Words That Stick: Using Keyword Cohesion to Improve Text Segmentation

Amit Maraj
Ontario Tech University
2000 Simcoe St N., Oshawa, ON
amit.maraj@ontariotechu.net

Miguel Vargas Martin
Ontario Tech University
2000 Simcoe St N., Oshawa, ON
miguel.martin@ontariotechu.ca

Masoud Makrehchi
Ontario Tech University
2000 Simcoe St N., Oshawa, ON
masoud.makrehchi@ontariotechu.ca

Abstract

Text Segmentation (TS) is the task of segmenting bodies of text into coherent blocks, mostly defined by the topics each segment contains. Historically, techniques in this area have been unsupervised, with more success recently coming from supervised methods instead. Although these approaches see better performance, they require training data and upfront training time. We propose a new method called Coherence, where we use sentence embeddings to pull representational keywords as the main constructor of sentences when comparing them to one another. Additionally, we include a storage of previously found keywords for the purposes of creating a more accurate segment representation instead of just the immediate sentence in question. We show improved results over current state-of-the-art unsupervised techniques when analyzed using P_k and WindowDiff scores. Coherence also requires no fine-tuning.

1 Introduction

We present Coherence, a method that utilizes related words and their contextual meanings within sentences for effective Text Segmentation (TS). In the past decade, advancements in the field of TS have been primarily dominated by supervised techniques (Badjatiya et al. (2018), Koshorek et al. (2018), Somasundaran et al. (2020), Barrow et al. (2020), Lo et al. (2021), and Inan et al. (2022)), which require training data and are computed on a sentence-wise basis (i.e., each sentence is compared with adjacent sentences for evaluation). In contrast, Coherence uses contextual keyword embeddings for comparison, reducing potential noise and unnecessary sentence-level information that may not be helpful to the TS task.

Coherence uses a sliding window technique, traditionally used in supervised TS, to predict segment breaks (e.g., $P(S_{n-1}, S_n, S_{n+1}) = 1$). However, Coherence enhances this method by incorporating contextual information (through contextual keyword embeddings).

Coherence demonstrates performance improvements, particularly in P_k scores, and does not require fine-tuning. By leveraging pre-trained sentence encoders like BERT, LaBSE, and S-BERT, Coherence leverages extracted keywords to form an end-to-end flow. The core of Coherence lies in collecting and utilizing important keywords during the segmentation process. These keywords are represented as contextual embeddings, capturing essential information about their usage within sentences (for example, differentiating "bridge" in the context of crossing a river from "bridge" in the context of a human's nose). This process is inspired by the multi-headed attention mechanism in the Transformer architecture, providing a nuanced understanding of sentence relationships without the need for extensive training and data.

1. A novel approach to unsupervised TS that achieves state-of-the-art (SOTA) results on a variety of diverse and widely accepted TS datasets in the research community.
    - Coherence does not require fine-tuning and is shown to perform competitively and even outperform current SOTA unsupervised systems in some benchmarks.
    - Using pre-trained sentence embeddings, Coherence leans on both similar and diverse keywords to create more orthogonality in representations of sentences.
2. A keyword collection mechanism called Keyword Map, which creates segment representations through its most important keywords.
    - The Keyword Map stores important sentence-based representations through contextual keywords for later reference during comparison.

Proceedings of the 28th Conference on Computational Natural Language Learning, pages 1–9
November 15-16, 2024

3. An approach to unsupervised TS that has explainability in the prediction process, through the extraction of important keywords.

We show that without the need for expensive fine-tuning and highly-dimensional sentence embeddings as training data, we can achieve performance improvements in a space that has been more recently dominated by advancements in supervised learning. Using orthogonal keywords in addition to similar keywords provides more breadth in keyword representation to further bolster results. All our code can be found on the Human-Machine Lab GitHub Repository [1].

