"And cut!" Exploring Textual Representations for Media Content Segmentation and Alignment

ISMAIL HARRANDO, EURECOM, France
RAPHAËL TRONCY, EURECOM, France

Text segmentation is a traditional task in NLP where a document is broken down into smaller, coherent segments. While several methods and benchmarks exist for well-formed, and clean textual documents that can be found in long articles or synthetic datasets, segmenting media content comes with different challenges such as the errors produced by the procedure of *Automatic Speech Recognition* and the lack of sentence end markers that are found in written text (e.g. punctuation marks or HTML tags). Many radio or TV programs are also conversational in nature (e.g. interview and debate), and thus, rely less on repeated words unlike encyclopedia text (e.g. Wikipedia articles) that are frequently used for textual segmentation evaluation. This is even further compounded when working with non-English content. In this work, we present an approach to content segmentation that leverages topical coherence, language modeling and word embeddings to detect change of topics. We evaluate our approach on a real production dataset of French programs that have been manually annotated for segments. We also show how, when a ground truth summary of the content is provided such as segment titles, we can align them to their corresponding segments using the same representations.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; • **Information systems** → **Content analysis and feature selection**.

Additional Key Words and Phrases: Content segmentation; Content Alignment

1 INTRODUCTION

As the amount of multimedia content created and published every day has seen a remarkable growth in the recent years, the ability to serve end-users the content they are interested in becomes a crucial ingredient to ensure their engagement. There is therefore a need to segment available long-format content into shorter pieces that can match a user's preferences better. For instance, segmenting a news broadcast into multiple stories spanning different themes and topics can help online content distribution platforms to serve different users with different parts of the same broadcast. Content segmentation has also been shown to improve other media-related tasks such as content retrieval [19], content summarization [9], and sentiment analysis [10].

While the task of document or text segmentation has been studied extensively in the literature, segmenting multimedia content present challenges that are particular to the medium: multimodality, automatic transcription errors, lack of proper punctuation, and presentation style (more informal talking, the use of pronouns and references instead of repeating words, etc.). To tackle the task of media content segmentation using automatically generated subtitles, we propose a textual approach that relies on combining several linguistic methods (topic modeling, words embeddings and sentence encoders) with minimal supervision to generate richer representations of the content that we then use to predict segment boundaries.

Proceedings of 2nd International Workshop on Data-driven Personalisation of Television (DataTV-2021) at the ACM International Conference on Interactive Media Experiences (IMX 2021), June 2021.

2 RELATED WORK

While work on the task of document segmentation dates back to at least as early as 1984 [16], the most popular approach to text segmentation, *TextTiling*, was proposed by Marti Hearst in 1997 [7], who devised an unsupervised approach in 3 steps: first, the text is divided into fixed-length sequences of words (called *blocks*), which are then transformed into a Bag of Words representation. The cosine similarity between adjacent sequences is computed, and the boundary between segments is determined at the position where the similarity is at its lowest, based on a sliding window. This classic text segmentation algorithm has been subsequently enhanced by different improvements addressing multiple challenges for the algorithm. [1] showed how introducing the time spoken by every participant in a recorded meeting as a feature can be used to better predict segment boundaries, as participants are typically not interested in every part of the meeting. [20] proposed to use word embeddings instead of word counts (bag of words) as more robust representations of the blocks to compare, and introduced a new heuristic to better capture the semantic coherence with the distributed document representations. More recently, He et al. proposed an improvement over the last step of the algorithm, boundaries detection, by average-smoothing the similarity curve for adjacent blocks [6]. This allows the local variations within topics to be smoothed-out whereas the topic switch would be perceived more clearly.

