Fusing Sentence-Level and Document-Level Representations for Keyphrase Extraction

Marko Pranjić[1,2] **Kristijan Armeni**[3] **Senja Pollak**[1]
[1]Jožef Stefan Institute, Ljubljana, Slovenia
[2]Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3]Independent Researcher
marko.pranjic@ijs.si senja.pollak@ijs.si
armeni@validateresearch.com

Abstract

Supervised sequence-labeling approaches to keyphrase extraction operate at token level, but whether a phrase is relevant may depend on context beyond its sentence. Processing an entire document at token resolution can increase sequence length and introduce irrelevant information. We investigate a mixed-resolution context-fusion architecture that retains token representations for the target sentence while compressing available surrounding sentences into pooled representations. Both representation types are placed in document order and processed jointly by a self-attention module before IOB prediction. We evaluate the model on kp20k, inspec, and kptimes against SEKE, a recent state-of-the-art supervised system with approximately 60% more total parameters. The model achieves comparable F1 scores on the two scientific-domain benchmarks, while a larger performance gap remains on the journalistic benchmark. Error analysis shows that the model generates substantially more candidates than there are gold keyphrases. However, many predictions counted as false positives under exact-match evaluation partially overlap with gold keyphrases, while exact matches tend to receive higher scores than partial and non-matches.

1 Introduction

Keyphrases provide compact descriptions of a document's main topics and support indexing, categorization, and information retrieval (Turney, 2002). In scientific publishing, automatically extracting such phrases can help organize and retrieve information from rapidly growing literature collections. When these collections are large, manual annotation is impractical, motivating automated keyphrase extraction.

The "keywordiness" of a word or a phrase is largely determined by the global document context. In natural language processing, representations that depend on the word's contextual role in the text, known as contextualized word embeddings, are a hallmark of transformer language models pretrained on large text corpora (Devlin et al., 2019). This is underpinned by the model's ability to weigh the importance of all left and right context words when learning the representations for the current word, a process known as self-attention (Vaswani et al., 2017; Bahdanau et al., 2014). Contextual embeddings and self-attention have proven powerful architectural components in a wide array of downstream NLP tasks, including sequence labeling for keyphrase extraction (Sahrawat et al., 2020; Martinc et al., 2022, 2025).

Although in principle transformer large language models (LLMs) process the entire context when performing a task (e.g. a complete document when extracting keyphrases), *in practice* this risks burying the relevant signal among distracting context, especially as document length increases (Liu et al., 2023; Martínez-Cruz et al., 2024; Hong et al., 2025). Alternatively, extracting keyphrases sentence by sentence means transformers would discard document-level context altogether. Keyphrase-extraction approaches have addressed representation and context limitations through recurrent classification heads (Sahrawat et al., 2020; Martinc et al., 2022), task-specific training objectives (Kulkarni et al., 2022), document-attention and graph-based augmentation (Santosh et al., 2020; Martínez-Cruz et al., 2025), and mixture-of-experts (MoE) components (Martinc et al., 2025).

We investigate a particular context-augmentation construction for supervised keyphrase extraction. Instead of processing the entire document at token resolution, we retain token embeddings for the target sentence, compress each surrounding sentence into one pooled vector, and combine both representation types in a single ordered input to a three-layer self-attention module (Figure 1). This provides an explicit representation of the document

Proceedings of the 4th Workshop on Artificial Intelligence for Scientific Publications, pages 122–132
November 10, 2026 ©2026 Association for Computational Linguistics

context without sacrificing the token resolution of the target sentence.

Our contributions are threefold:

1. We investigate a mixed-resolution context-fusion architecture for keyphrase extraction that preserves token-level representations for the target sentence while compressing surrounding document context into sentence-level representations. A three-layer self-attention module jointly contextualizes target-token representations with sentence-level representations of the surrounding context before keyphrase prediction.

2. We evaluate the complete 170.9M-parameter architecture on three established keyphrase-extraction benchmarks under fine-tuning settings aligned with SEKE, a recent state-of-the-art supervised keyphrase-extraction system with 274M parameters. The architecture achieves comparable performance on the scientific-domain benchmarks while exhibiting substantially weaker performance on the journalistic benchmark, characterizing both the feasibility and current limits of the approach.

3. We characterize the model's principal extraction errors and show that low precision is associated with systematic over-generation. The candidate sets nevertheless contain substantial partial overlap with gold keyphrases, while exact matches tend to receive higher scores than partial and non-matches, suggesting that candidate selection and phrase delimitation are important targets for further improvement.

