2004) relies on word frequencies, while YAKE (Campos et al., 2020) incorporates additional factors such as word co-occurrences, positions, and casings. In contrast, graph-based methods represent documents as graphs, with words as nodes, and use graph algorithms to estimate word importance. TextRank (Mihalcea and Tarau, 2004) pioneered this approach by adapting the PageRank algorithm (Page et al., 1998) for keyphrase extraction. Subsequent work has extended this framework: SingleRank (Wan and Xiao, 2008) incorporates information from neighboring documents, TopicRank (Bougouin et al., 2013) clusters nodes within a topic space, and PositionRank (Florescu and Caragea, 2017) integrates a position bias into the PageRank algorithm.

Recent advances use Pretrained Language Models (PLMs) to extract semantic features from documents. These approaches can be categorized into embedding-based, prompt-based, and self-attention-based methods. Embedding-based approaches are the most common, focusing on generating embedding vectors from documents and candidate keyphrases. For example, EmbedRank (Bennani-Smires et al., 2018) uses Doc2Vec (Mikolov et al., 2013) and Sent2Vec (Pagliardini et al., 2018) to derive document and phrase embeddings, using cosine similarity for ranking. SIFRank (Sun et al., 2020) extends this approach by integrating ELMo (Peters et al., 2018) embeddings and the SIF model (Arora et al., 2017) to refine document embeddings. JointGL (Liang et al., 2021) applies a similar strategy using BERT (Devlin et al., 2019) embeddings while incorporating local similarities derived from document graphs. AttentionRank (Ding and Luo, 2021) introduces cross-attention with BERT embeddings to generate document vectors, using self-attention maps (SAMs) for further refinement. Finally, MDERank (Zhang et al., 2022) takes an innovative approach by masking candidate tokens and comparing their similarity to the original document.

Prompt-based methods take a different approach, using PLMs to predict keyphrases directly through prompt conditioning. For example, PromptRank (Kong et al., 2023) estimates the likelihood of candidate phrases by using the logits generated by the T5 (Raffel et al., 2020) PLM, which is specifically prompted for keyphrase extraction. In contrast, self-attention-based methods, such as SAMRank (Kang and Shin, 2023), focus on using SAMs from PLMs. SAMRank aggregates attention vectors from a specific SAM to measure the attention a candidate phrase receives from other tokens. While SAMRank demonstrates the usefulness of SAMs in keyphrase extraction, it highlights the challenge of effectively selecting the most relevant SAMs without label information.

2.2 Self-Attention Map

Vaswani et al. (2017) introduced the "Scaled Dot-Product Attention" mechanism, as shown in Equation 1. Here, the Query (Q), Key (K), and Value (V) are linear transformations of the embedding representation of the same input sentence. The Softmax function assigns an attention score to the tokens in the Key based on their relevance to a token in the Query, resulting in a matrix known as the Self-Attention Map (SAM). This SAM is used to compute a weighted sum of the Value vectors, which is then transformed into a new embedding representation of the input sentence for further processing by the same attention mechanism in subsequent layers.

$$Att\left(Q,K,V\right) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

In the transformer model proposed by Vaswani et al. (2017), and in subsequent PLMs, each layer contains multiple heads. As a result, each SAM can focus on different aspects of text processing (multiple heads) at different syntactic levels (multiple layers). Clark et al. (2019) examined the characteristics of different SAMs in the BERT model (Devlin et al., 2019) and identified several types: SAMs that specialize in attending to previous or next tokens, SAMs that focus on separator tokens (e.g., special tokens, periods, and commas), and SAMs that attend to a large number of words (uniform attention). Furthermore, they identified specific SAMs that preserve syntactic information, suggesting that some SAMs might specialize in attending to keyphrases within the text. Supporting this hypothesis, Kang and Shin (2023) demonstrated that attention scores from a manually selected SAM can be leveraged to achieve SOTA performance in keyphrase extraction tasks. However, identifying and selecting SAMs specialized in attending to keyphrases remains an open challenge in this approach.

3 Methodology

In this section, we introduce our proposed method, Attention-Seeker. It consists of four main steps: (1) Generation of candidate phrases using Part-of-Speech (POS) sequences. (2) Extraction of Self-Attention Maps (SAMs). (3) Estimation of a vector of attention scores from the most relevant SAMs. (4) Scoring the importance of each candidate phrase based on these attention scores. The main contribution of our method lies in step 3, where we propose a novel approach to estimate the vector of attention scores by identifying and exploiting information from the most relevant SAMs.

3.1 Candidate Generation

We follow a well-established approach for generating candidate phrases (Kang and Shin, 2023; Ding and Luo, 2021; Bennani-Smires et al., 2018). First, we tokenize and POS tag all words in the document by using the Stanford CoreNLP tool[1]. Next, we extract noun phrases (tagged as “NN”, “NNS”, “NNP”, “NNPS”, and “JJ”) using the NLTK’s RegexpParser[2]. These phrases are then defined as the document’s keyphrase candidates.

3.2 Extraction of the Self-Attention Maps

We extract the SAMs from each layer and head of the Large Language Model LLAMA 3-8B (AI@Meta, 2024) using the Huggingface library[3]. Similar to SAMRank (Kang and Shin, 2023), we handle the extraction of SAMs differently depending on the document length. For short documents, we input the entire text directly into the LLM. For long documents, we first input the document’s abstract into the LLM, then divide the remaining text

[1]https://stanfordnlp.github.io/CoreNLP/
[2]https://github.com/nltk
[3]https://github.com/huggingface/transformers

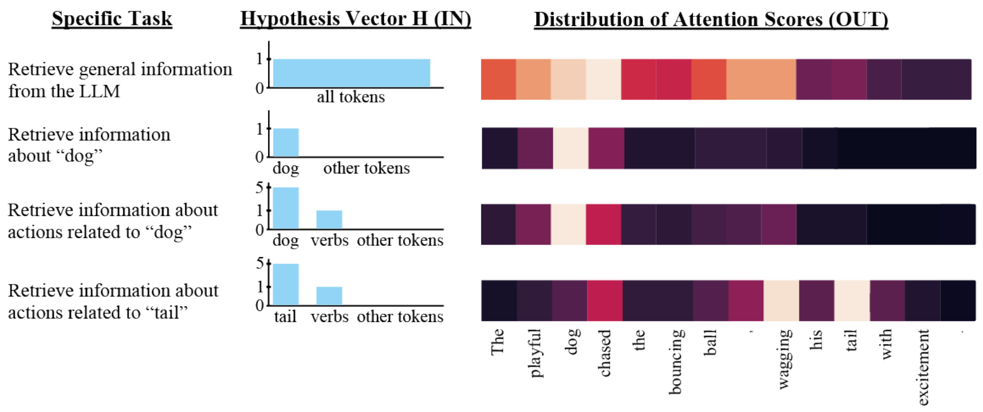


Figure 1: The effect of different hypothesis vectors H in an LLM's distribution of attention scores.

into equally sized segments (same number of tokens) and process each segment independently with the LLM.