With the paradigm shift to neural networks in the 2010s, multiple neural models were proposed to address this task as a supervised learning problem. Recently, Lukasik et al. [12] proposed an approach based on training a BERT-based [5] model on the task, where they compared 3 potential architectures to detect segment boundaries. They show that relying on attention between words and then between segments improves the results significantly on some standard benchmarks. Similarly, Yoong et al. [21] relied on BERT and the attention mechanism, and proposed 3 training pipelines: a naive approach where a BERT model is given two sentences as input and is trained to decide if they belong to the same segment or not (binary classification); in a second approach, all sentences of the documents are fed to the model, and a decision is made on the `[SEP]` token separating them; finally, in a third approach, the segment boundary is modeled as a `[SEP]` token, and thus the task of segment prediction becomes one of a masked token prediction.

While the aforementioned methods are mostly evaluated on either synthetic datasets (where unrelated documents are concatenated to produce a segment boundary) or Wikipedia articles, some research work was particularly devoted to media content. In [18], Sehikh et al. proposed an supervised approach based on a Bi-LSTM that is trained on a synthetically generated dataset to predict content segments of French News programs. Similarly, Scaiano et al. [17] proposed an approach for automatic segmentation of movie subtitles to improve information retrieval from films. They based their approach on TextTiling, but used synsets instead of words only to construct the Bag of Word representation of sentences. They also propose a filtering of segments based on the expectation that the similarity curve should be sinusoidal, and thus a minimum difference between the peaks (highest similarity) and valleys (lowest similarity) should be present to validate a proposed segment. Berlage et al.[2] proposed improving automated segmentation of radio programs by adding audio embeddings to the text input. Finally, Zhang and Zhou [22] used a temporal convolutional network (TCN) combined with BERT features to perform dialog stream segmentation, while introducing speaker information as part of the input sequence, and observed significant improvement over several dialog segmentation datasets.

3 APPROACH

The main steps of our approach are similar to TextTiling [7], i.e. partitioning text into fixed-size sequences of words, or *blocks*, computing pairwise similarity between adjacent blocks, and assigning segment ends to the minima of the

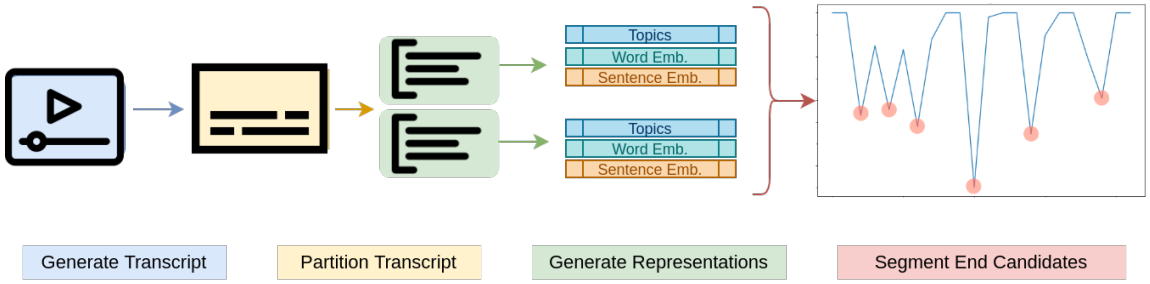

Fig. 1. High level illustration of the approach: (1) Generate a transcript of the program using ASR. (2) Partition the transcript into *blocks* of equal size N. (3) Generate different representations of the textual content of each block. (4) After measuring the similarity between each block and its neighborhood, each "valley" in the similarity curve is a candidate to be the topic transition block (i.e. end of the segment)

similarity curve (Figure 3). We extend this approach by leveraging multiple text representations instead of simple word counts or embeddings, and by smoothing the similarity curve by considering a window of adjacent similarity.

The high-level description of our approach is illustrated in Figure 1. We detail each steps in the following subsections.

3.1 Transcript Partitioning

One of the main differences between traditional documents and automatically generated transcripts is the lack of natural sentence end markers. While most ASR systems cut long utterances into smaller sentences, they vary considerably in length, and tend to be too short to carry meaningful topical information. As a simple partitioning method, we divide the content of each program, as generated by the ASR system and after removing stopwords, into *blocks* of a fixed number of words per block N.

