Topic Segmentation Using Generative Language Models

Pierre Mackenzie [1] **Maya Shah** [1]
[1] Adarga, London

Abstract

Topic segmentation using generative Large Language Models (LLMs) remains relatively unexplored. Previous methods use lexical or semantic similarity between parts of a document to decide on boundaries, but they lack the long range dependency, vast knowledge contained and end-to-end prediction availble through prompting LLMs. Here, we propose a new prompting strategy and compare to semantic similarity-based methods. We also support the adoption of a less commonly used evaluation metric: Boundary Similarity. Results show that LLMs can be more effective segmenters than existing methods, but issues remain to be solved before they can be relied upon for topic segmentation.

1. Introduction

1.1. Topic Segmentation Task Definition

Topic segmentation is the problem of dividing a string of text into constituent parts or 'segments'. Each segment should be semantically self-contained such that it is about one thing. The precise definition of a segment should be dependent upon the specific use cases. For this work, segment boundaries shall always lie on sentence boundaries.

We can interpret segmentation as a binary classification task. Given a list of input sentences S, of length n, the model must decide whether there exists a segment boundary between each pair of adjacent sentences. There are $n-1$ possible boundaries, and therefore our solution space is 2^{n-1}. Formally, the model must find a mapping f from the list of sentences to a binary vector of length $n-1$: $f(S) = \mathbf{y}$ where $\mathbf{y} = \{y_1, y_2 \ldots, y_{n+1}\}$ and each of the $y_i \in \{0, 1\}$.

Relative to generative tasks such as summarisation, the space of possible solutions is much smaller, but the problem remains subjective as where a boundary should lie can be ambiguous. Frequently, humans cannot agree on a correct solution (Hearst, 1997).

1.2. Motivation

Topic segmentation can be important as a processing step before some other NLP task, or can be important in its own right.

Tasks such as information retrieval, long-document summarisation and classification can all benefit from first being broken down by topic. For many open source or resource-constrained models, context windows limit the size of input for such tasks which can be constraining, especially for smaller models (Chung et al., 2022). Although context windows can be increased (Chen et al., 2023) and newer models are consistently increasing context lengths with ever-more-powerful GPUs, this alone does not fix all problems as LLMs do not fully utilise long context windows (Liu et al., 2023) (Petroni et al., 2020).

Furthermore, segmentation can be imporant for its own sake. One might use segments to generate a contents page for a long document, individual summaries of segments within a document or generate titles for each segment. Alternatively, one might use segmentation to break down a long document into smaller, more manageable parts for a user to read, or as a citation for RAG in which the model must generate an answer to a question based on a segment of text.

Our application at Adarga is to use topic segmentation as both a pre-processing step for a document understanding pipeline and as an important step in its own right. This pipeline is used to extract structured information from unstructured text, and the segmentation step is used to break down the document into smaller, more manageable parts. When presented to a user, only semantically self-contained sections of a long document are presented to a user, which can be more easily understood and acted upon.

1.3. Related Work

(Xing, 2024) provides a recent, broad overview of topic segmentation. Their work discusses the history of segmentation, dialogue/hierarchical/multi-modal segmentation, as well as exploration of large language models for topic segmentation.

Methods prior to the era of neural networks often relied on lexical similarity metrics. An influential framework introduced by (Hearst, 1997) in their paper *TextTiling* involves computing lexical similarity scores between adjacent sentences before boundaries are placed where similarity scores are lowest. A variety of such lexical similarity metrics were proposed (Galley et al., 2003; Eisenstein, 2009), but were proven to be too superficial compared to semanic similarity that came after it. Such a framework is still in use today but with semantic similarity metrics generated by embeddings from language models like SentenceBERT (Reimers & Gurevych, 2019).