2 Related Works

Initially, Hearst (1997) introduced TextTiling, an unsupervised algorithm that identifies segment boundaries through lexical overlaps. Similarly, Choi (2000) demonstrated the efficacy of unsupervised methods by analyzing sentence similarities, categorizing their work within linear TS methodologies. These initial contributions set a new standard in the field.

The landscape of TS shifted with the advent of advanced word and sentence embeddings, paving the way for supervised techniques. Koshorek et al. (2018) explored the potential of processing large TS datasets through a Bi-LSTM, analyzing three sentences at a time to understand their interrelations. Building on this, Badjatiya et al. (2018) proposed a sentence-wise model utilizing attention mechanisms to enhance performance further. Recent supervised approaches have increasingly incorporated LSTMs and Transformers as foundational components, as seen in works by Somasundaran et al. (2020), Barrow et al. (2020), Lo et al. (2021), and Inan et al. (2022). These studies have showcased the effectiveness of adding topic information and emphasizing sentence contextuality in achieving top-tier results.

Despite the dominance of supervised models, unsupervised TS techniques continue to show promise. Misra et al. (2009) revisited the classic TextTiling approach, refining it with LDA to identify more precise keywords. Riedl and Biemann (2012) combined LDA and TextTiling for another innovative unsupervised solution. Furthermore, Glavaš et al. (2016) introduced a novel unsupervised graph-based method, analyzing sentences as nodes within a graph to predict segment boundaries. These unsupervised models underscore the ongoing exploration and diversity in TS methodologies. While unsupervised approaches in the field continue to be important due to their flexibility and lack of need for domain-specific training data, more research has recently focused on supervised approaches. Fragkou et al. (2004)'s approach to TS relied upon within-segment word similarity and prior information about segment length, but does not incorporate inter-sentence comparisons. In contrast, Brants et al. (2002) approaches unsupervised TS by using Probabilistic Latent Semantic Analysis (PLSA) to identify similar words an in inter-sentence level. They then apply a TextTiling based approach for identifying changes in frequency between sentences.

Another technique by Solbiati et al. (2021) takes a unique approach to unsupervised TS by grouping a series of sentences together, stacking them on top of each other, and performing max pooling. The resulting matrix is a mixture of sentences, which can then be used to compare to other matrices. They perform their analysis on meeting data, which shows improvements upon other techniques. More recently, John et al. (John et al., 2017) utilize an LDA-based TextTiling approach that produces strong results. The boundary adjustment technique proposed in this work is a retroactive solution to TopicTiling (Riedl and Biemann, 2012) that helps improve results.

3 Methodology

The core of Coherence is its ability to pull out keywords from provided sentences. To accomplish this, we use a library called KeyBERT [2]. This library goes through each word in a sentence, creates an embedding for the word and compares it with the embedding of the sentence at hand. Keywords are identified as the ones with a higher similarity to the sentence embedding. Because of this, there is no need to globally scan the document beforehand, which other techniques like TF-IDF and LDA require. Utilizing BERT allows the KeyBERT library to effectively look into the attention being paid at every word and phrase to identify important words. KeyBERT has been shown to outperform other topic modelling and keyword extraction techniques like LDA and YAKE (Campos et al., 2020).