3.3 SAMs' Relevance Scoring: Hypothesis Engineering

All Self-Attention Maps (SAMs) in a Large Language Model (LLM) capture distinct attention patterns over tokens, and their distribution of attention scores serves as a unique fingerprint. We propose to use these fingerprints to measure the relevance of each SAM within the LLM. Given a SAM of size $n \times n$, our method requires the design of a hypothesis vector H of length n that encapsulates the desired properties of the attention distribution for a given task. Then, we calculate a similarity score between the SAM and the vector H that quantifies the relevance of the SAM to the given task. We use this score as a weight to combine the attention distributions of all SAMs into a single output distribution for the LLM. This averaged distribution aggregates information across the different contexts captured by all layers and heads, prioritizing the contributions of the most relevant SAMs.

Figure 1 illustrates the effect of a hypothesis vector H on the output distribution of attention scores for the same input text. The first case represents the default output distribution of the LLM, where all SAMs are considered equally relevant. In this scenario, H is defined as a uniform vector with all elements set to 1, resulting in equal similarity scores for all SAMs. In the second case, H is defined as a binary vector where only the elements corresponding to "dog" are set to 1. This configuration assigns greater relevance to SAMs that focus on syntactic relationships involving the noun "dog" and facilitates the retrieval of information about "dog". A simple post-processing step, which masks the attention scores of "dog" from the final distribution, allows the extraction of associated words such as "playful" and "chased".

The third and fourth cases demonstrate the use of combined hypothesis vectors. Both cases include a hypothesis vector designed to prioritize SAMs that capture syntactic information related to verbs (assigning 1 to tokens corresponding to verbs and 0 to others). In the third case, this verb-focused vector is combined with the hypothesis vector for "dog", while in the fourth case, it is combined with the vector for "tail". As shown in Figure 1, the resulting distributions differ significantly, highlighting the verb most related to "dog" in the third case and the verb most related to "tail" in the fourth case. These examples illustrate how task-specific hypothesis vectors can guide the extraction of syntactically relevant information.

Although the examples shown in Figure 1 use simple hypothesis vectors, future works could explore advanced methodologies to design optimal vectors tailored to specific tasks. In this study, we propose two simple designs for keyphrase extraction: one for short and another for long documents. Both designs start from a vector H that focuses on the keyphrase candidates. This approach prioritizes SAMs that model the inter-relationships of these candidates. Among the relevant SAMs, we can include those capturing long-term dependencies, object-direct relationships, coreferences, and keyphrases.

3.4 Attention Scores Estimation: Short Documents

Our proposed method for short documents is summarized in Figure 2. Here, the "Attention Score

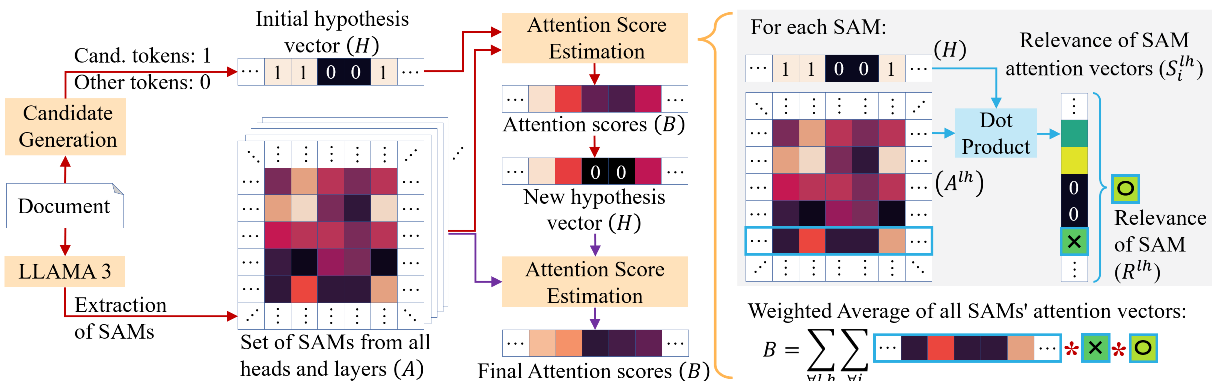

Cand. tokens: 1
Other tokens: 0
Candidate Generation
Document
LLAMA 3
Extraction of SAMs
Initial hypothesis vector (H)
Set of SAMs from all heads and layers (A)
Attention Score Estimation
Attention scores (B)
New hypothesis vector (H)
Attention Score Estimation
Final Attention scores (B)
For each SAM:
(H)
Relevance of SAM attention vectors (S_i^{lh})
Dot Product
(A^{lh})
Relevance of SAM (R^{lh})
Weighted Average of all SAMs' attention vectors:

Figure 2: The core architecture of Attention-Seeker for short documents.

Estimation" module first estimates the relevance of SAMs (R^{lh}) and their internal attention vectors (S_i^{lh}). These relevance values are then used to compute a weighted average of the attention vectors across all SAMs. Accordingly, the final vector of attention scores B is determined from the most relevant parts of the most relevant SAMs.

We define the hypothesis vector H as in Equation 2. This vector prioritizes SAMs that pay attention to potential keyphrases (candidates), while de-emphasizing those that primarily attend to token positions or separator tokens (Clark et al., 2019).

$$H_i = \begin{cases} 1, & \text{token } i \in \text{candidate} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Given a SAM A^{lh}, where l is the current layer and h is the current head, the relevance of each attention vector S_i^{lh} is calculated as the dot product between H and A_i^{lh}. This calculation is formally expressed as the matrix multiplication shown in Equation 3, where S^{lh} is the vector of S_i^{lh} values.

$$S^{lh} = A^{lh} \cdot H \tag{3}$$

We further set S_i^{lh} to 0 for tokens i that do not belong to candidate phrases (post-processing masking). This is an empirical decision, and can be justified by the existence of uniformly distributed attention vectors (e.g., periods preserving contextual information for subsequent sentences) or randomly distributed attention vectors for "non-op" contexts (Clark et al., 2019).

As shown by Equation 4, the relevance of each SAM R^{lh} is calculated as the average relevance score of its individual attention vectors.

$$R^{lh} = \frac{1}{n}\sum_{i=0}^{n} S_i^{lh} \tag{4}$$

The final attention vector B is computed as a weighted average of all attention vectors across all SAMs, with the weights determined by their respective relevance scores, as shown in Equation 5.

$$B = \sum_{\forall l,h}\sum_{\forall i} A_i^{lh} * S_i^{lh} * R^{lh} \tag{5}$$

As illustrated in Figure 2, the resulting vector B is converted into a new hypothesis vector H through post-processing masking, where B_i values corresponding to non-candidate tokens are set to 0. Similar to the initial binary vector H, this updated vector prioritizes SAMs that focus on possible keyphrases. However, it further emphasizes SAMs that allocate more attention to the most important candidates, thereby favoring SAMs likely to specialize in attending to keyphrases.