Code availability. Code and experiment configurations are available at `https://github.com/knowledge-technologies/context-fusion-kpe`.

2 Related work

Our work takes as a starting point the approaches leveraging transformer-based language models (Vaswani et al., 2017) for keyphrase extraction. In one of the earlier works, Sahrawat et al. (2020) combined pretrained transformer language models with recurrent neural networks and showed that using contextualized tokens improves keyphrase extraction performance across several benchmarks and models over static embeddings. Martinc et al. (2022) proposed a transformer-only architecture consisting of a next-word prediction pretraining followed by a keyphrase classification objective as a competitive alternative to unsupervised graph-based and recurrent neural network based keyphrase extractors at the time.

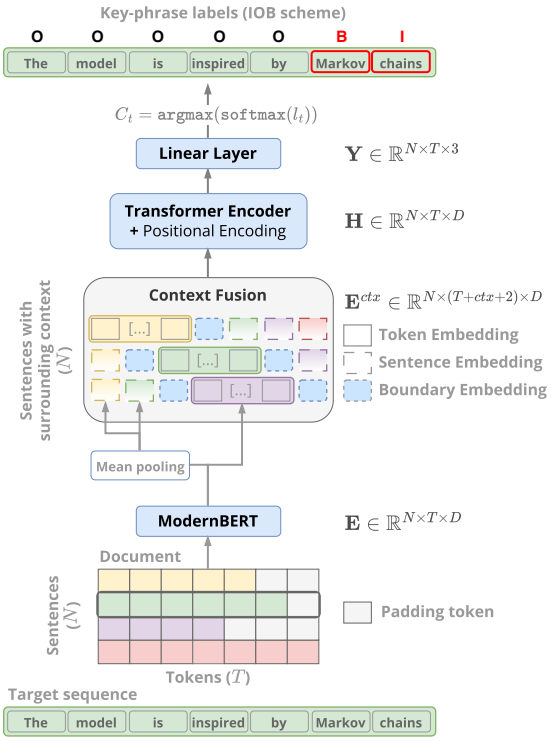


Figure 1: Model architecture overview. Token embeddings of the target sentence and pooled embeddings of the surrounding sentences form a single ordered input to the context-fusion module.

The use of broader document context for keyphrase extraction is well established. In unsupervised extraction, Liang et al. (2021) jointly modeled phrase–document relevance and local graph structure. More directly, DAKE (Santosh et al., 2020) enriched sentence-level BiLSTM word representations with supporting information from other sentences through document-level attention and gating before CRF decoding. Martínez-Cruz et al. (2025) instead augmented contextualized language-model token representations with embeddings of a document-wide co-occurrence graph. These studies establish the value of augmenting local representations with document information. Our architecture differs in the fusion operation: each surrounding sentence becomes one coarse-grained position alongside the fine-grained target-token positions in a single ordered sequence, enabling joint self-attention between both representation types rather

than explicit document attention and gating or concatenated graph features.

Most recently, Martinc et al. (2025) introduced SEKE, a state-of-the-art supervised keyphrase-extraction system that combines a DeBERTa backbone with a mixture-of-experts (MoE) (Jacobs et al., 1991) and BiLSTM token classification head. They froze the original backbone weights while training low-rank adaptations to its query and value projections (Hu et al., 2022), together with all parameters of the custom classification head. During inference, the full DeBERTa backbone remains active, while a learned router selects the top-k experts that process each token.

3 Methods

3.1 Benchmarks

We fine-tuned the model on the validation splits of three keyphrase benchmarks: kp20k dataset (Meng et al., 2017), kptimes (Gallina et al., 2019), and inspec (Hulth, 2003) (as in (Martinc et al., 2025); see Section 3.5). Summary statistics for validation datasets are presented in Table 1.

3.2 Preprocessing

We needed to prepare the benchmark data to conform to our annotation and extraction scheme. The document title, abstract, and full text (if present) were first segmented into sentences using the SpaCy[1] library with the en_core_web_sm model. Each sentence was tokenized using the Huggingface tokenizer for the pretrained model we used (answerdotai/ModernBERT).