3.2 Text Representation

To find segment boundaries, we need to find the blocks in the transcript where a *topic shift* takes place, i.e. where the similarity between the current block and the following one (or ones), is lowest. To do that, we generate several textual representations that allow us to measure similarity between blocks from the transcript. Since all these methods produce a fixed-size vector representation, we compute the similarity between blocks using cosine similarity (i.e. normalized dot product).

The curve of adjacent blocks similarity tends to be spiky: a lot of peaks and valleys come naturally from the variability of the vocabulary between immediately consecutive utterances. Therefore, we also consider measuring the similarity of each block to the ones following it within a perimeter of $window\_size$. This has both a smoothing effect for sharp transitions in similarity as well as removing saddle points (stretches of the curve where the score does not change).

3.2.1 Word Embeddings. Pretrained word embeddings have been a fixture in most NLP tasks, especially for unsupervised methods. For our experiments, we use the pretrained French *fastText* embeddings [3]. Beyond their empirical performance as standalone word embeddings, fastText embeddings have the capacity of generating a representation even for words that are outside of the training vocabulary by leveraging their sub-word components. We use the 300-d pretrained vectors, available on the official website.[1]

[1] https://fasttext.cc/docs/en/crawl-vectors.html

3.2.2 Sentence Encoder. Another way to represent the textual content is through the use of Sentence Encoders which attempt to capture the meaning of a sentence through both its constituent words and its grammatical structure. While there is a rich literature on the topic, most state-of-the-art applications use *Sentence-BERT* [15], which uses pretrained BERT to construct semantically meaningful sentence embeddings that can be compared using cosine-similarity. We use the `sentence-transformers` Python package[2] to generate sentence embeddings for our program content.

3.2.3 Topic Modeling. Since the ultimate goal of this task is to segment text into topically coherent segments, it shares several aspects with Topic Modeling. While generally used to infer topic information about given texts, the output of a topic model can be used as a "feature vector", or a representation of a given text, i.e. as a linear combination of its latent topical components. We select LDA as our topic model based on empirical evaluation of several models (using the Python library `Tomodapi` [11]). We train the model on a synthetic dataset that we create by concatenating adjacent blocks from our original dataset (as adjacent blocks are highly likely to talk about the same topic) as well as succeeding lines from the automatically generated transcript (i.e. before partitioning into blocks). It is worth noting that LDA has also the property of producing sparse representation, i.e. every document only falls into a few (3 or less) topics, which makes most of the representation components null.

Figure 2 visualizes the representations for an example on the dataset using LDA features.

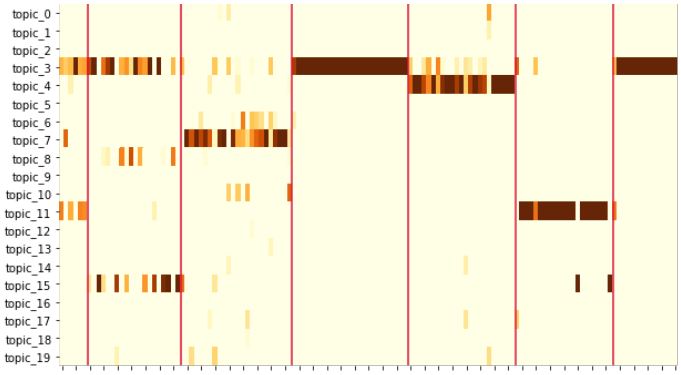

Fig. 2. Visualizing the topic distribution over an example in the dataset. The vertical lines represent the ground truth segment boundaries

3.3 Boundaries selection

As mentioned above, we consider a *boundary candidate* to be a minimum in the similarity curve, i.e. the similarity scores resulting from comparing the content of the block at position i with that at position $i+1$. In the case of $window\_size > 1$, we average the similarity scores between the content at block i and all blocks between $i+1$ and $i+window\_size$. Figure 3 shows an example of the process (with $window\_size = 3$).

An important parameter in the boundaries selection is the *number of segments* in the program. Because our main goal is to find good textual representations for the task of segmentation, we consider the number of parts as given, i.e. for every program, we only propose as many segments as there are in the ground truth. We show in Section 4 some simple heuristics to guess this ground truth information.