3.5 Attention Scores Estimation: Long Documents

Long documents are segmented into an abstract and equally sized segments, as described in Section 3.2. These segments are processed independently as short documents (Section 3.4) excluding the relevance scores for attention vectors (S_i^{lh}) from the final calculation (Figure 3). This simplification avoids the complexities of applying context-dependent relevance scores across diverse segments, which could reduce effectiveness without a robust normalization strategy.

For each segment s, the relevance of each SAM (R^{lh}) is defined as the average of the relevance scores of its corresponding attention vectors (Equation 6).

$$R^{lh} = \frac{1}{n}\sum_{i=0}^{n} A_i^{lh} \cdot H \tag{6}$$

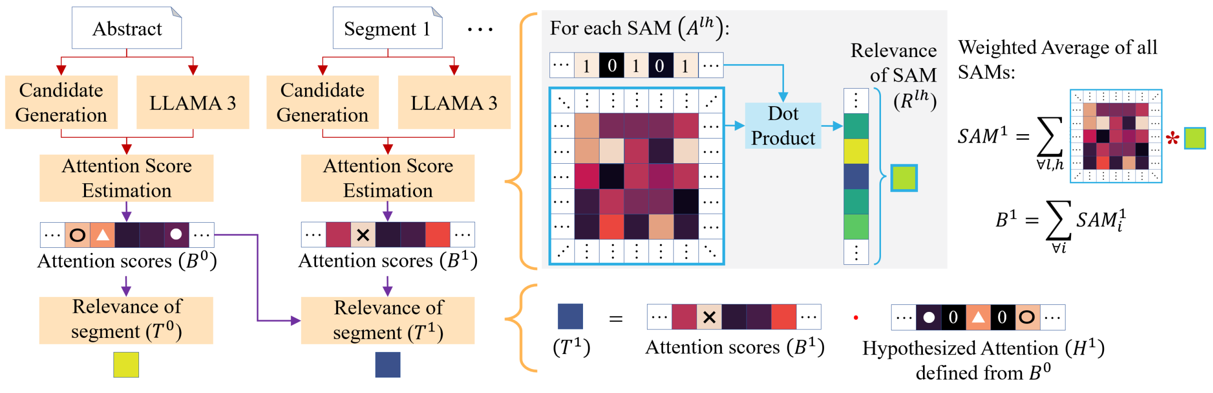

Abstract
Segment 1
Candidate Generation
LLAMA 3
Attention Score Estimation
Attention scores (B^0)
Attention scores (B^1)
Relevance of segment (T^0)
Relevance of segment (T^1)
For each SAM (A^{lh}):
Dot Product
Relevance of SAM (R^{lh})
Weighted Average of all SAMs:
$SAM^1 = \sum_{\forall l,h}$
$B^1 = \sum_{\forall i} SAM_i^1$
(T^1)
Hypothesized Attention (H^1) defined from B^0

Figure 3: The core architecture of Attention-Seeker for long documents.

We then apply l_1-normalization to this vector (Equation 7) to compute a weighted average SAM^s of all SAMs from the segment s (Equation 8). This normalization ensures that SAM^s preserves the standard properties of a SAM (e.g., the attention values in each row sum to 1).

$$R_n^{lh} = \frac{R^{lh}}{\sum_{\forall l,h} R^{lh}} \tag{7}$$

$$SAM^s = \sum_{\forall l,h} A^{lh} * R_n^{lh} \tag{8}$$

The attention score vector B^s for the segment s is obtained by summing all rows in the average SAM (Equation 9).

$$B^s = \sum_{\forall i} SAM_i^s \tag{9}$$

To compute the relevance of each segment (T^s), we define a new hypothesis vector H^s for each segment. This vector assigns the attention scores from the abstract (B^0) to tokens in the segment that are included in the candidate phrases of the abstract, while assigning 0 to all other tokens (Equation 10). This hypothesis vector prioritizes segments that focus more on phrases considered important by the abstract.

$$H_i^s = \begin{cases} B_j^0, & \text{token } i = \text{token } j \text{ of } B^0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The relevance score T^s for each segment s is finally calculated as the dot product between its attention score vector B^s and the hypothesis vector H^s (Equation 11).

$$T^s = B^s \cdot H^s \tag{11}$$

3.6 Candidate Final Score Calculation

In short documents, the final score of a candidate phrase is calculated by summing the attention scores B_i corresponding to the candidate tokens. Following Kang and Shin (2023), if a candidate phrase consists of a single word, its final score is divided by its frequency. For long documents, we apply the same method to each segment with the vector of attention scores $T^s * B^s$ (B^s multiplied by the relevance of the segment). The final score for each candidate is then obtained by summing its scores across all segments.

4 Experiments and Results

4.1 Datasets and Evaluation Metrics

We evaluate Attention-Seeker using four benchmark datasets: Inspec (Hulth, 2003), SemEval2010 (Kim et al., 2010), SemEval2017 (Augenstein et al., 2017), and Krapivin (Krapivin et al., 2009). These datasets consist of scientific papers and vary in length. Specifically, Inspec and SemEval2017 contain paper abstracts (short documents), while SemEval2010 and Krapivin contain full papers (long documents). Table 1 presents more detailed statistical information about each dataset.

The performance of keyphrase extraction is eval-

	Inspec	SE2017	SE2010	Krapivin
# Docs	500	493	100	460
Ave. Words	135	194	8154	8545
Ave. Sent.	6	7	380	312
Ave. Keys	9	17	15	6
Unigram (%)	13.5	25.7	20.5	17.8
Bigram (%)	52.7	34.4	53.6	62.2
Trigram (%)	24.9	17.5	18.9	16.4