Annotating target tokens. To annotate class labels to the tokenized dataset, we performed matching of the tokens in the document with the ground truth keyphrases provided in the 'keyword' (or 'keywords') field. We first stemmed both document and ground truth keyphrases using the nltk Porter stemmer. For each sentence, we then iterated over the stemmed ground truth keyphrases and checked for matches with stemmed document words. This stem-based matching allowed us to boost the relevant training signal by ignoring the variance due to word surface forms.

3.3 Task definition

Keyphrase extraction is a 3-way classification problem. The task involves classifying each token at position t in the target sentence j either as a start of a new keyphrase (B - beginning), continuation of a multi-token keyphrase (I - inside) or a non-keyphrase token (O - outside). This is known as the IOB labeling scheme. An example is shown in Table 3.

3.4 Model architecture

The overall architecture consists of two stages: a document encoder and a classification head. The document encoder provides the surrounding context for the target sentence for which the candidate keyphrases are extracted. The final output is a 3-way prediction for each token in the target sequence.

Token embedding layer (ModernBERT) The document encoder is the pretrained bidirectional encoder ModernBERT (Warner et al., 2024), applied independently to each sentence. Every sentence is tokenized with the matching tokenizer and truncated to at most 512 subtokens, and the last hidden state of the encoder provides the token-level embeddings. For a target sentence j, these are the embeddings $\mathbf{w}_{j,t} \in \mathbb{R}^d$ used in the fused sequence below; for context sentences, the same embeddings are mean-pooled into the sentence embeddings $\mathbf{s}_k$. The embedding dimension is $d = 768$, matching the hidden size of `ModernBERT-base`.

Context fusion with sentence embeddings For each target sentence j in a document, we construct a fused input sequence containing, in order: the *sentence* embeddings of the preceding document context, a learnable boundary embedding, the *token* embeddings of the target sentence, a second learnable boundary embedding, and the *sentence* embeddings of the following document context:

$$\mathbf{x}_j = \big[\mathbf{s}_{j-c_{\text{pre}}}, \ldots, \mathbf{s}_{j-1},\ \mathbf{m}_{\text{pre}},\ \mathbf{w}_{j,1}, \ldots, \mathbf{w}_{j,T_j},\ \mathbf{m}_{\text{post}},\ \mathbf{s}_{j+1}, \ldots, \mathbf{s}_{j+c_{\text{post}}}\big]$$

where: $\mathbf{s}_k \in \mathbb{R}^d$ is the mean-pooled sentence embedding of sentence k (before or after the current target sentence j); $c_{\text{pre}} = \min(j, n_{\text{left}})$ and $c_{\text{post}} = \min(N - j - 1, n_{\text{right}})$ are the context window sizes, bounded by document length N; $\mathbf{w}_{j,t} \in \mathbb{R}^d$ is the token embedding of token t in target sentence j; $\mathbf{m}_{\text{pre}}, \mathbf{m}_{\text{post}} \in \mathbb{R}^d$ are learnable boundary embeddings. During training, sentences are processed in document-ordered batches of 50;

[1] https://spacy.io/

Statistic	kp20k	kptimes	inspec
Domain	CS	news	CS
Nr. Documents	19,992	10,000	1,500
Mean abstract length (words)	146.5	776.3	117.1
Max abstract length (words)	1,547	5,617	490
Mean keyphrases/doc (all/in doc)	5.3 / 3.8	5.0 / 2.7	9.6 / 7.7
Vocabulary size (first 1k docs)	11,659	34,875	10,277
Words per keyphrase (all/in doc)	1.9 / 1.8	2.2 / 1.6	2.3 / 2.4

Table 1: Validation dataset statistics for the three benchmarks

for documents longer than one batch (primarily kptimes), the context is drawn from the sentences within the current batch. Fused sequence lengths are therefore bounded well below the 768 positional embeddings of the context-fusion head.

Context fusion and output projection. Learnable positional embeddings are added to each position: $\tilde{\mathbf{x}}_{j,i} = \mathbf{x}_{j,i} + \mathbf{p}_i, \quad 0 \le i < |\mathbf{x}_j|$. The resulting sequence is passed through a three-layer self-attention module with a 768-dimensional representation split across four attention heads:

$$\mathbf{h}_j = \text{ContextFusion}(\tilde{\mathbf{x}}_j)$$

Token-level logits for sentence j are extracted from each position t in the sequence through a linear layer: $\ell_{j,t} = \mathbf{W}_{\text{out}}\,\mathbf{h}_{j,t}$. All parameters are fine-tuned: the ModernBERT encoder has 149.0M parameters and the context-fusion head 21.9M, for 170.9M parameters in total.