[2] https://github.com/UKPLab/sentence-transformers

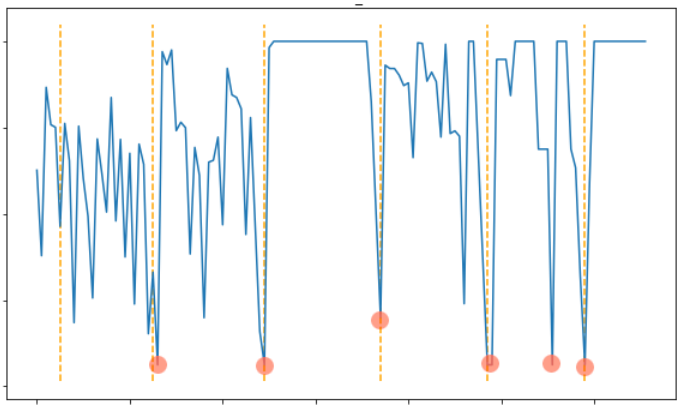

Fig. 3. An example of a similarity curve generated by topical similarity. The circles highlight the valleys that correspond to the segment boundaries selected by our approach. We note that in this case, the number of segments is given. The dashed lines represent the ground truth segment boundaries

4 EXPERIMENTS AND RESULTS

In this section, we describe the dataset we are using for our experiments as well as the different experimental settings. For the evaluation, we consider segmentation as a classification task, where each block is assigned a label: 0 if it is part of a homogeneous segment, or 1 if it represents a topic transition block, i.e. having a low similarity to the blocks following it.

4.1 Dataset

For our evaluation, we use a production dataset from INA (the French National Audiovisual Institute) containing 46 programs from the same week of publication (May 19th to 26th, 2014), with a total runtime of 15 hours, that were segmented into 476 parts, of 112 seconds duration in average. The segmentation is done manually by archivists and each part is given a title. Most of the programs that are provided are news broadcasts, with the segments corresponding to news stories, but the dataset also includes some sport and cultural event coverage[3]. Each program in this dataset has been automatically transcribed using the LIUM ASR system [4]. It is worth noting that the segmentation boundaries contain some noise as they do not perfectly align with ASR nor does the total duration of segments usually add up to the duration of the program.

4.2 Segmentation

For each of the textual representation, we evaluate the data using traditional classification measures (Precision, Recall, F1 score) which quantify the amount of exact segment boundaries detected by each method, as well as two segmentation-specific metrics [14]:

- P_k: computes the probability that two blocks (sentences) i and j such that $i + j = k$ are within the same reference (ground truth) segment. Concretely, moving a sliding window of size k, each time there is a disagreement between the hypothetical segmentation (produced by the algorithm) and that of the ground truth (i.e. the ground truth saying that the two blocks belong to the same segment but the model predicts otherwise), a counter is increased.

[3]The reader interested in the dataset can contact the authors.

The final P_k score is the value of this counter divided by the number of evaluated windows. Thus, it is equal to 0 if the two segmentations are identical, and 1 if there is a disagreement in every possible window of evaluation.

- *WindowDiff*$_k$: a variation of P_k that "penalizes false-positives and near-misses equally"[13]. It does so by considering not only whether the blocks fall into the same or different segments, but also whether there are extra segmentation boundaries (i.e. false positives) within the evaluation window k. Similarly to P_k, the metric gets closer to 0 the closer the predicted segmentation is to the ground truth.

As per convention, we set $k = 2$ for both P_k and WindowsDiff$_k$, which corresponds to half the average length of a segment (in blocks).