Table 1: Statistics of datasets

Method	**Inspec**			**SemEval2017**			**SemEval2010**			**Krapivin**		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Statistic-based Methods												
TF-IDF	11.28	13.88	13.83	12.70	16.26	16.73	2.81	3.48	3.91	-	-	-
YAKE	18.08	19.62	20.11	11.84	18.14	20.55	11.76	14.40	15.19	8.09	9.35	11.05
Graph-based Methods												
TextRank	27.04	25.08	36.65	16.43	25.83	30.50	3.80	5.38	7.65	6.04	9.43	9.95
SingleRank	27.79	34.46	36.05	18.23	27.73	31.73	5.90	9.02	10.58	8.12	10.53	10.42
TopicRank	25.38	28.46	29.49	17.10	22.62	24.87	12.12	12.90	13.54	8.94	9.01	8.30
PositionRank	28.12	32.87	33.32	18.23	26.30	30.55	9.84	13.34	14.33	-	-	-
Embedding-based Methods												
EmbedRank d2v	31.51	37.94	37.96	20.21	29.59	33.94	3.02	5.08	7.23	4.05	6.60	7.84
EmbedRank s2v	29.88	37.09	38.40	-	-	-	5.40	8.91	10.06	8.44	10.47	10.71
SIFRank	29.11	38.80	**39.59**	22.59	32.85	38.10	-	-	-	1.62	2.52	3.00
AttentionRank	24.45	32.15	34.49	23.59	34.37	38.21	11.39	15.12	16.66	-	-	-
MDERank	27.85	34.36	36.40	20.37	31.21	36.63	13.05	18.27	20.35	11.78	12.93	12.58
JointGL	30.82	36.28	36.67	20.49	29.63	34.05	10.78	13.67	14.64	-	-	-
Prompt-based Methods												
PromptRank	31.73	37.88	38.17	**27.14**	**37.76**	**41.57**	17.24	20.66	21.35	16.11	16.71	16.02
Self-Attention Map-based Methods												
SAMRank	34.25	38.18	38.11	24.74	33.51	37.01	17.86	20.99	22.07	17.38	16.78	15.15
Attention-Seeker	**35.49**	**40.14**	39.22	25.40	34.53	38.50	**19.00**	**23.07**	**23.81**	**20.79**	**18.25**	**16.22**

Table 2: Performance of keyphrase extraction of baseline models and Attention-Seeker on four datasets. The **bold** indicates the best performance and the underline indicates the two best performances.

uated with the F1 score at the top 5 (F1@5), 10 (F1@10), and 15 (F1@15) keyphrases predicted by the model (candidates with the highest final scores). The final set of predicted keyphrases is determined after applying the NLTK's PorterStemmer for stemming and removing duplicate candidates.

4.2 Baselines

Our baselines include statistics-based methods: TF-IDF (Spärck Jones, 2004), YAKE (Campos et al., 2020); graph-based methods: TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013), PositionRank (Florescu and Caragea, 2017); embedding-based methods: EmbedRank (Bennani-Smires et al., 2018), SIFRank (Sun et al., 2020), AttentionRank (Ding and Luo, 2021), MDERank (Zhang et al., 2022), JointGL (Liang et al., 2021); prompt-based methods: PromptRank (Kong et al., 2023), and self-attention-based models: SAMRank (Kang and Shin, 2023). To ensure a fair comparison, we implemented SAMRank using the average of all SAMs (a non-parametric method) extracted with the LLAMA 3-8B model (BERT and GPT-2 results can be found on Appendix N). The optimized results of SAMRank are detailed in Appendix A. Additionally, we omitted the proportional score, as it led to lower performance in this setting (Appendix C). Consequently, the reported results of SAMRank represent the most important baseline for our model, as they correspond to the same method when all relevance scores are assigned equal values.

4.3 Results

Table 2 summarizes the performance of Attention-Seeker compared to baseline models across four benchmark datasets. The results show that Attention-Seeker achieves state-of-the-art (SOTA) performance on the Inspec, SemEval2010, and Krapivin datasets, while securing the second-best performance on the SemEval2017 dataset. The non-parametric SAMRank also performs well on all datasets, highlighting the effectiveness of Self-Attention Maps (SAMs) for unsupervised keyphrase extraction. In particular, the performance of self-attention-based methods on long document datasets (SemEval2017 and Krapivin) suggests a promising direction for extracting keyphrases from long documents using SAMs from LLMs. In this context, the consistent outperformance of Attention-Seeker over SAMRank demon-

Method	Inspec			SemEval2017			SemEval2010			Krapivin		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Base	34.25	38.18	38.11	24.74	33.51	37.01	16.80	20.26	20.94	16.38	16.44	14.78
$B + S^{lh}$	34.98	39.30	38.99	25.02	34.38	38.14	15.77	20.37	21.43	16.70	16.55	15.07
$B + R^{lh}$	34.44	38.54	38.24	24.85	33.79	37.11	16.79	20.60	21.26	16.99	16.48	14.91
$B + f$	34.80	39.54	39.08	25.00	34.30	37.63	15.90	19.80	21.00	16.59	16.67	15.24
$B + S^{lh} + R^{lh}$	35.07	39.56	39.05	25.25	34.57	38.17	16.63	20.41	21.36	16.92	16.58	15.25
$B + S^{lh} + R^{lh} + f$	35.23	40.22	39.48	25.27	34.78	38.33	15.83	20.40	21.52	16.64	16.72	15.45
Attention-Seeker	35.49	40.14	39.22	25.40	34.53	38.50	16.74	20.26	21.73	17.14	16.89	15.27

Table 3: Performance of Attention-Seeker using different relevance scores in **short** documents.

strates the advantages of considering the relevance of different SAMs and document segments when estimating the attention scores.

Although PromptRank remains the best-performing method on the SemEval2017 dataset, it requires hyperparameter tuning for each specific benchmark (Kong et al., 2023), which limits its suitability for applications without labeled data. Furthermore, Kong et al. (2023); Song et al. (2024) demonstrate that the choice of prompts significantly affects its performance, requiring an extensive search to identify the optimal prompt. In contrast, Attention-Seeker achieves the second-best performance on this dataset without requiring parameter tuning. As Kang and Shin (2023) note, different input texts activate different SAMs for keyphrase identification. Attention-Seeker adapts to these variations by analyzing the SAMs triggered by the input, positioning it as a robust and adaptable solution for unsupervised keyphrase extraction.

5 Ablation study

5.1 Short documents

Attention-Seeker integrates several modules designed to improve its overall performance. Specifically, it employs a relevance score for each SAM's attention vector (S^{lh}), a relevance score for each SAM (R^{lh}), and a filtering step f applied to non-candidate tokens of S^{lh} (Section 3.4). To evaluate their contributions, we applied these modules separately. We also tested configurations using both scores S^{lh} and R^{lh}, with and without the filtering step. Table 3 presents the performance of Attention-Seeker in all these cases. In the "Base" configuration, all attention vectors and SAMs are assigned equal relevance, and no filtering is applied. The following configurations progressively incorporate the relevance scores (S^{lh} and R^{lh}) and the filtering step (f) into the base method (B). The final version of Attention-Seeker corresponds to "$B + S^{lh} + R^{lh} + f$" applied twice (Figure 2). Performance on the SemEval2010 and Krapivin datasets is evaluated using only the abstracts of the papers (short documents).