Inference computation. A batch of sentences from the same document is encoded once by ModernBERT to obtain token-level representations. Average pooling yields one sentence-level representation per sentence, without another encoder pass. For each target sentence, we construct an input combining its fine-resolution token representations with the coarse-resolution sentence representations of the surrounding sentences. Each resulting input is processed only by the context-fusion head, which contains approximately 13% of the model's parameters (21.9M of 170.9M).

3.5 Fine-tuning

Following Martinc et al. (2025), we fine-tuned our model in two regimes: *validation set fine-tuning* and *training set fine-tuning* (kp20k only). The main difference between the two is the size of the dataset used in fine-tuning. In validation set fine-tuning, we fine-tuned the model on the validation set split for the respective benchmark. We randomly sampled 80% of the documents from the validation split[2]. We fine-tuned the model for up to 20 epochs and used early stopping (metric: F1@10, delta: 0.001, patience: 5 epochs). In training-set fine-tuning, we used the full training set split for fine-tuning. We fine-tuned for 20 epochs without early stopping. This corresponds to the regime dubbed "pretraining" in (Martinc et al., 2025, Appendix E, Table 5).

In both cases, we used AdamW with learning rates of 2×10^{-5} for the base encoder and 5×10^{-5} for the context-fusion head. The effective batch size was 100 sentences, implemented as two 50-sentence forward passes. We used cross-entropy loss with class weights inversely proportional to class frequency: $w_c = \frac{n_samples}{(n_classes * n_samples_c)}$. Special tokens (CLS, SEP, PAD) and continuation subtokens were masked with `IGNORE_IDX` and excluded from loss computation.

3.6 Keyphrase extraction

Token-level tag assignment. At inference time, we converted the `IOB` class logits for token t in sentence j into a normalized score vector

$$\boldsymbol{\pi}_{j,t} = \text{softmax}(\boldsymbol{\ell}_{j,t}).$$

The predicted tag and its score were then defined separately as

$$\hat{y}_{j,t} = \arg\max_c \pi_{j,t,c}, \qquad q_{j,t} = \pi_{j,t,\hat{y}_{j,t}}.$$

For words tokenized into multiple subtokens, only the first subtoken's predicted tag $\hat{y}_{j,t}$ and score $q_{j,t}$ are retained during keyphrase extraction; subsequent subtokens of the same word are ignored (Ta-

[2]This was to match the dataset size used in (Martinc et al., 2025), where 80% of each validation split was used for fine-tuning and the remaining 20% for hyperparameter optimisation.

Model/Benchmark	kp20k		kptimes		inspec	
F1	@5	@10	@5	@10	@5	@10
Validation Set Fine-tuning						
SEKE (Martinc et al., 2025)	32.7	33.0	44.9	45.1	47.2	56.5
Ours (ModernBERT)	31.3	30.3	40.8	33.0	48.7	54.3
Training Set Fine-tuning						
SEKE (Martinc et al., 2025)	36.3	36.5	54.8	55.0	49.4	58.8
Ours (ModernBERT)	37.5	35.9	—	—	—	—

(a)

Metric	kp20k		kptimes		inspec	
	@5	@10	@5	@10	@5	@10
Validation Set Fine-tuning						
Precision	26.4	22.6	31.7	21.8	57.1	47.5
Recall	38.3	45.9	57.6	68.2	42.5	63.4
Training Set Fine-tuning						
Precision	31.9	27.0	—	—	—	—
Recall	45.5	53.9	—	—	—	—

(b)

Table 2: Performance by benchmark and fine-tuning regime: (a) F1 for our model and SEKE; (b) precision and recall for our model. SEKE scores are from Martinc et al. (2025, Tables 2 and 5); validation-set values are means over five runs and training-set values are from a single run.

ble 4). This step resulted in one `I/O/B` tag and associated score for each word in the sentence.

Keyphrase assembly and scoring. To assemble the multi-word keyphrases, we iterated over all words in the sentence and started a new keyphrase if the predicted tag was `B` with $q \geq 0.5$, and appended a word to the current keyphrase if its predicted tag was `I` with $q \geq 0.5$. Current keyphrase construction was terminated if a `B` or `I` tag had $q < 0.5$, or if an `O` tag was encountered. Each keyphrase was then assigned a score by averaging the scores of its constituent words.