Neural networks have seen much use for topic segmentation. The task can be conceived as a sequence labelling task and as such, BiLSTMs have been employed for this purpose (Wang et al., 2016) as well as attention-based methods (Lukasik et al., 2020). Many methods leverage a hierarchical framework in which one model extracts sequential features from the text, and another model constructs sentence boundaries based on these features. These models have also been LSTMs (Koshorek et al., 2018; Badjatiya et al., 2018), or more recently transformers (Glavaš & Somasundaran, 2020; Lo et al., 2021). Of these, the most recent a large pre-trained transformer for feature extraction, followed by a transformers classifier. However, these methods may not take full advantage of rich representations contained in the largest models as there is a bottleneck introduced by the hierarchical structure. We propose that an end-to-end method may be more effective.

LLMs have been the state of the art for a variety of NLP tasks including translation, summarisation and information extraction for a number of years now. However, there has been little research into the usage of LLMs for topic segmentation. A loss-based approach was proposed by (Feng et al., 2021) for annotation of dialogues and then extended by (Xing, 2024). This approach computes the mean NLL (Negative Log Likelihood) token-wise loss of a pre-trained model when predicting the tokens in a sentence. Boundaries are then placed where this loss is above some threshold, indicating that the sentence was hard to predict. While this approach is almost directly able to leverage the vast knowledge contained in LLMs and a tune-able threshold is appealing, it is also an arbitrary threshold, and there remains a strong inductive bias that the loss will be higher at segment boundaries, even in nuanced cases.

The only work we found to explore the use of directly prompting an LLM for topic segmentation was (Xing, 2024), whose paper on the subject is currently under review. The find that prompting ChatGPT is the best dialogue segmentation model unless the input exceeds ChatGPT's limit. They propose two prompting methods: one which asks the LLM to return the original model with special characters delimiting the segment boundary, and one in which the LLM is asked to return pair-wise semantic coherence scores in the range of 0 to 1 for adjacent sentences. The first method falls short of our requirements as there is no guarantee that the model will return the original document unedited, and is wasteful of tokens. The second method is methodologically flawed as the returned scores have no guarantee of directly corresponding to semantic coherence *for topic segmentation*, are very hard to interpret and there will still be a bottleneck in turning scores into segment boundaries. We propose a new prompting method which we believe is more effective and more interpretable, less wasteful of tokens, and not limited by the maximum token limit of the model.

In this work, we compare our new prompting work to our own existing method which uses SentenceBERT embeddings and cosine similarity to decide on segment boundaries. We do not compare to other prompting methods nor methods based on hierarchites of transformers due to resource constraints. However, our focus is on whether our new prompting strategy is a feasible replacement for methods which used semantic similarity. Further work should more thoroughly compare our new prompting method to other prompting schema, to loss-based approaches and to other hierarchical methods.

2. Method

2.1. Datasets

There are 4 types of datasets used in the experiments in this work: a small human-annotated dataset, a scrape of English wikipedia, a 'concatenated' wikipedia scrape and a synthetic GPT3.5 generated dataset.

2.1.1. Human-Annotated Dataset

Lacking the resources to create a large dataset annotated by humans, this work uses a very small manually segmented dataset of 10 documents. These documents are a mix of news articles, wikipedia articles, and miscellaneous documents such as podcast transcripts and scientific reports. This was intended to represent varying difficulties of segmentation, and provide examples which could be manually inspected to interpret segmenter behaviour. Only one annotator was used, and the segments were created by hand. Further work might involve multiple annotators on a much larger dataset, with a more rigorous process to ensure consistency and quality.

2.1.2. Wikipedia Dataset

A plain text English wikipedia scrape[1] was used articles where headings were delimited by special characters. The articles were then automatically segmented based on head-

[1] https://www.kaggle.com/datasets/ltcmdrdata/plain-text-wikipedia-202011/data

ings, and filtered to remove articles with very few segments, too short segments, or too much punctuation such as tables and figures. After this process, there remained approximately 1000 segmented articles. We generate two version of this dataset: one with headings removed, and one with headings included. We evaluate models on both versions of the dataset, but include only the version with headings removed in the final results. For full results, see the Appendix **??**.

2.1.3. Concatenated Wikipedia Dataset

We randomly sampled segments from the previous Wikipedia dataset and concatenated them in order to form new incoherent articles, with segments drawn from completely different domains. Intuitively, this dataset should be easier to segment as there are no semantic links between segments.