Table 3 shows that each module contributes positively to the overall performance of our method, with the S^{lh} score providing the most significant improvements. While our proposed relevance scores are supported by the observations of Clark et al. (2019), the benefit of filtering attention vectors from certain tokens is less straightforward. Clark et al. (2019) observed that in SAMs specialized in attending to verbs of direct objects, attention vectors of non-nouns often attend to the [SEP] token. They suggest that in such cases, this token might function as a "non-op", implying that the remaining attention scores would be randomly distributed (the model ignores this information). Our results in the "$B+f$" configuration suggest this hypothesis is correct, as the filtering step would be removing these noisy scores. However, results from SemEval2010 indicate that our filtering approach may also cause slight performance degradation, likely due to the simplicity of the current filtering mechanism. Therefore, future research should investigate how SAM's attention vectors differ based on different query tokens and determine which tokens produce more effective attention distributions for representing contextual information.

5.2 Long documents

Attention-Seeker segments the document into abstract and equal-length segments of the remaining content. Since this approach slightly differs from the base method (Kang and Shin, 2023; the entire document is segmented into equal parts), we evaluated its impact on the final performance. In addition, we evaluated the contributions of the rele-

Method	SemEval2010			Krapivin		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
Base	17.86	20.99	22.07	17.38	16.78	15.15
B_{as}	17.35	21.04	21.05	17.05	16.50	14.83
$B_{as} + R^{lh}$	17.34	21.27	21.83	17.44	16.79	15.17
$B_{as} + T^s_b$	18.20	21.52	22.72	18.58	17.39	15.88
$B_{as} + T^s$	19.05	22.75	23.69	20.49	17.94	16.01
Attention-Seeker	19.00	23.07	23.81	20.79	18.25	16.22

Table 4: Performance of Attention-Seeker using different relevance scores in **long** documents.

vance score for each SAM (R^{lh}) and the relevance score for each document segment T^s. We considered two versions of this second relevance score: One uses our proposed refined vector H^s (T^s), and the other uses a simple binary vector H^0_b that assigns one to abstract candidates and zero otherwise (T^s_b). Table 4 summarizes the results. In the "Base" and "B_{as}" configurations, all SAMs and segments are treated as equally relevant ("B_{as}" considers the abstract as the first segment). The "B_{as}" configuration is then extended with relevance scores (R^{lh}, T^s_b, and T^s), and the final Attention-Seeker corresponds to $B_{as} + R^{lh} + T^s$.

Table 4 shows that our two proposed relevance scores contribute positively to the performance of Attention-Seeker, with the relevance of segments providing the most significant improvements. In addition, refining the hypothesis vector H^s further improves the performance of the segment scores ($T^s > T^s_b$). Despite these advances, our segmentation approach introduces some performance degradation compared to the base model, likely due to the difference in length between the abstract and other segments. While the overall improvement achieved by our method outweighs the initial degradation, further research may consider exploring new methods for estimating segment relevance without compromising the performance of the base model.

Finally, the contribution of SAMs' relevance scores R^{lh} in Table 4 is consistent with the results in Table 3, suggesting that our approach for short documents is equally effective when applied to segments of long documents. This result suggests that the inclusion of the relevance score S^{lh} has the potential to further improve the performance of Attention-Seeker for long documents. Further research should explore methods for normalizing attention scores across contexts that vary in syntax, number of nouns, topics, and other factors.

5.3 Relevance of SAMs

Attention-Seeker's score R^{lh} identifies the SAMs most relevant for keyphrase extraction, eliminating the need for labeled data to select an optimal SAM. Our analysis of the R^{lh} scores across samples from four datasets (Inspec, SemEval2017, SemEval2010, and Krapivin) reveals that SAMs from intermediate layers (9 to 15 out of 32), along with a few SAMs from the first and last layer contribute significantly to the keyphrase extraction task (see Appendix D).

6 Conclusion

We proposed Attention-Seeker, a self-attention-based method for unsupervised keyphrase extraction that eliminates the need for manual parameter tuning. Our method assumes that certain Self-Attention Maps (SAMs) are specialized to focus on the most important phrases within a document. Accordingly, Attention-Seeker reframes keyphrase extraction as the task of identifying where the most crucial information is encapsulated in SAMs. For long documents, we extended this method by first identifying the most relevant segments from which to extract attention scores.

Attention-Seeker outperformed most baselines on four benchmark datasets, demonstrating the effectiveness of our approach. Since Attention-Seeker is the first method to estimate the relevance from the internal scores of a Large Language Model (LLM), its scoring mechanism is very simple and has potential for improvement, as suggested in our study. Future research should explore more sophisticated methods for relevance estimation, which could lead to further performance improvements. In addition, optimizing these relevance estimates may provide deeper insights into the internal processes of LLMs, potentially contributing to the design of more efficient pretrained language models and their application in unsupervised tasks.

7 Limitations

Attention-Seeker estimates the relevance of attention vectors by defining a set of desired characteristics and measuring how closely each attention vector matches them. However, the current implementation of this approach presents two major limitations. First, it uses the dot product between the desired vector and SAMs' attention vectors. While this method is effective, it could be improved by applying normalization techniques such as l_2-normalization (cosine similarity), softmax normalization (cross-attention), or softmax with temperature scaling.

Second, the hypothesis vectors (H), which serve as reference vectors, are initially defined as binary vectors that focus on candidate phrases. This simplistic formulation may introduce noise into the attention distribution of the candidates and partially overlook important contextual information. Although we refine this vector (H) by using the attention scores extracted from the document (B), the initial estimation of B still relies on the original binary vector. To address these limitations, future work could explore more sophisticated hypothesis vectors and improved methods for measuring their alignment with SAM's attention vectors.

Another important limitation is that the long document version of Attention-Seeker requires the first segment to be the abstract of the document. This condition limits the application of our proposed method to documents with an abstract. In addition, our ablation study showed that this approach negatively affects the overall performance of our method. Accordingly, we believe that further research should explore alternative methods for defining the hypothesis vector H^s without relying on information extracted from the abstract. Possible solutions could include extracting the most important tokens from all segments under the assumption of equal relevance, and defining the hypothesis vector H^s based on these tokens.

Finally, the implementation of our method is currently limited to open LLMs. Since Attention-seeker relies on the information provided by the Self-Attention Maps of Language Models, it cannot be applied to private models with restricted access to their weights, such as Open AI's GPT-4o/Open-o1 and Antropic's Claude models. This limitation is common in the field, as previous methods also rely on internal model information (e.g., SAMs, embeddings, logits). However, as open models continue to improve and close the performance gap with closed models, we believe our approach will provide valuable insights for future research.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR22D1 and JSPS KAKENHI Grant Number JP22H00551, Japan.

References

AI@Meta. 2024. Llama 3 model card.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

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.

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.