3.7 Evaluation

For evaluation we largely followed Martinc et al. (2025). To ensure robustness against surface-form variations, we replaced hyphens (−), slashes (/), and asterisks ($\star$) with spaces, and lowercased and Porter-stemmed both gold and candidate keyphrases prior to evaluation. Candidate keyphrases were additionally deduplicated by retaining only the highest-scoring occurrence among duplicates. For each document d, we computed precision $P_d@k$ and recall $R_d@k$ from the top-k ranked candidates. If fewer than k candidates were available, precision was computed over the available candidates. We then averaged precision and recall separately across the D evaluated documents:

$$\overline{P}@k = \frac{1}{D}\sum_{d=1}^{D} P_d@k, \qquad \overline{R}@k = \frac{1}{D}\sum_{d=1}^{D} R_d@k.$$

The reported F1 score was their harmonic mean:

$$F_1@k = \frac{2\,\overline{P}@k\,\overline{R}@k}{\overline{P}@k + \overline{R}@k}.$$

This aggregation follows the SEKE evaluation implementation, ensuring direct comparability with its reported results. Gold keyphrases were taken as the union of those stem-matched within the document text, and documents with no remaining gold keyphrases were excluded from metric computation.

4 Results

4.1 Benchmark performance

General benchmark performance. Overall, our model performs comparably with the state-of-the-art SEKE model (MoE BiLSTM architecture) proposed by Martinc et al. (2025). When fine-tuned on the smaller validation sets of the kp20k and inspec benchmarks, our model performed within the margin of +/- 1–3 percentage points (Table 2(a)) relative to SEKE. When fine-tuned on the larger

training set of the kp20k benchmark, our model showed an F1@5 score of 37.5 (Table 2(a)) which is competitive with the reported SEKE score of 36.3. By far the largest performance divergence between the two models is seen on the kptimes benchmark where SEKE outperformed our model by 4.1 (F1@5) and 12.1 (F1@10) percentage points.

Precision. Precision ranged from 21.8% to 57.1% across the evaluated settings (Table 2(b)), indicating that many top-ranked candidates did not match the annotated gold keyphrases. Recall generally exceeded precision across the evaluated settings, consistent with a tendency to over-generate keyphrase candidates. Performance was weakest on the journalistic kptimes benchmark, whose documents are longer, have a larger vocabulary, and cover a broader range of topics than the scientific abstracts in kp20k and inspec (Table 1).

4.2 Characteristics of extracted keyphrases

Eager keyphrase extraction. We examined overgeneration by comparing the numbers of predicted and gold keyphrases in 100 randomly sampled test documents per benchmark. Figure 4 in the Appendix reports the median counts. Across all three benchmarks, the median predicted count substantially exceeded the median gold count, with the largest relative difference on kptimes.

Error analysis. With the model generating a large number of keyphrases, are the generated candidate keyphrases relevant or do they mostly represent irrelevant noise? In Table 5 in the Appendix, we show a sample of gold and extracted keyphrases from six documents in the test set stratified by document-specific F1 scores into high, mid, and low accuracy tiers. Each predicted keyphrase is marked using the surface-match categories described below.

Score distribution. To examine whether keyphrase scores distinguish correct from incorrect predictions, we analysed 100 randomly sampled kp20k test documents and all 784 keyphrase candidates produced for them by the model fine-tuned on the full kp20k training set. For this descriptive analysis, we classified a candidate as an exact surface-form match, a partial match if it shared at least one word with a gold keyphrase, and a non-match otherwise. These categories were used only for the error analysis; benchmark evaluation used normalized and stemmed exact matching. As Figure 2 shows, exact matches generally received higher scores: 18.5% scored 0.9 or lower, compared with 47.2% of partial matches and 55.5% of non-matches.

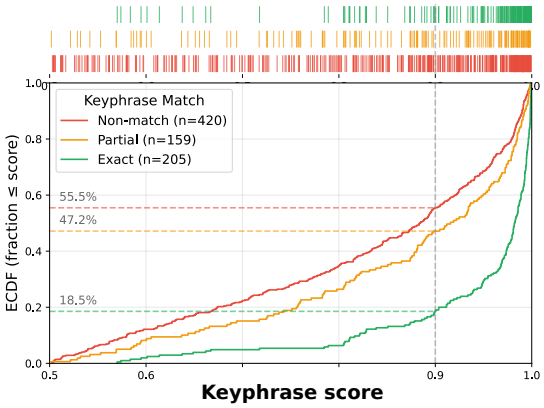


Figure 2: Keyphrase-score distributions by surface-match type. Curves show empirical cumulative distributions; dashed guides indicate the proportions with scores of 0.9 or lower, and rug marks represent individual predictions.