Considering the three text representations described in Section 3, we propose several variants:

- **Sentence-BERT**: we consider three variants representing pretrained multilingual models on different tasks: `distilusebase-multilingual` (distilled base multilingual BERT), `paraphrase-xlm-r-multilingual` (XLM[8] fine-tuned on the task of paraphrasing), and `stsb-xlm-r-multilingual` (XLM fine-tuned on the task of Semantic Textual Similarity Benchmark).
- **fastText**: for both variants we use pretrained French fastText embeddings. We test two similarity measures: averaging all the embeddings in each block to form a block representation that is then used for cosine similarity (`fastText-avg`), or, as suggested by [20], we keep the best cosine similarity between two blocks, i.e. the similarity scores for the most similar words in the two successive blocks (`fastText-max`).
- **LDA**: We train an LDA model with the same hyper parameters with different number of topics, thus changing the size of the representation vector. We set both *alpha* and *eta* (the Dirichlet priors for the per-document topic distributions and per-topic word distributions, respectively) to *'auto'* (learned from the corpus), while varying the number of topics T between 10, 20 and 30.

As previously mentioned, the similarity scores are computed using cosine similarity (normalized dot product) between the vector representations of adjacent blocks or within a window thereof. We set the block size $N = 20$.

In Table 1, we show the results on the INA dataset using the different text representations used to measure textual similarity between content blocks. For the *Combined* line, we consider a linear combination of similarity scores generated from the best performing variant from each representation (on our evaluation dataset, the combination 0.6, 0.3, 0.1 for `LDA-20`, `fastText-avg` and `S-BERT-stbt`, respectively). Among the text representations, we see that LDA performs best for both the classification and segmentation metrics. The combined score, however, generally outperforms the individual representations, showing that each of the representations contain different but complementary information.

In Table 2, we improve on the previous approach by extending the similarity measure to a window of size > 1, as the smoothing effect can cover some of the noise that is present in the data. This turns out to be the case, as extending the similarity to a vicinity of 3 (selected empirically) blocks instead of just one, we see a noticeable improvement over almost all representations. We also report the best results on the *Combined* representation, which outperforms all individually presented methods (in this setting, we use `S-BERT-paraphrase` in the combined representation as it provides better results).

4.2.1 Block Size. In this section, we study the empirical effect of the size of the unit partitioning block N. We repeat the experiments explained in this section for block size 10, 20 and 30. In Table 3, we report the results on the dataset using the *Combined* representation with *window_size* = 3, as it still performs best among all approaches considered.

Approach	*Pre*	*Rec*	*F1*	P_k	WD
S-BERT-paraphrase	0.235	0.311	0.261	0.467	0.505
S-BERT-distiluse	0.255	0.343	0.284	0.445	0.476
S-BERT-stsb	0.266	0.352	0.296	0.447	0.495
fastText-max	0.235	0.271	0.251	0.416	0.440
fastText-avg	0.258	0.300	0.277	0.401	0.439
LDA ($T = 10$)	0.297	0.377	0.330	0.378	0.424
LDA ($T = 20$)	0.291	0.421	0.335	0.398	0.447
LDA ($T = 30$)	0.297	**0.440**	**0.344**	0.412	0.474
Combined	**0.321**	0.371	**0.344**	**0.355**	**0.392**

Table 1. Segmentation results on the INA dataset ($window\_size = 1$). We observe that for P_k and WD, lower values are better.

Approach	*Pre*	*Rec*	*F1*	P_k	WD
S-BERT-paraphrase	0.281	0.377	0.313	0.427	0.492
S-BERT-distiluse	0.253	0.342	0.283	0.443	0.503
S-BERT-stsb	0.270	0.352	0.298	0.422	0.474
fastText-max	0.245	0.281	0.262	0.423	0.451
fastText-avg	0.278	0.324	0.298	0.399	0.454
LDA ($T = 10$)	0.397	0.469	0.429	0.313	0.368
LDA ($T = 20$)	0.399	0.473	0.431	0.319	0.370
LDA ($T = 30$)	0.374	0.453	0.409	0.340	0.396
Combined	**0.431**	**0.500**	**0.462**	**0.291**	**0.345**

Table 2. Segmentation results on the INA dataset. We observe that for P_k and WD, lower values are better.