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. 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.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, Vancouver, Canada. Association for Computational Linguistics.

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.

Byungha Kang and Youhyun Shin. 2023. SAMRank: Unsupervised keyphrase extraction using self-attention map in BERT and GPT-2. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10188–10201, Singapore. Association for Computational Linguistics.

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, Uppsala, Sweden. Association for Computational Linguistics.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, and Xiaoyan Bai. 2023. PromptRank: Unsupervised keyphrase extraction using prompt. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9788–9801, Toronto, Canada. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrase extraction.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matej Martinc, Blaž Škrlj, and Senja Pollak. 2022. Tnt-kid: Transformer-based neural tagger for keyword identification. *Natural Language Engineering*, 28(4):409–448.

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, Barcelona, Spain. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, Nevada, United States. Curran Associates, Inc.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In *Advances in Information Retrieval*, pages 328–335, Cham. Springer International Publishing.

Mingyang Song, Yi Feng, and Liping Jing. 2024. A preliminary empirical study on prompt-based unsupervised keyphrase extraction. *Preprint*, arXiv:2405.16571.

Mingyang Song, Liping Jing, and Lin Xiao. 2021. Importance Estimation from Multiple Perspectives for Keyphrase Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2726–2736, Online and

Punta Cana, Dominican Republic. Association for Computational Linguistics.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 60(5):493–502.

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*, SIGIR'19, page 755–764, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, page 855–860. AAAI Press.

Linhan Zhang, Qian Chen, Wen Wang, Chong Deng, ShiLiang Zhang, Bing Li, Wei Wang, and Xin Cao. 2022. MDERank: A masked document embedding rank approach for unsupervised keyphrase extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 396–409, Dublin, Ireland. Association for Computational Linguistics.

A Optimal Performance of SAMRank on LLAMA 3-8B

We adapted the original SAMRank method (Kang and Shin, 2023) from the official repository `https://github.com/kangnlp/SAMRank`, using the LLAMA 3-8B model instead of GPT-2. Due to the structural similarities between GPT-2 and LLAMA 3-8B, no major modifications were required. Table 5 shows the results obtained by the optimal SAMs on four benchmark datasets. We evaluated the performance of the method using both Global (G) and Final (F) scores.

The performance of the final score in SAMRank LLAMA 3-8B is consistent with the results reported by Kang and Shin (2023) for SAMRank using GPT-2, LLAMA 2-7B and LLAMA 2-13B.

Dataset	**F1@5**	**F1@10**	**F1@15**	**SAM**
SAMRank Final Score (S)				
Inspec	33.19	39.04	39.50	L:19 H:15
SemEval2017	24.50	34.05	37.65	L:21 H:11
SemEval2010	16.30	18.04	19.74	L:12 H:2
Krapivin	18.55	17.25	15.63	L:8 H:8
SAMRank Global Score (G)				
Inspec	35.60	40.47	40.21	L:9 H:7
SemEval2017	25.64	36.41	39.80	L:20 H:28
SemEval2010	17.10	19.32	20.40	L:12 H:2
Krapivin	19.18	17.60	15.93	L:12 H:2

Table 5: Performance of keyphrase extraction of SAMRank (Global and Final scores) on four datasets and their corresponding heads.

However, in our implementation, the independent Global Score provides the best results, suggesting that the Proportional Score may be less effective in this context. While the ablation study conducted by Kang and Shin (2023) demonstrated the benefits of the Proportional Score for BERT and GPT-2 implementations, its effectiveness for larger LLMs such as LLAMA remains unclear. We believe that further research is needed to determine the impact of the Proportional Score on LLM implementations of SAMRank.

The independent implementation of the Global Score in SAMRank outperforms Attention-Seeker in short documents, suggesting room for improvement in our method. As discussed in the main paper, these improvements could include more effective strategies for defining the desired characteristics of attention vectors and better measuring their alignment with these characteristics. In future implementations, we might expect results similar to those reported in Table 5 (Global Score). However, it is important to note that these results may be inflated by non-relevant SAMs achieving high performance by chance, due to suboptimal labels and the large number of 1024 possible trials.

Finally, Attention-Seeker outperforms the Global Score of SAMRank in long documents, highlighting the importance of our proposed relevance score for document segments T^s. While this score proves crucial for extracting keyphrases from long documents, the results also respond to the limitation of SAMRank in selecting a single SAM for all document segments. When different segments have their own optimal SAM, this constraint leads to suboptimal results. In addition, the reduced likelihood of selecting a non-relevant SAM

Method	Inspec			SemEval2017			SemEval2010			Krapivin		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
BERT Model												
SAMRank (B)	32.43	38.26	38.84	22.93	32.14	35.85	15.19	18.70	20.21	11.99	12.51	12.14
Attention-Seeker	**34.54**	**40.31**	**40.15**	**24.13**	**34.17**	**38.30**	**17.27**	**21.53**	**23.05**	**19.09**	**17.60**	**15.56**
SAMRank (O)	33.96	39.35	39.73	24.08	33.40	37.53	15.28	18.36	18.03	16.35	15.91	14.52
GPT-2 Model												
SAMRank (B)	33.96	38.24	38.11	23.99	32.51	35.75	17.23	20.76	21.56	13.37	15.19	14.27
Attention-Seeker	**34.14**	**39.46**	38.89	24.30	33.83	37.16	**18.82**	**23.08**	**23.73**	**19.89**	**17.95**	**15.80**
SAMRank (O)	33.92	39.44	**39.72**	**24.80**	**34.75**	**38.78**	15.88	18.36	19.03	17.49	16.46	14.92

Table 6: Performance of keyphrase extraction of SAMRank and Attention-Seeker on Small Language Models. SAMRank (B):non-parametric base method. SAMRank (O): optimal manual selection (Kang and Shin, 2023).

contributes to a more stable selection with lower performances. This stability is reflected in the results of the SemEval2010 and Krapivin datasets in Table 5, where in 3 out of 4 cases, the selected SAM is consistently from layer 12, head 2.

B Performance in Small Language Models: BERT and GPT-2

We replicated the evaluation of Attention-Seeker and the non-parametric SAMRank method (Base) from the main paper, applying our method to smaller Language Models, specifically BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019). The results of this evaluation are summarized in Table 6.

These results align with those obtained using the LLAMA 3-8B implementation (Table 2) and indicate that Attention-Seeker always outperforms the non-parametric SAM (base model that treats all layers and heads as equally relevant; Appendix C). These findings emphasize the importance of considering the relevance of an LLM's layers and heads in the process of keyphrase extraction.