We also consider the use of an LDA based ap-

[1] https://github.com/HumanMachineLab/Coherence

[2] https://github.com/MaartenGr/KeyBERT

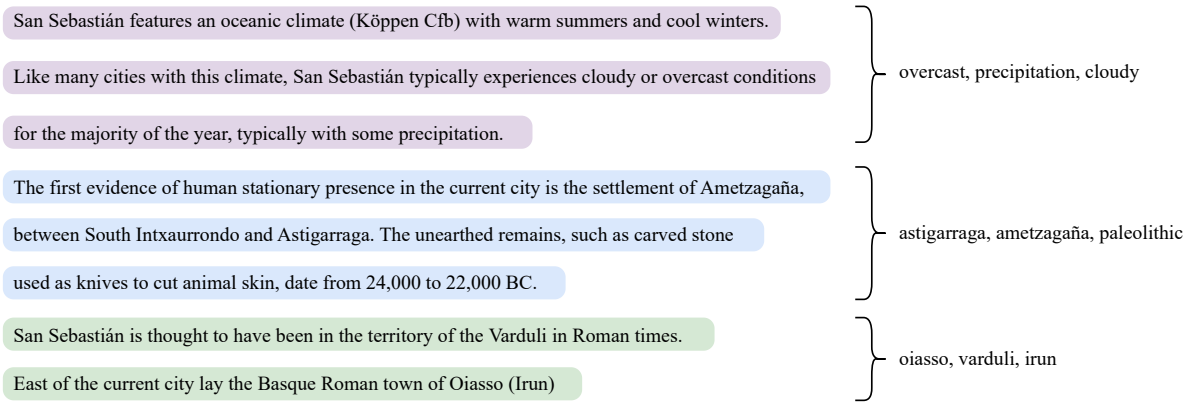

Figure 1: Topics gathered throughout the keyword extraction phase within the "wiki" dataset. Due to the natural pruning of the Keyword Map, only the most pertinent topics are retained. Additionally, importance of the keyword to its original sentence is also maintained. Results shown here are extracted from the "wiki" dataset starting at sample 643.

proach, such as BERTopic [3] as the keyword extractor, but elect to stick with KeyBERT due to the following advantages:

- Pulling keywords using KeyBERT does not require upfront training, whereas BERTopic does.
- Because BERTopic uses LDA to pull topics, the requirement to be aware of the entire document's worth of text beforehand increases processing time and reduces flexibility.
- Inherently, LDA does not use word embeddings to pull important topics, which means that extracted words are in the form of text. Since our technique compares words in vector space, constructing embeddings for extracted words will not retain sentence contextuality.

Coherence is broken down into two major phases. We use sentences to denote the important keywords derived from said sentences - sentences are not compared verbatim, rather the keywords that make up the sentence are compared. At every step, the current sentence is compared against prior sentences, as long as they exist in the Keyword Map (we elaborate on conditions where a sentence's keywords would not end up in the Keyword Map later in this section).

3.1 Keyword Extraction Phase

In this phase, we use KeyBERT to extract important keywords from sentences at each iteration. KeyBERT uses BERT or any BERT-based model (e.g., RoBERTa, DistilBERT, ALBERT, etc.) to create a representation of each sentence. It then takes the sentence embedding and each respective word embedding (as provided by BERT as well). The higher the similarity between the word and the sentence, the more likely it is considered a keyword. After extraction, keywords are sorted in descending order based on its importance to its parent sentence. This importance is calculated based on how strong the keyword is in similarity to its parent sentence.

3.2 Prediction Phase

In this phase, we use information from previous sentences in the segment to compare with the keyword representation of the current sentence.

Keyword Map. We first create a representation of the current segment through storage of keywords gathered throughout the iteration process. We determine the top n keywords that should be stored per sentence and save them in a map. We then use this map to compare against the current sentence during iteration. For example, when we store 3 keywords in the map per sentence and we are on the fifth sentence in the segment, we will compare the current sentence to 12 keywords in the map (3 keywords times 4 previous sentences).

When storing keywords in the map, we compare the current sentence's keywords to the pre-existing keywords in the map. We take the most similar (with respect to the pre-existing keywords in the map) keywords in the current sentence and store it in the map. This allows us to build a Keyword Map that is representative of the overall topics within the segment. After k (average length of segment size in the dataset) sentences, we prune the Keyword

[3] https://github.com/MaartenGr/BERTopic

Map at every step by removing the oldest set of keywords, adopting a queue-based FIFO structure.