Block Size	*Pre*	*Rec*	*F1*	P_k	WD
10	0.178	0.327	0.222	0.320	**0.334**
20	**0.431**	**0.500**	**0.462**	**0.291**	0.345
30	0.521	0.345	0.400	0.419	0.456

Table 3. Comparing performance as a function of the partitioning block size

From the results, we see clearly that for $N = 10$, the smaller blocks fail to capture enough topical information, as we see a significant drop in all metrics. As for $N = 30$, we see an increase of Precision (i.e. a higher ratio of true positives), but at the cost of recall and overall F1-score.

4.2.2 Number of segments. For the previous experiments, we set the number of segments for each program to be equal to that of the ground truth, which is an ideal setting just to evaluate the performance of the representations. In Table 4, we present experiments with two simple heuristics:

- **1/6**: we pick the number of segments to be equal to a sixth of the number of blocks generated for the program. As we computed the ratio of *blocks to segment* to be equal to 1/6.

- **Thresh**: we only keep the segmentation candidate at position i if it satisfies the following inequality:

$$h(i) = min(hr(i), hl(i))$$

$$\frac{1}{N}\sum_{j}^{N} h(j) - sim(j, j+1) < h(i) - sim(i, i+1)$$

  with $hr(i)$ and $hl(i)$ two functions returning the nearest peak (maximum) to the right and the left of i, respectively, and they are both defined for each program. In concrete terms, this means we only keep the candidates which are situated at valleys that are deeper (expressed in the left-hand term) than the average valley in the entire similarity curve (right-hand term).
- **GT** (ideal case): we reproduce the results from the previous experiments with the number of segments to be picked is equal to that of the ground truth.

Block Size	*Pre*	*Rec*	*F1*	P_k	*WD*
GT	0.431	**0.500**	**0.462**	0.291	0.345
1/6	0.266	0.478	0.340	**0.278**	**0.297**
Thresh	**0.451**	0.297	0.384	0.329	0.394

Table 4. Comparing performance as a function of the number of segment selection method

As we see the results in Table 4, the different methods offer different compromises. While using 1/6, by virtue of detecting less boundaries on short programs, we get better P_k and WindowDiff$_k$ scores than when using the ground truth, but the classification scores are comparatively low. Whereas for *Thresh*, we get segmentation scores that are close to GT, while not losing as much in classification scores.

4.3 Aligning segments with description metadata

In our ground truth, every annotated segment is given a title that corresponds to a summary of its content. Given how in production, there is typically metadata about the content of the program (e.g. news segment titles), we further test the scenario of aligning the automatically generated transcript with the existing metadata. In this setting, we consider at first the number of segments given (to be equal to the number of provided segment titles), and we create an alignment by measuring the similarity between each block in the transcript (we keep the block size $N = 20$) and a title from the ground truth annotation, using all the representations we mentioned above. To find the segment boundaries, we measure the similarity of each title to all blocks. Starting from the first title $t = 0$, segment boundaries are put where the similarity to title $t + 1$ is higher than that to t (similarity to the next segment title is higher than the one to the current examined title, signaling a topic switch).

As we can see in Table 5, the results based on content alignment with the titles, while comparable to the segmentation results on P_k and WindowDiff, are significantly lower on classification metrics. Upon analysis, we see that this is probably due to the shortness of the descriptive titles, which do not carry enough information to measure similarity significantly, regardless of the chosen textual representation (all methods perform comparatively the same). A combined decision (obtained by assigning the coefficients $0.5, 0.3, 0.2$ to the similarity score of `S-BERT-paraphrase`, `fastText` and `LDA`, respectively), however, does improve the results, which highlights again the fact that leveraging on multiple textual representations is key to improving the overall segmentation results.

Approach	*Pre*	*Rec*	*F1*	P_k	*WD*
S-BERT-paraphrase	0.281	**0.377**	0.313	0.427	0.492
fastText-avg	0.241	0.243	0.243	0.406	0.448
LDA ($N = 20$)	0.264	0.263	0.264	0.387	0.432
Combined	**0.390**	0.271	**0.319**	**0.296**	**0.342**

Table 5. Alignment results on the INA dataset

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a method for content segmentation based on combining multiple text representations, and we show that topic modeling is a useful representation for this task. More advanced methods for deriving and combining the representations, as well as finding the number of segments in the program, can be considered in the future. We would also like to explore the use of multimodal features to further improve the segmentation: audio features such as silence periods and speaker turns, and visual features (e.g. visual shot similarity, scene segmentation) can also help complementing textual content for programs that present more visual diversity.