2.1.4. Synthetic Dataset

The final dataset used was generated synthetically by GPT-3.5. The source data was a mix of CTC sentinel data[2], UN-Peacekeeping corpus[3] and an internal dataset at Adarga. These documents were segmented by querying OpenAI's API with the model `gpt-3.5-turbo-16k` using the overlapping prompt schema defined in section 2.3. This dataset was exclusively used for fine-tuning the *FlanT5*3.1.5 model. It was not used for evaluation as the results would be biased in favor of the generative models.

2.2. Evaluation

The methodology in evaluating the quality of a model's segmentation in this work follows Chris Fournier's *Evaluating Text Segmentation using Boundary Edit Distance* (Fournier, 2013). This work uses edited versions of the Boundary Edit Distance proposed in this work, along with the associated information recall metrics 'boundary precision' and 'boundary recall'.

The boundary edit distance algorithm pairs matches of segment boundaries between a reference segmentation and a hypothesised segmentation, regardless of distance between matches. Exact matches score 1 and boundaries without matches score 0, whilst matches within some distance n score partial points based on a weighting function. For this work, the function always decreases score linearly as the boundary gets farther away from the true boundary. Boundary Edit Distance (B) is the mean score for all these matches, while Boundary Precision/Recall (BP/BR) measure the proportion of the hypothesis/reference boundaries which are matched, respectively.

A model is punished for both missing a boundary and for placing a boundary where there is none in a symmetric and intuitive manner. For further justification of the usage of the boundary edit distance algorithm rather than relying on more traditional metrics such as WD and Pk (Pevzner & Hearst, 2002), see *Evaluating Text Segmentation using Boundary Edit Distance* (Fournier, 2013), or see our own investigations at segmenter evaluation metrics.

2.3. LLM-Based Text Segmentation

How can we get LLMs to output segment boundaries?

We might first consider passing in the input text as a prompt and asking the LLM to copy out the text, adding markers indicating where it has placed boundaries as is suggested by (Xing, 2024). However, not only is this wasteful of tokens, especially if the input text is long, but crucially, the GLM may fail to copy the input accurately, may change the formatting, and we found through qualitative experimentation that boundary placement was not any better than our final method. These problems are addressed by (Xing, 2024) through repeated prompting until the input and output sequence lengths match, but this still does not guarantee integrity of the data. In our use case, guaranteeing that the input data would remain the same was of the utmost importance, so we used a different strategy.

2.3.1. Prompting Method

We first annotate the text with indices between each sentence. Here is an example: 'Hello World. [1] The sky is blue. [2] The sun is is yellow. [3] The grass is green. [4] Machine learning is a rapidly evolving field'. We then ask the LLM to return a list of indices corresponding to boundaries. In the previous example, the ideal response might be '1,4'. In practice, the texts and list of segment boundaries are much longer.

We add a system prompt which describes the segmentation task, desired output format and primes the model to be a talented linguist. We also add a variety of short examples in line with the few-shot prompting technique (Brown et al., 2020), which improved performance. This did not necessarily reflect the fact that the LLM learned how to segment better. Instead, through manual testing, we suspect that it learned the ideal segment length and amount of information that should be contained within a segment, which was implicitly contained in the few-shot prompt (and also the testing datasets, therefore increasing performance). This suggests that different implicit definitions of a segment could be imposed by a few-shot prompt to a LLM, dependent upon use case.

An example prompt is contained in [XXX Appendix A].

[2]`https://ctc.westpoint.edu/ctc-sentinel/`
[3]`https://peacekeeping.un.org/en/reports`

2.3.2. Overlapping Prompts

This prompting method works so long as the input text is within the context window of the LLM. At the time of experimentation, and with the use of gpt-3.5-turbo, we had a limit of 16k tokens. Many of our input texts exceeded this limit. Therefore, we needed to split up these long documents into smaller chunks that can be processed by the LLM. However, this cannot be done by simply splitting every at the nearest sentence before every 16k new tokens for two reasons. Firstly, we do not know whether this sentence boundary should serve as a segment boundary, and secondly, the LLM loses valuable context which helps to choose where to place boundaries at the extremes of the 16k tokens.