Comparison. The current sentence's keywords are compared to the previous sentence's keywords and every keyword in the Keyword Map. All the comparisons are summed and then averaged to get an overall similarity score. This similarity score, which is calculated as shown in Formula 2, ends up being a representation of how cohesive the current sentence is with all the sentences previous to it. Words in earlier sentences of the segment are also de-emphasized so they do not hold as much weight in the comparison as words that are closer to the current sentence. A value of $1/distance(curr\_sent, prev\_sent)$ is applied to all words in the previous sentence. For example, a word embedding belonging to a sentence that occurred 2 sentences prior will have a weight of $1/2$ applied to it.

The output from the prediction phase is a logit that is the average of all the comparisons between the current sentences and every sentence in the Keyword Map, which can be seen in Figure 3.

As shown in Figure 3, the Keyword Map is built throughout the inference process. This map acts as a representation of the segment currently being scanned. Because important segment-based information can exist in more places than the current sentence, the Keyword Map builds a representation of keywords found earlier in the segment.

During the prediction process, the contents of the Keyword Map along with the current sentence's keywords in the sliding window are compared using cosine similarity and an average. If the contents of the Keyword Map and current sentence are dissimilar enough (based on a parameter– $prediction\_threshold$), the system predicts a one, indicating that the second sentence is the start of a new segment. Upon a positive prediction, the Keyword Map gets emptied so it can begin collecting new keywords. If the Keyword Map and current sentence are similar, the system predicts a zero and continues to build the Keyword Map. To avoid the Keyword Map becoming too large over time, especially with longer segments, it is pruned after n size (e.g., if the Keyword Map has five sentences worth of keywords and we add another sentence worth of keywords, we remove the oldest sentence). For example, we prune the map after it grows to 26 sentences (the average segment size) for the Clinical dataset (Malioutov, 2006).

Values for the $prediction\_threshold$ are tested between zero and one at every tenth interval and notice that the lowest P_k and WindowDiff scores consistently show up when 0.5 is used.

4 Metrics

Two popular metrics that exist solely to benchmark TS systems are P_k and WindowDiff (WD), which have become commonplace for work in the TS field. P_k is the probability that a pair of chosen sentences with a distance of k are incorrectly classified. Both the WD and P_k metrics use a sliding window of fixed size w over the document and compare the predicted segments with the reference ones. k is determined as half of the average true segment size of the document. Since P_k and WD are both penalty metrics, lower values indicate better performance. While P_k is the most widely and still is the most accepted metric in the TS space, WD was originally proposed as an update to the P_k metric. P_k can be thought of as the probability that two segments drawn from a document are incorrectly identified as belonging to the same segment. WD operates almost identically, but uses a sliding window to penalize systems that tend to overpredict, resulting in false positives - something that P_k does not acknowledge as an errant prediction. Both P_k and WD thus lie between zero and one and an algorithm that assigns all boundaries correctly receives a score of zero. WD is considered a better measure than P_k as the P_k metric suffers from issues such as a lack of false positive prediction penalization (Pevzner and Hearst, 2002).

5 Data

Unsupervised TS methods are often evaluated using constructed datasets, which amalgamate segments from varied sources into composite documents, as evidenced by studies from Choi (Choi, 2000) and Galley et al (Galley et al., 2003).

5.1 Choi Dataset:

Introduced by Choi (2000) in 2000, this dataset has become a staple for TS research, referenced in works by Misra et al. (2009), Brants et al. (2002), Fragkou et al. (2004), Glavaš et al. (2016), Sun et al. (2008), and Galley et al. (2003). It is crafted from the Brown corpus, containing 700 documents that simulate real text structure. The compilation includes 400 documents with segments varying from 3-11 sentences, alongside 100 documents for each segment length category: 3-5, 6-8,

$$Coherence(S_{n-1}, S_n) = \frac{1}{x}\sum_{j=1}^{x}\frac{1}{x}\sum_{i=1}^{x} cos(u_j(S_{n-1}), w_i(S_n))$$

Formula 2: The similarity calculation between two sentences, where each keyword in the respective sentence is compared with every other keyword in the comparing set of keywords (e.g., the set of w keywords are gathered from S_n and the set of u keywords are gathered from S_{n-1}), where w and u are keywords. Each line indicates a cosine similarity calculation and once all the calculations are done from a keyword on the left to all keywords on the right, they are summed and averaged. This process continues for all the keywords and the total average is taken. Additionally, each keyword (w of S_n) has a weighting applied to it, indicating its importance to the sentence it was derived from originally.