Partial matches Examples show that, apart from exact matches and non-matching candidates, the model frequently extracted one or more partial matches. For example, in a document with the gold keyphrase *conventional surfactant*, the model extracted partial matching candidates *surfactant* and *surfactant systems*. The general pattern of partial matches across all performance tiers is shown in Figure 3. Partial matches account for approximately 20% of candidates in the mid-performance tier and 25% in the low-performance tier. These candidates receive no credit under exact-match evaluation.

Acronym variants. Qualitative inspection also revealed candidates that were acronyms of longer gold keyphrases. Because the evaluation requires an exact match after normalization and stemming, these pairs receive no credit: the predicted acronym is counted as a false positive and the unretrieved gold keyphrase as a false negative. Our partial-match criterion also classifies such cases as non-matches because the acronym and its expanded form share no words. These cases illustrate that exact-match evaluation may understate the usefulness of some predictions, although we did not quantify their prevalence.

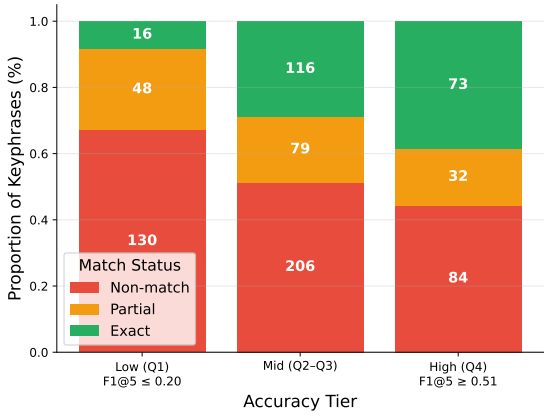


Figure 3: Keyphrase-candidate composition across performance tiers on the kp20k test set, based on per-document F1@5. Partial matches account for approximately 20% and 25% of candidates in the mid- and low-performance tiers, respectively. These candidates share at least one word with a gold keyphrase but receive no credit under exact-match evaluation. Numbers inside the bars indicate counts.

5 Discussion

In this paper, we presented an initial evaluation of a transformer-based keyphrase extraction model leveraging explicit fusion of local (sentence) and global (document) context. The complete architecture is smaller overall than the state-of-the-art comparison system developed by Martinc et al. (2025) (170.9M versus 274M parameters) and achieves comparable performance on the two scientific-domain benchmarks without its mixture-of-experts and BiLSTM components, although it performs substantially worse on the journalistic benchmark. This parameter-count difference distinguishes the architectures but does not by itself establish lower computational cost.

Despite the overall promising extraction performance, the model showed low precision and (over)eager extraction. It tended to generate large keyphrase candidate pools, especially on longer documents, leading to an increase in false positives. While we confirmed that the candidate pools often contained relevant tokens (e.g. substrings of target keyphrases), this is an area where further improvement is needed. One possible way to address this would be to tune the score threshold, which was fixed at 0.5 in our experiments, using held-out data.

More broadly, why train dedicated small language models in an era of large language models with increasing scale? Small language models (sLMs) fine-tuned for dedicated tasks can still outperform their larger generic pre-trained counterparts under certain settings (Martínez-Cruz et al., 2024). This is noteworthy especially considering the smaller computational footprint and greater potential for interpreting the models (Chen and Varoquaux, 2026). Because keyphrase extraction is a starting point for several downstream tasks like indexing, categorization, and summarization, ongoing development of language models that balance flexibility and computational footprint can outweigh the benefits of larger models.

6 Conclusion

We investigated a mixed-resolution context-fusion architecture that combines token-level information from the target sentence with sentence-level information from the surrounding context in a single ordered sequence. Across three benchmarks, the full model achieved performance comparable to SEKE, a recent state-of-the-art supervised system, on the two scientific-domain datasets, although a larger gap remained on the journalistic dataset. Notably, SEKE's overall parameter count is approximately 60% larger than that of our model. This competitive performance is consistent with the hypothesis that compressed document context can support keyphrase extraction, but it does not establish that context fusion itself improves performance because the architecture was evaluated only as a whole. Error analysis showed systematic over-generation, but many predictions counted as false positives under exact-match evaluation partially overlapped with gold keyphrases, and exact matches tended to receive higher scores than partial and non-matches. These findings identify candidate selection and phrase delimitation as promising targets for improvement. Controlled ablations and direct measurements of computational cost remain necessary to isolate the contribution of context fusion and assess its efficiency.