Therefore, we instead send 16k prompts with some overlap. The overlap is calculated as twice the maximum segment length. In our experiments, we set a maximum segment length of 750 tokens, thus there is an overlap of 1500 tokens between prompts. We must then decide which boundaries to accept in this overlapping region. Given two generations which were prompted by 1500 overlapping tokens, we choose to accept the segment boundaries contained within the first 750 tokens of the overlapping section from the first generation, and the boundaries in the final 750 tokens from the second prompt. We did not experiment with involving the responses from both outputs, but found that there seemed to be no sign of degrading performance towards segment boundaries.

While this method is wasteful of up to 1500 tokens per prompt, this is a small enough fraction of the 16k context that we were satisfied with the solution. If maximum segment lengths were much longer, say 5k tokens, the context may need to be limited more severely.

2.3.3. Segment Validation

We also performed some validation on the segments returned by the LLM. This primarily involved verifying that the returned segments are within a maximum and minimum segment length. Segments that are too short (for example, a model would sometimes return just a heading), were concatenated with another segment, and segments that are too long were recursively segmented by the same model, but with another prompt that asks the model to generate only one boundary at a time, which uses a similar few-shot prompting strategy to above. This way, a segment that is too long will be split in 2 recursively until all segments are within the specified lengths.

3. Experiments

3.1. Models

3.1.1. Baselines

We use two naive baselines as a point of reference. First, a segmenter which splits every n sentences. Based on preliminary testing, we decided to split every 5 sentences which we call the *Split5* segmenter. We also define a *RandomF0.1* segmenter which splits at 10% of boundaries, placed randomly, with each potential boundary equally likely.

3.1.2. BERT Segmenter

Our existing method generates a sequence of sentence similarities using sentence embeddings generated by (Reimers & Gurevych, 2019). Similarities are calculated as a weighted sum of the cosine similarity to the previous n sentences. Ideal boundaries are then generated as troughs in the sequence of similarities, before post-processing of segments ensures that no segment is too long or too short, either in sentence length or in token length. We name this the *BERT* segmenter. Further details are omitted for proprietary reasons.

3.1.3. BERT-Graph Segmenter

(Costacurta, 2023) describes 'text tiling' (topic segmentation) also using BERT-generated similarity scores followed by graph clustering to find the best segments and post-processing to ensure that the clusters return valid segments. Further details of this method can be found in the linked article. This model is called *BERTGraph* segmenter in our experiments. Code for this model was copied from the repository linked in the article.

3.1.4. GPT-3.5

OpenAI's `gpt-3.5-turbo-16k` was queried using the prompting and segment validation strategy defined in Section 2. We call this model *GPT3.5*. We used the largest model available to us, which is the 16k token context window. Note that this was done during the summer of 2023; the cheapest tier of model has increased in capabilities since then. Given the nature of the task, we chose to use deterministic outputs from the model, using topk $= 1$, rather than sampling. We are looking for the best possible segmentations, rather than a variety of possible segmentations, and this also helps with reproducibility of segmentations.

3.1.5. Flan-T5-Finetuned

Our intended application of segmentation requires us to use our own models, therefore we fine-tune Google's Flan-T5 large (Chung et al., 2022) on a combination of wiki, concatenated wiki and synthetic segmentations generated by

GPT-3.5 using LORA (Hu et al., 2021). We call this model *FlanT5*. We used the same deterministic output generation parameters as with *GPT3.5*. Because the model has been fine-tuned, we use a much shorter prompt with a brief instruction and no few-shot examples. Additionally, given the model's shorter context window, we have many more overlapping prompts per document.

3.2. Quantitative Results

Models are tested on the human-annotated dataset, the wikipedia dataset, and the concatenated wikipedia dataset. We do not test on the synthetic dataset results would be biased in favor of the generative models which generated the segmentations. Indeed, while performance of the LLMs on the synthetic dataset is far from perfect (less than boundary similarity of 0.5 with $n = 2$), it is disproportionately better than other models on the synthetic dataset.