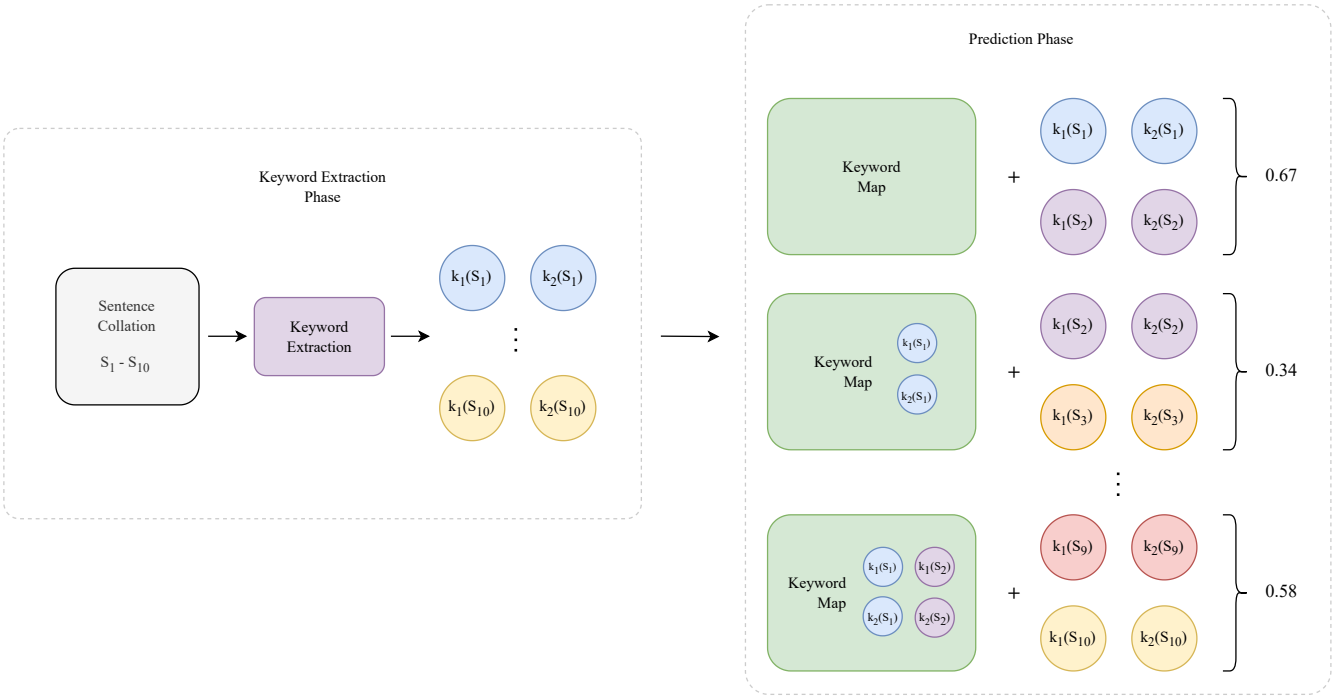

Figure 3: Architecture for Coherence. Keywords extracted from in the extraction phase are passed toward the prediction phase and stored in the Keyword Map. The current sentence's keywords are derived from current and previous sentences and are denoted with h, i, j, k, and l. The output from the prediction phase is a logit that is representative of the cohesion between the current sentence's keywords and the keywords compared to from the Keyword Map.

and 9-11 sentences.

5.2 Manifesto Dataset:

To complement the synthetic Choi dataset, Coherence's effectiveness is also tested on real political texts from the Manifesto Project dataset. This collection of documents has been meticulously segmented into seven topics, such as economy and welfare, and foreign affairs, by field experts. The curation of this dataset is attributed to Glavaš et al. (2016).