7 Limitations

Our empirical comparison is limited to SEKE (Martinc et al., 2025). Although SEKE reports comparisons with a broader set of supervised, unsupervised, and generative systems, reproducing those systems under our preprocessing and training protocol was beyond the scope of this study. Broader controlled comparisons are needed to establish the model's position relative to other approaches. Our evaluation focuses on three English-

language benchmarks spanning scientific and journalistic text. While this selection enables comparison across domains and document lengths, evaluation on additional domains and languages would provide a broader account of generalization. The reported scores are based on a single training run; repeated runs are needed to estimate variance and determine the reliability of small performance differences between models. Our model is smaller overall than SEKE (170.9M versus 274M parameters), although all 170.9M of its parameters are fine-tuned, compared with 45M trainable parameters reported for SEKE. Because total and trainable parameter counts reflect different computational costs, direct measurements of training cost, inference latency, and memory use would provide a more complete efficiency comparison. Finally, the present experiments evaluate the architecture as a whole. They establish the feasibility of combining token-level target representations with compressed document context, but do not isolate the effect of that context. Comparisons with a target-sentence-only model, together with controls that vary the context window or replace document context with unrelated sentences, are needed to determine whether and how document context affects extraction performance.

8 Ethical Considerations

This study uses publicly available benchmark datasets and involves no human participants or private data. Incorrectly extracted keyphrases could lead to misleading semantic indexing or retrieval, particularly if model outputs are used without human verification.

Acknowledgments

This work was supported by the Slovenian Research and Innovation Agency (ARIS) through the core research program Knowledge Technologies (P2-0103), the project Embeddings-based techniques for Media Monitoring Applications (EMMA; L2-50070), and the project Linguistic Accessibility of Social Assistance Rights in Slovenia (LASAR; J5-50169).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Lihu Chen and Gaël Varoquaux. 2026. What is the Role of Small Models in the LLM Era: A Survey. *arXiv preprint*. ArXiv:2409.06857 [cs.CL], v8.

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.

Ygor Gallina, Florian Boudin, and Beatrice Daille. 2019. KPTimes: A Large-Scale Dataset for Keyphrase Generation on News Documents. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 130–135, Tokyo, Japan. Association for Computational Linguistics.

Kelly Hong, Anton Troynikov, and Jeff Huber. 2025. Context Rot: How Increasing Input Tokens Impacts LLM Performance. Technical report, Chroma.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Mayank Kulkarni, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2022. Learning Rich Representation of Keyphrases from Text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 891–906, Seattle, United States. Association for Computational Linguistics.

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.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint*. ArXiv:2307.03172 [cs].

Matej Martinc, Thi Hong Hanh Tran, Senja Pollak, and Boshko Koloski. 2025. SEKE: Specialised Experts for Keyword Extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2025*,

pages 14191–14205, Suzhou, China. 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.

Roberto Martínez-Cruz, Debanjan Mahata, Alvaro J. López-López, and José Portela. 2025. Enhancing keyphrase extraction from long scientific documents using graph embeddings. *Applied Intelligence*, 55(10):711.

Roberto Martínez-Cruz, Alvaro J. López-López, and José Portela. 2024. ChatGPT vs state-of-the-art models: a benchmarking study in keyphrase generation task. *Applied Intelligence*, 55(1):50.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

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.

Tokala Yaswanth Sri Sai Santosh, Debarshi Kumar Sanyal, Plaban Kumar Bhowmick, and Partha Pratim Das. 2020. DAKE: Document-Level Attention for Keyphrase Extraction. In *Advances in Information Retrieval*, pages 392–401, Cham. Springer International Publishing.

Peter D. Turney. 2002. Learning to Extract Keyphrases from Text. *arXiv preprint*. ArXiv:cs/0212013.

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, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. _eprint: 2412.13663.