Due to resource constraints, we could not test *GPT3.5* on the full wikipedia or full concatenated wikipedia datasets. Instead, we took the largest subset that fit within resource constraints. We evaluated all other models on the full datasets to verify that similar results are obtained. Further details on the experimental procedure are witheld for proprietary reasons.

We evaluate primarily using the previously discussed boundary similarity metric with $n = 2$. Results can be seen in Table 1. Evaluation was also run with boundary similarity $n = 5$, but the metric becomes noisier with higher values of n. Indeed, the baselines begin to report better performance with high enough n. Full results for both the smaller and larger datasets and all evalutation metrics can be found in the Appendix **??**.

We see that *GPT3.5* outperforms all other models on the human-annotated dataset, the wikipedia dataset, and the synthetic dataset, with the other LLM *FlanT5* coming in second in the same datasets. The close performance between the models on the synthetic dataset implies that *FlanT5* is a good approximation of *GPT3.5* for this task, on boundaries generate by *GPT3.5*. However, the biggets performance gap is on the human-annotated dataset. Although this dataset is small, it represents a meaningful distribution shift and, unsurprisingly, the fine-tuned model is unable to generalize as well as the base model.

Interestingly, we see that both BERT-based models are superior in the supposedly easier task posed by the concatenated wikipedia dataset. We theorise that these models are better at finding clear boundaries between different domains, but struggle with more nuanced segment boundaries found in a news article, for example. Whereas, the LLMs are better at finding more nuanced segments that a human might produce, but are not significantly better at finding these easier boundaries.

	Human	Wiki	Conc-Wiki	Synthetic
GPT3.5	**0.38**	**0.25**	0.29	**0.35**
FlanT5	0.25	0.24	0.41	0.33
BERTGraph	0.20	0.15	0.45	0.21
BERT	0.18	0.09	**0.46**	0.18
RandomF0.1	0.09	0.11	0.10	0.10
Split5	0.13	0.19	0.19	0.23

Table 1. Boundary similarity scores with $n = 2$ for each model and the human, wiki, concatenated wiki and synthetic datasets.

We also looked at precision and recall to better characterize the behavior of each model. Results can be seen in Table 2. *GPT3.5* has the highest precision and recall in the human-annotated dataset and the highest recall on both the wiki and concatenated wiki datasets. This property of having high recall is true in general for the LLMs, even in the concatenated wiki dataset where the BERT models have higher overall boundary similarity. By contrast, precision scores are closer in general and much better for the BERT models on the concatenated wiki dataset. This suggests that, in general, the BERT models are more hesitant in placing boundaries, but when they do, they are more likely to be correct. It should be noted that this behaviour in both the LLMs and BERT models is subject to the prompt engineering and segment processing procedures used.

	Human		Wiki		Conc-Wiki	
	BP	BR	BP	BR	BP	BR
GPT3.5	**0.51**	**0.60**	0.36	**0.55**	0.42	**0.63**
FlanT5	0.38	0.46	**0.43**	0.37	0.65	**0.63**
BERTGraph	0.47	0.25	0.39	0.21	0.79	0.54
BERT	0.33	0.39	0.23	0.37	**0.91**	0.50
RandomF0.1	0.16	0.21	0.34	0.16	0.20	0.16
Split5	0.17	0.48	0.33	0.39	0.25	0.52

Table 2. Boundary Precision (BP) and Recall (BR) with $n = 2$ for each model and the human, wiki, and concatenated wiki datasets.

3.3. Qualitative Evaluation

Limited manual testing and inspection of the segmentations produced by the models was also conducted, both during prompt engineering and after the models were trained.

Through manual inspection of segmentations with the human-annotated dataset, we found that *GPT3.5* generally found the boundaries which seemed reasonable from a human perspective, especially for simple documents like short news articles. The fine-tuned *FlanT5* model imitated this behavior, but was less consistent. BERT segmenters would find reasonable segments, but after the manual gluing and splitting procedure, would often lead to off-by-1 errors.