5.3 Clinical Dataset:

We also use the Clinical dataset put together by Malioutov (2006) to showcase our results. This dataset consists of a set of 227 chapters from a medical textbook. Each chapter is marked into sections indicated by the author which forms the segmentation boundaries. It contains a total of 1136 sections.

5.4 Fiction Dataset:

To include even more diversity in our results, we also showcase our results on the Fiction dataset put together by Kazantseva and Szpakowicz (2011), which is a collection of 85 fiction books downloaded from Project Gutenberg. Segmentation boundaries are the chapter breaks in each of the books.

5.5 Wiki Dataset:

Finally, we test Coherence's performance on a curated Wikipedia dataset, introduced by Badjatiya et al. (2018), is also presented. This dataset consists of randomly selected set of 300 documents having an average segment size of 26. The documents widely fall under the narrative category.

6 Results

Coherence shows and improvement over SOTA unsupervised results in the space. Results are reported on the Choi, Manifesto, Clinical, Fiction, and Wiki Datasets (Choi (2000), Glavaš et al. (2016), Malioutov (2006), Kazantseva and Szpakowicz (2011), Badjatiya et al. (2018)). Our performance on these datasets shows the versatility of Coherence. This gives us hope that with the use of pre-trained models, unsupervised approaches can prove to be viable in the TS space. Results on the Choi and Manifesto datasets are reported against pre-existing SOTA unsupervised TS approaches. Results for the Clinical, Wiki, and Fiction datasets are compared against Badjatiya et al. (2018)'s work.

Results on the Clinical and Fiction datasets are competitive with Badjatiya et al. (2018)'s pre-existing supervised approach. Coherence does not do as well on the Wiki dataset however. We believe this is due to the subjectivity in TS datasets at the labelling level. The Wiki dataset has an average segment length of 26 sentences for example. On Choi's dataset, Coherence performs extremely well, outperforming all previous SOTA unsupervised TS techniques. Coherence also performs competitively, with stronger results in the WD metric on the Manifesto dataset. This performance improvement on WD versus P_k indicates that Coherence makes less false positive predictions than pre-existing techniques.

We show that, in comparison to previous SOTA unsupervised techniques, Coherence outperforms in a variety of datasets using both the P_k and WD metrics as benchmarks. This comes without the need for fine-tuning or domain adaptation. Since the keyword extraction phase of Coherence is modular, we believe that as sentence and word embedding technology continues to improve, so will the results of Coherence.

The lack of need for fine-tuning a model is advantageous and as of such, each round of inference takes roughly 25ms - 125ms on a cloud-based A100 GPU. Additionally, Coherence provides utility without the need for training or domain adaptation. The lightweight lift of Coherence allows it to be used against various datasets, due to the strength of the sentence encoder. The applicability of Coherence to new and unseen test datasets can prove to be useful in production settings.

7 Limitations

Coherence shows improvements over pre-existing SOTA unsupervised systems such as TopicTiling (Riedl and Biemann, 2012) and GraphSeg (Glavaš et al., 2016).

The authors for the works found in Table 2 do not present their findings using the same metrics, nor do they provide their codebase, and due to resource limitations, we are not able to replicate their works to evaluate and report on WD. We acknowledge that this is a limitation of our work, but we also illustrate the strengths and improvements of our system using a wide array of available datasets.

Some reliance for Coherence comes from the pre-trained sentence encoder (KeyBERT) in the keyword extraction phase. Although this seems like a limitation, it can be a strength in the flexibility of the system. Future iterations of pre-trained sentence encoders can be used to replace KeyBERT and enhance Coherence's output. We show the flexibility of our system by achieving superior results on a wide array of available datasets without the need for tedious fine-tuning. This implies that as keyword extraction techniques become stronger, so shall our system.