Similar to the analysis in Appendix A, we further compare the results of Attention-Seeker with the optimal performance of SAMRank reported by Kang and Shin (2023). For short documents, the optimal results from SAMRank suggest room for improvement in our method, while for long documents, Attention-Seeker consistently outperforms SAMRank. Although these results are similar to those obtained with the LLAMA 3-8B implementation, Attention-Seeker outperforms the optimal SAMRank in a significant number of cases for short documents. These results suggest that the simplicity of our hypothesis vector is more effective when using smaller models, while larger models may require more sophisticated vectors to achieve optimal performance.

C Results of Non-Parametric SAMRank

The non-parametric SAMRank introduced in this study implements the method proposed by SAMRank (Kang and Shin, 2023). However, instead of selecting a single SAM from the heads of the LLM (as in the original parameterized method), it averages all SAMs. Given the set of SAMs $A = \{A^{lh}, \forall l, h\}$, the averaged SAM A^{av} is calculated as shown in Equation 12 (p=q=32 for LLAMA 3-8B).

$$A^{av} = \frac{1}{p * q} \sum_{l=1}^{p} \sum_{h=1}^{q} A^{lh} \tag{12}$$

The global score is calculated as shown in Equation 13 (n is the number of tokens of the document or segment).

$$G = \sum_{i=1}^{n} \sum_{j=1}^{m} A^{av}_{ij} \tag{13}$$

The proportional score is calculated as shown in Equation 14. Here, A^x is equal to the matrix A^{av}, where its elements A^{av}_{i0} are set to zero.

$$P = \sum_{j=1}^{n} \frac{A^x_{ij}}{\sum_{i=1}^{n} A^x_{ij}} \tag{14}$$

Equation 14 is a simplification of the equations proposed by Kang and Shin (2023), as their redistribution of the global score G is cancelled out by the column normalization of the matrix. The only observable effect of this redistribution is that all elements A^{av}_{i0} become zero, since the global score G for the first token is zero.

The final score S is determined by the sum of the global score G and the proportional score P, as

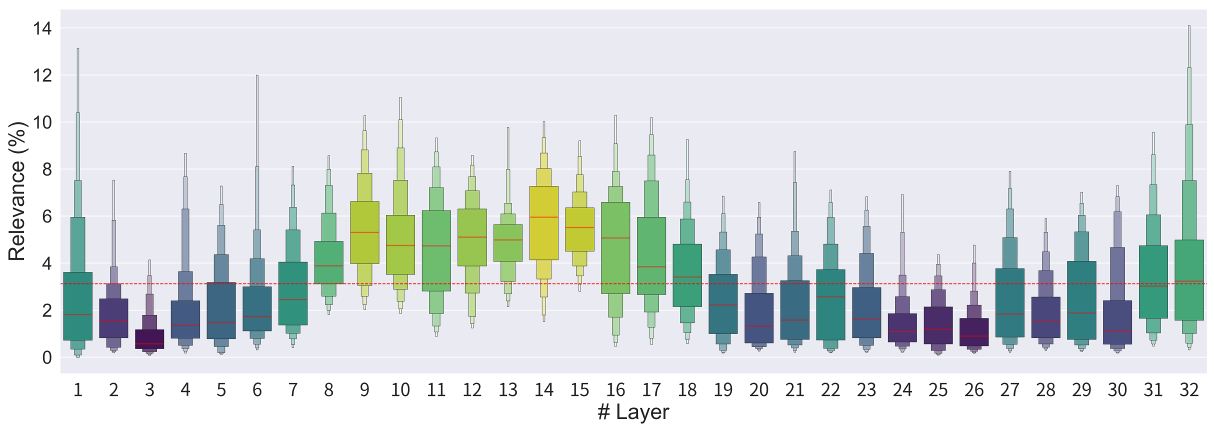


Figure 4: Relevance of LLAMA 3-8B layers for keyphrase extraction estimated by Attention-Seeker over four datasets. (The lighter the color, the higher the relevance rank of the corresponding layer)

shown in Equation 15. The score for each candidate phrase is derived from the attention scores of this vector, as detailed by Kang and Shin (2023).

$$S = G + P \tag{15}$$

Dataset	**F1@5**	**F1@10**	**F1@15**
SAMRank Global Score (G)			
Inspec	34.25	38.18	38.11
SemEval2017	24.74	33.51	37.01
SemEval2010	17.86	20.99	22.07
Krapivin	17.38	16.78	15.15
SAMRank Proportional Score (P)			
Inspec	27.21	33.81	35.20
SemEval2017	20.49	25.90	33.01
SemEval2010	15.75	19.07	19.34
Krapivin	14.84	14.45	13.27
SAMRank Final Score (S)			
Inspec	30.23	36.62	38.08
SemEval2017	21.95	31.83	35.39
SemEval2010	16.36	20.13	20.87
Krapivin	16.06	15.46	14.29

Table 7: Performance of keyphrase extraction of the three scores of a non-parametric SAMRank on four datasets.

We implemented the non-parametric SAMRank method using the model LLAMA 3-8B, evaluating their three different scores to ensure that the non-parametric version benefits from these scores similarly to the original SAMRank. Table 7 shows the performance achieved by the non-parametric SAMRank in the three cases.

We observe a similar result than in the optimized implementation of SAMRank. The Proportional Score degrades the improvement of the overall performance, causing the implementation with the Global Score to achieve the best results. Accordingly, the performance reported in Table 2 is collected from the implementation of SAMRank that only use the Global Score.

D Most Relevant SAMs identified by Attention-Seeker

Attention-Seeker estimates the relevance of SAMs, their internal attention vectors, and their corresponding segments. While analyzing the relevance of attention vectors and document segments requires a deep study of different cases (influence of both the input text and the selected SAM), the relevance of SAMs is more closely related to the model characteristics. Specifically, the layers of LLMs represent different levels of text processing, with higher layers capturing more complex token relationships. To investigate the impact of different layers on keyphrase extraction, we extracted the relevance scores R^{lh} estimated by Attention-Seeker across all samples of the four datasets used in our study (Inspec, SemEval2017, SemEval2010, and Krapivin).