A Additional implementation details and analyses

Token	Label	Reason
`[CLS]`	`IGNORE`	Special token
`neural`	B	Starts keyphrase
`net`	I	Continuation (first subtoken)
`##works`	`IGNORE`	Continuation subtoken
`are`	O	Outside keyword
`powerful`	O	Outside keyword
`[SEP]`	`IGNORE`	Special token

Table 3: Example IOB labeling with subtoken splitting. Continuation subtokens (e.g. ##works) are ignored during training, but inherit the predicted label of the first subtoken during inference to allow word-level predictions.

Word	Subtokens	Subtoken tags (scores)	Decoded word tag
deep	deep	B (0.92)	B
learning	learn, ##ing	I (0.88), O (0.40)	I
models	models	I (0.85)	I
are	are	O (0.95)	O
weak	weak	B (0.40)	O ($q < 0.5$)

Table 4: Example of subtoken-to-word I/O/B tag assignment. For words split into multiple subtokens (learning $\rightarrow$ learn, ##ing), only the first subtoken's score and predicted tag are retained; continuation subtokens (##ing) are ignored. Predictions with $q < 0.5$ are treated as outside (O) during keyphrase generation.

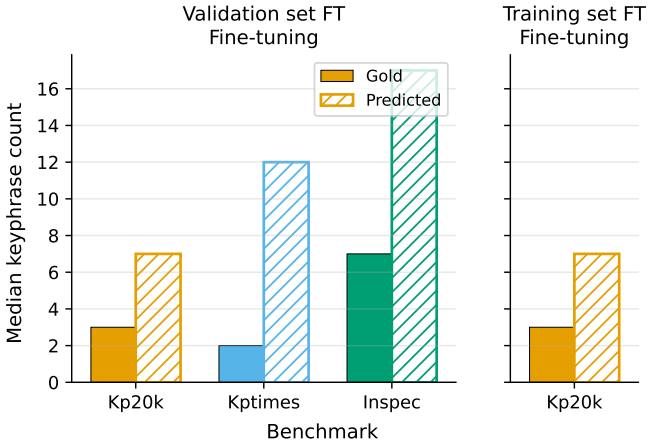


Figure 4: Median numbers of gold and predicted keyphrases across 100 randomly sampled test documents per benchmark. The model consistently predicts more keyphrases than are present in the gold annotations, ranging from approximately 2.3 times as many for kp20k and inspec to 6 times as many for kptimes.

ID	F1@10	Gold Keywords	Extracted Keywords
High Accuracy (in top quartile)			
1	66.7%	boundary element method, object oriented programming, subregions	boundary element method (1.00), object oriented programming (0.99), boundary integral equation methods (0.97)
2	40.0%	cellular structure, fill material, fill materials, retaining wall, shear resistance	distinct element method (1.00), shear resistance (0.99), retaining wall (0.99), displacement incremental analysis (0.98), cellular structure (0.97), distinct element method analysis (0.85), shear behavior (0.83), steel frame (0.81), friction (0.71), wall (0.59)
Mid Accuracy (in mid quartiles)			
3	30.8%	conventional surfactant, lattice monte carlo, mixed micellization, synergism	lattice monte carlo (0.99), surfactant (0.97), aqueous solution (0.96), mixed gemini (0.95), mixed micellization (0.94), surfactant systems (0.90), monte carlo simulation (0.87), gemini (0.83), simulation (0.62)
4	30.8%	deteriorating jobs, multitasking, tabu search	scheduling (1.00), tabu search (0.99), algorithms (0.99), deteriorating jobs (0.98), complexity (0.98), framework (0.96), multitasking environments (0.95), mathematical programming (0.95), lagrangian relaxation (0.94), optimal (0.93), search algorithm (0.81), planning (0.79)
Low Accuracy (in bottom quartiles)			
5	22.2%	wireless sensor networks	adapt (0.98), interpreted (0.97), adaptlets (0.97), wireless (0.93), customization (0.92), wireless sensor networks (0.83), performance (0.82), sensor networks (0.78)
6	0.0%	bareiss algorithm, bulk synchronous parallel, toeplitz system, toeplitz systems	bsp (0.97), load balance (0.97), systolic array (0.95), scheduling (0.92), toeplitz (0.90), data distribution (0.78), speedup (0.78)

Table 5: Examples of ground-truth and extracted keyphrases for six randomly sampled documents from the kp20k test set stratified by document-level accuracy. Color legend: Exact match, Partial match (dashed border). Numbers in parentheses indicate model scores.