Most of the processing time comes from the keyword extraction phase, due to the keyword extraction library. Roughly 90% of this time comes from the keyword extraction phase, whereby KeyBERT needs to compare every keyword with its parent sentence embedding. The majority of the RAM utilization also comes from this phase, as the embedding model (LaBSE in our case) is loaded into memory for inference. In our experiments, Coherence required less than 3GB RAM throughout testing. With techniques like quantization, smaller models can perform this keyword extraction step more efficiently. This limitation is due to the selected keyword extraction library; KeyBERT in our case. The majority of processing time in the KeyBERT library comes from creating contextual embeddings for each sentence before comparing each word in the sentence it was pulled from with the sentence itself.

8 Conclusion

In this work, we present Coherence, which is a novel approach to unsupervised TS that leverages contextual keywords from sentences to represent text segments. We show that the emphasis on contextual keywords can build representations of

	Clinical		Wiki		Fiction	
	$P_k \downarrow$	WD $\downarrow$	$P_k \downarrow$	WD $\downarrow$	$P_k \downarrow$	WD $\downarrow$
Badjatiya et al. (2018)	**33.0**	**31.0**	**34.0**	**32.0**	38.0	**31.0**
Coherence	37.1	38.9	50.2	53.4	**35.7**	61.6

Table 1: Results on the "clinical", "wiki", and "fiction" datasets (Badjatiya et al., 2018; Malioutov, 2006; Kazantseva and Szpakowicz, 2011). We compare our results to Badjatiya's fine-tuned neural model and show competitive results, without the need for fine-tuning.

	3 – 5		6 – 8		9 – 11		3 – 11	
	$P_k \downarrow$	WD $\downarrow$	$P_k \downarrow$	WD $\downarrow$	$P_k \downarrow$	WD $\downarrow$	$P_k \downarrow$	WD $\downarrow$
Choi (2000)	12.0	–	9.0	–	9.0	–	12.0	–
Brants et al. (2002)	7.4	–	8.0	–	6.8	–	10.7	–
Fragkou et al. (2004)	5.5	–	3.0	–	1.3	–	7.0	–
Misra et al. (2009)	23.0	–	15.8	–	14.4	–	16.1	–
Glavaš et al. (2016)	5.6	8.7	7.2	9.4	6.6	9.6	7.2	9.0
Coherence	**4.4**	**6.2**	**3.1**	**3.3**	**2.5**	**2.6**	**4.0**	**4.4**

Table 2: Results on the synthetic Choi (Choi, 2000) dataset.

	$P_k \downarrow$	WD $\downarrow$
Riedl and Biemann (2012)	33.39	38.31
Glavaš et al. (2016)	**28.09**	34.04
Coherence	31.71	**33.42**

Table 3: Results on the Manifesto (Glavaš et al., 2016) Dataset. We show the versatility of Coherence providing competitive results in a different domain.

segments, which can be used for TS. Coherence demonstrates improvements over SOTA unsupervised TS techniques, particularly in the metrics of P_k and WindowDiff. The main contributions of Coherence include the diverse extraction of keywords and an efficient keyword collection mechanism which we termed Keyword Map.

Our results on the Choi, Manifesto, Clinical, Wiki, and Fiction datasets show that Coherence can perform well in a variety of domains. While we also include our results on the newer WikiSection dataset, other supervised TS approaches show superior results.

Future work will focus on enhancing Coherence to consider the contextual relationship between extracted keywords, while exploring the method's applicability across various domains and datasets. Coherence offers a solution that can be adapted to various domains without the need for fine-tuning.

Acknowledgments

The first and second authors acknowledge the support of an NSERC Discovery Development Grant (DDG-2024-00031).

References

Pinkesh Badjatiya, Litton J Kurisinkel, Manish Gupta, and Vasudeva Varma. 2018. Attention-based neural text segmentation. In *European Conference on Information Retrieval*, pages 180–193. Springer.