For documents with far more complex documents such as a podcast transcript, or with messy data like tables or artefacts from pdf to text conversions, *GPT3.5* would sometimes return indices with a regular pattern. For example, the LLM might return '$[1, 15, 22, \ldots, 76, 79, 82, 85, 88, \ldots, 184, 187, \ldots]$'. Often, the pattern would continue far beyond the number of sentences in the input indicating that the model became stuck in a regular pattern. Perhaps better prompt engineering, a more rigorous data-processing procedure, the use of newer models would help or the use of better generation parameters, but our current approach was resource constrained and required the ability to pass noisy documents to the model. A more thorough investigation of the logits computed by the model is required to understand how and when this occurs, and how to mitigate it.

4. Conclusion

Our work empirically compares generative LLMs with previous methods which use BERT embeddings and cosine similarity. In order to be more token-efficient and to provide guarantees that the original document will be unedited, we propose a new overlapping prompt schema, which centres on asking the LLM to return a list of indices corresponding to segment boundaries. We also support the use of boundary similarity and its associated information recall metrics as an evaluation for topic segmentation. Results indicate that LLMs can be more effective segmenters than existing methods where more nuanced segmentations are required, but that when the input is noisy or the segment boundaries are clear, BERT-based methods are more reliable. Future work should focus on addressing highlighted issues with LLMs, such as the regular patterns found in segmentations, and a more thorough comparison of promptin methods. Lastly, larger human-annotated datasets should be constructed and used to better assess generalisation capabilities.

References

Badjatiya, P., Kurisinkel, L. J., Gupta, M., and Varma, V. Attention-based neural text segmentation. In Pasi, G., Piwowarski, B., Azzopardi, L., and Hanbury, A. (eds.), *Advances in Information Retrieval*, pp. 180–193, Cham, 2018. Springer International Publishing. ISBN 978-3-319-76941-7.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation, 2023. URL https://arxiv.org/abs/2306.15595.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

Costacurta, M. Text tiling done right: Building solid foundations for your personal llm, 2023.

Eisenstein, J. Hierarchical text segmentation from multi-scale lexical cohesion. In Ostendorf, M., Collins, M., Narayanan, S., Oard, D. W., and Vanderwende, L. (eds.), *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 353–361, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL https://aclanthology.org/N09-1040.

Feng, X., Feng, X., Qin, L., Qin, B., and Liu, T. Language model as an annotator: Exploring dialogpt for dialogue summarization, 2021. URL https://arxiv.org/abs/2105.12544.

Fournier, C. Evaluating text segmentation using boundary edit distance. In Schuetze, H., Fung, P., and Poesio, M. (eds.), *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1702–1712, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/P13-1167.

Galley, M., McKeown, K. R., Fosler-Lussier, E., and Jing, H. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 562–569, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075167. URL https://aclanthology.org/P03-1071.

Glavaš, G. and Somasundaran, S. Two-level transformer and auxiliary coherence modeling for improved text segmentation, 2020. URL https://arxiv.org/abs/2001.00891.

Hearst, M. A. Texttiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1): 33–64, mar 1997. ISSN 0891-2017.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Koshorek, O., Cohen, A., Mor, N., Rotman, M., and Berant, J. Text segmentation as a supervised learning task, 2018. URL https://arxiv.org/abs/1803.09337.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts, 2023.

Lo, K., Jin, Y., Tan, W., Liu, M., Du, L., and Buntine, W. Transformer over pre-trained transformer for neural text segmentation with enhanced topic coherence. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3334–3340, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.283. URL https://aclanthology.org/2021.findings-emnlp.283.

Lukasik, M., Dadachev, B., Simões, G., and Papineni, K. Text segmentation by cross segment attention, 2020. URL https://arxiv.org/abs/2004.14535.

Petroni, F., Lewis, P., Piktus, A., Rocktäschel, T., Wu, Y., Miller, A. H., and Riedel, S. How context affects language models' factual predictions, 2020. URL https://arxiv.org/abs/2005.04611.