Figure 4 shows the distribution of R^{lh} scores for each layer, with the scores normalized for consistent comparison. For better visualization, the top 5% of outliers are filtered out from each distribution, distributions are color-coded according to their relevance rank, and a red line is added at $Relevance = 3.125$ to indicate a uniform distribution where all layers are equally relevant. The analysis shows that the heads between layers 8 and 18, especially layers 9, 12, 14, and 15, received the highest relevance scores. This result suggests that

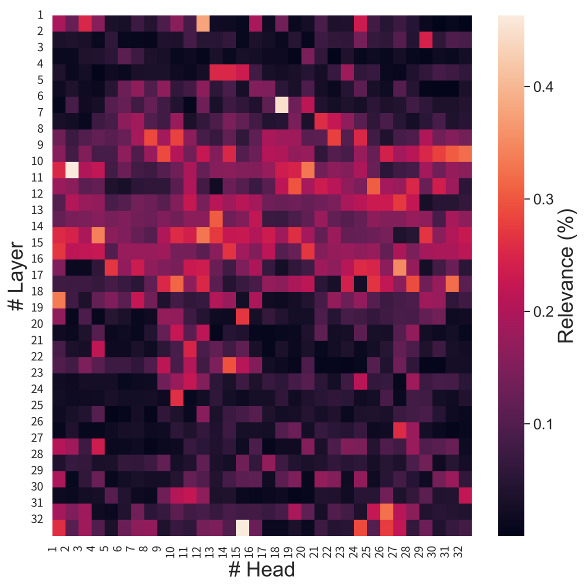
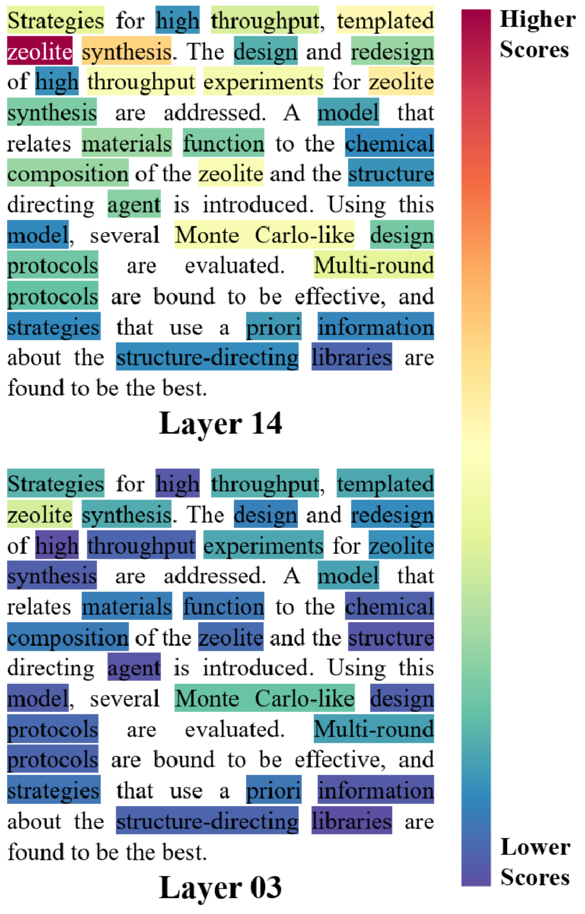


Figure 5: Relevance of LLAMA 3-8B heads for keyphrase extraction estimated by Attention-Seeker in one sample of the Inspec dataset.

heads located in intermediate layers of an LLM tend to specialize in processing tasks related to noun-noun relationships and selecting the most relevant nouns/phrases.

The first and last layers show high variability in relevance scores across heads, with some receiving remarkably high scores. This suggests that attention to keyphrases is a fundamental task: At low levels, it helps to generate well-contextualized embeddings, while at high levels, it contributes to building a complete representation of the document context. At both levels, however, this process is complemented by numerous other tasks performed by different heads.

Figure 5 supports the previous observation by illustrating the relevance scores R^{lh} assigned to each head for a sample from the Inspec dataset. The score distribution across layers aligns with the general pattern in Figure 4, where most heads in the intermediate layers contribute significantly to keyphrase extraction, while only a few heads in the first and last layers exhibit high relevance. For the same Inspec sample, we compared the attention scores of candidate phrase tokens in layer 3 (lowest overall relevance score) and layer 14 (highest overall relevance score). Figure 6 shows the difference between these two layers: Layer 14 assigns higher attention scores to all candidates, with a clearer contrast between phrases, while layer 3 tends to focus on non-candidate tokens, resulting in lower, more uniformly distributed attention scores.



Strategies for high throughput, templated zeolite synthesis. The design and redesign of high throughput experiments for zeolite synthesis are addressed. A model that relates materials function to the chemical composition of the zeolite and the structure directing agent is introduced. Using this model, several Monte Carlo-like design protocols are evaluated. Multi-round protocols are bound to be effective, and strategies that use a priori information about the structure-directing libraries are found to be the best.

Layer 14

Strategies for high throughput, templated zeolite synthesis. The design and redesign of high throughput experiments for zeolite synthesis are addressed. A model that relates materials function to the chemical composition of the zeolite and the structure directing agent is introduced. Using this model, several Monte Carlo-like design protocols are evaluated. Multi-round protocols are bound to be effective, and strategies that use a priori information about the structure-directing libraries are found to be the best.

Layer 03

Figure 6: Attention scores in Layers 3 and 14 in one sample of the Inspec dataset.

E Computational cost

We evaluated the computation time required for the three methods with the best general performance in unsupervised keyphrase extraction: SAMRank, PromptRank, and Attention-Seeker. The computations were performed on an NVIDIA RTX A6000 GPU with 48 GB of memory. In all implementations, documents were processed individually (batch size = 1). We conducted the evaluation using four Language Models: BERT, GPT-2, and T5 for small-sized models, and LLAMA 3-8B for medium-sized models.

The results are shown in Table 8. PromptRank generally requires more computation time than the other two methods. This is due to its specific processing requirements: it extracts the logit for each "document" + "candidate" combination, effectively processing almost the same text n times, where n is the number of keyphrase candidates. A potential solution is to process all combinations as a batch, but this would shift the computational burden from time to memory resources.

Model	**Method**	**Inspec**	**SemEval2017**	**SemEval2010**	**Krapivin**
Small-sized Language Models					
BERT	SAMRank	0.04	0.06	6.76	6.74
	Attention-Rank	0.12	0.13	7.89	7.89
GPT-2	SAMRank	0.05	0.06	7.16	7.33
	Attention-Rank	0.12	0.13	7.90	8.11
T5	PromptRank	2.25	2.51	3.44	8.85
Medium-sized Language Models					
LLAMA 3-8B	SAMRank	0.22	0.30	11.98	12.05
	Attention-Rank	0.68	0.76	14.00	14.01
	PromptRank	5.88	9.89	45.12	42.82

Table 8: Average computation time (seconds) required by SAMRank, Attention-Rank, and PromptRank on small and medium-sized Language Models to process one document (batch size = 1).

Among the attention-based methods, SAMRank shows better computational efficiency compared to Attention-Seeker. This difference is expected because Attention-Seeker builds on SAMRank by introducing additional computations, including vector similarity calculations and linear transformations of SAMs. Despite these additional steps, both methods have comparable memory consumption, are efficient enough for real-time processing of short documents, and show similar computation times for long documents. Due to its superior performance, Attention-Seeker remains the preferred option overall. However, in scenarios where processing speed is critical for short documents and a small trade-off in F-score (about 2%) is acceptable, the non-parametric SAMRank would be the better choice.