Joe Barrow, Rajiv Jain, Vlad Morariu, Varun Manjunatha, Douglas W Oard, and Philip Resnik. 2020. A joint model for document segmentation and segment labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 313–322.

Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. 2002. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of the eleventh international conference on information and knowledge management*, pages 211–218.

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.

Freddy YY Choi. 2000. Advances in domain independent linear text segmentation. *arXiv preprint cs/0003083*.

Pavlina Fragkou, Vassilios Petridis, and Ath Kehagias. 2004. A dynamic programming algorithm for linear text segmentation. *Journal of Intelligent Information Systems*, 23(2):179–197.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 562–569.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 125–130. Association for Computational Linguistics.

Marti A Hearst. 1997. Text tiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

Hakan Inan, Rashi Rungta, and Yashar Mehdad. 2022. Structured summarization: Unified text segmentation and segment labeling as a generation task. *arXiv preprint arXiv:2209.13759*.

Adebayo Kolawole John, Luigi Di Caro, and Guido Boella. 2017. Text segmentation with topic modeling and entity coherence. In *Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016)*, pages 175–185. Springer.

Anna Kazantseva and Stan Szpakowicz. 2011. Linear text segmentation using affinity propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 284–293.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. *arXiv preprint arXiv:1803.09337*.

Kelvin Lo, Yuan Jin, Weicong Tan, Ming Liu, Lan Du, and Wray Buntine. 2021. Transformer over pre-trained transformer for neural text segmentation with enhanced topic coherence. *arXiv preprint arXiv:2110.07160*.

Igor Igor Mikhailovich Malioutov. 2006. *Minimum cut model for spoken lecture segmentation*. Ph.D. thesis, Massachusetts Institute of Technology.

Hemant Misra, François Yvon, Joemon M Jose, and Olivier Cappé. 2009. Text segmentation via topic modeling: an analytical study. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1553–1556.

Lev Pevzner and Marti A Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Martin Riedl and Chris Biemann. 2012. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 student research workshop*, pages 37–42.

Alessandro Solbiati, Kevin Heffernan, Georgios Damaskinos, Shivani Poddar, Shubham Modi, and Jacques Cali. 2021. Unsupervised topic segmentation of meetings with bert embeddings. *arXiv preprint arXiv:2106.12978*.

Swapna Somasundaran et al. 2020. Two-level transformer and auxiliary coherence modeling for improved text segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7797–7804.

Qi Sun, Runxin Li, Dingsheng Luo, and Xihong Wu. 2008. Text segmentation with lda-based fisher kernel. In *Proceedings of ACL-08: HLT, Short Papers*, pages 269–272.

A Appendix

Algorithm 1: Coherence

Result: Extract similar and diverse keywords with globally informed context through sentence batching.

```
keywords ← keyword_extraction([s0, ..., s9]) ;      /* s0, ..., s9 are sentences. */
keyword_map = [];
similarities = [];
predictions = [];
for i ... len(keywords) do
    curr_kws ← keywords[i + 1];
    prev_kws ← keyword_map[0 ... i];
    for w ∈ curr_kws do
        for k ∈ prev_kws do
            similarity ← cosine_similarity(k, w);
            similarities.insert(similarity);
            if similarity >= coherence_threshold then
                gits_map.insert(w) ;                    /* Add new keyword to map. */
            end
        end
    end
    if avg(similarities) >= coherence_threshold then
        predictions.insert(0) ;                 /* The current sentence is similar */
    else
        predictions.insert(1) ;             /* The current sentence is not similar */
    end
end
return predictions
```

Description: $coherence\_threshold$ is a hyperparameter set between 0 and 1, which can be used to enforce the strength keywords need to have between each other for entrance into the Keyword Map. Through our testing, we notice that this value will vary depending on the sentence encoder used (LaBSE in our case), since the strength of each keyword and its parent sentence are directly related to the encoder. To that end, we find the best results (based on P_k and WindowDiff scores) when this value is set to 0.7.