Pevzner, L. and Hearst, M. A. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002. doi: 10.1162/089120102317341756. URL https://aclanthology.org/J02-1002.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Wang, P., Qian, Y., Soong, F. K., He, L., and Zhao, H. Learning distributed word representations for bidirectional LSTM recurrent neural network. In Knight, K., Nenkova, A., and Rambow, O. (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 527–533, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1064. URL https://aclanthology.org/N16-1064.

Xing, L. *Versatile neural approaches to more accurate and robust topic segmentation*. PhD thesis, University of British Columbia, 2024. URL https://open.library.ubc.ca/collections/ubctheses/24/items/1.0440128.

A. Prompting Strategy

The prompting strategy used in this work is a simple schema that is designed to be general and applicable to any LLM. The schema is as follows:

1. The LLM is prompted with the input text, with integers in square brackets delimiting the sentence boundaries, few-shot examples of the task,a short instruction and a system prompt.
2. Segments are validated. This means they must be not too long nor too short, and that they do not contain too many punctuation marks as a proportion of the segment length.
3. Segments that are too long are recursively split into smaller segments through similar prompting strategy, but this time the LLM is asked to return a single segment boundary index.
4. This process is repeated until all segments are short enough.
5. Segments that are too short are merged with a neighbouring segment based on the semantic similarity to neighbouring sentences. This part could also be done via prompting, but we found this unnecessary.

An example prompt is shown below. Note that this is not the exact prompt used in the experiments, but a simplified version for illustrative purposes.
System:
You are an expert linguist and a master of nuance in meaning of written text. You obey instructions. You do not hallucinate. You are not a chatbot. You are not a summariser.
Prompt:
You are given a document with sentence boundaries marked by square brackets. Your task is to segment the document into coherent parts. Return a list of indices corresponding to the segment boundaries of the document. This list should ONLY be a list of integers, for example '1, 3, 5'. Some examples are shown below.
Text:
It was a sunny day in the park. [1] The birds were singing. [2] The children were playing. [3] The adults were chatting. [4] The dogs were barking. [5] The sun was shining. [6] The day was perfect. [7] However, then the rain came. [8] The children ran for cover. [9] The adults laughed. [10] The dogs howled. [11] The sun disappeared. [12] The day was ruined. [13] Fortunately, the next day was sunny again. [14] But it was actually too hot! [15] The children were sweating. [16] The adults were fanning themselves. [17] The dogs were panting. [18] The sun was scorching. [19] The day was unbearable.
Segments:
7, 13
. . . *[more examples]* . . .
Text:
The cat sat on the mat. [1] The dog sat on the floor. [2] The cat was black. [3] The dog was brown. [4] The cat was fluffy. [5] The dog was short-haired. [6] The cat was purring. [7] The dog was wagging its tail. [8] The cat was happy. [9] The dog was happy. [10] Then the cat went to London. [11] The dog went to Paris. [12] The cat saw the sights. [13] The dog saw the sights. [14] The cat ate fish and chips. [15] The dog ate croissants. [16] The cat drank tea. [17] The dog drank coffee. [18] The cat was happy. [19] The dog was happy.
Segments:
End Prompt
We use a similar prompt for the recursive prompting mechanism with the same system prompt. For example:
Prompt:
You are given a document with sentence boundaries marked by square brackets. Your task is to choose one segment boundary to split the document into two coherent parts. Return a single integer corresponding to the index of the segment boundary. This integer should be between 1 and the number of sentences in the document. Some examples are shown below.
Text:
The cat sat on the mat. [1] The cat was black. [2] The cat was fluffy. [3] The cat was purring. [4] The cat was happy. [5] On the other hand, the dog sat on the floor. [6] The dog was brown. [7] The dog was short-haired. [8] The dog was wagging its tail. [9] The dog was happy. [10]
Segment:
5
. . . *[more examples]* . . .
Text:
Jack and Jill went up the hill. [1] Jack fell down and broke his crown. [2] Jill came tumbling after. [3] This is a well known nursery rhyme that has been passed down through the generations. [4] It is a classic. [5] It is a favourite of many. [6] It is a favourite of mine. [7] It is a favourite of yours. [8] It is a favourite of everyone.
Segment:
3