6 ACKNOWLEDGMENTS

This work has been partially supported by the French National Research Agency (ANR) within the ANTRACT (ANR-17-CE38-0010) project, and by the European Union's Horizon 2020 research and innovation program within the MeMAD (GA 780069) project.

REFERENCES

[1] Satanjeev Banerjee and Alexander I. Rudnicky. 2006. A texttiling based approach to topic boundary detection in meetings. In 9^{th} *International Conference on Spoken Language Processing (INTERSPEECH)*. ISCA, Pittsburgh, PA, USA.

[2] Oberon Berlage, Klaus-Michael Lux, and David Graus. 2020. Improving Automated Segmentation of Radio Shows with Audio Embeddings. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020).

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[4] Fethi Bougares, Paul Deléglise, Yannick Estève, and Mickael Rouvier. 2013. LIUM ASR system for ETAPE French evaluation campaign: Experiments on system combination using open-source recognizers, Vol. 8082. 319–326.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186.

[6] Xin He, Jian Wang, Quan Zhang, and Xiaoming Ju. 2020. Improvement of Text Segmentation TextTiling Algorithm. *Journal of Physics: Conference Series* 1453 (2020).

[7] Marti A. Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics* 23, 1 (1997), 33––64.

[8] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[9] Joel Larocca Neto, Alexandre D. Santos, Celso A.A. Kaestner, and Alex A. Freitas. 2000. Generating Text Summaries through the Relative Importance of Topics. In *Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 300–309.

[10] J. Li, B. Chiu, S. Shang, and L. Shao. 2020. Neural Text Segmentation and Its Application to Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[11] Pasquale Lisena, Ismail Harrando, Oussama Kandakji, and Raphael Troncy. 2020. TOMODAPI: A Topic Modeling API to Train, Use and Compare Topic Models. In 2^{nd} *Workshop for NLP Open Source Software (NLP-OSS)*. Association for Computational Linguistics, 132–140.

[12] Michal Lukasik, Boris Dadachev, Kishore Papineni, and Gonçalo Simões. 2020. Text Segmentation by Cross Segment Attention. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 4707–4716.

[13] Igor Malioutov, Alex Park, Regina Barzilay, and James Glass. 2007. Making Sense of Sound: Unsupervised Topic Segmentation over Acoustic Input. In *45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, 504–511.

[14] Lev Pevzner and Marti A. Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics* 28, 1 (2002), 19–36.

[15] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3980–3990.

[16] John A. Rotondo. 1984. Clustering analysis of subjective partitions of text. *Discourse Processes* 7, 1 (1984), 69–88.

[17] Martin Scaiano, Diana Inkpen, Robert Laganière, and Adele Reinhartz. 2010. Automatic Text Segmentation for Movie Subtitles. In *Advances in Artificial Intelligence*, Atefeh Farzindar and Vlado Kešelj (Eds.). Springer Berlin Heidelberg, 295–298.

[18] Imran A. Sheikh, Dominique Fohr, and Irina Illina. 2017. Topic segmentation in ASR transcripts using bidirectional RNNS for change detection. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU*. IEEE, 512–518.

[19] Gennady Shtekh, Polina Kazakova, Nikita Nikitinsky, and Nikolay Skachkov. 2018. Applying Topic Segmentation to Document-Level Information Retrieval. In *14th Central and Eastern European Software Engineering Conference Russia* (Moscow, Russian Federation). Association for Computing Machinery.

[20] Yiping Song, Lili Mou, Rui Yan, Li Yi, Zinan Zhu, Xiaohua Hu, and Ming Zhang. 2016. Dialogue Session Segmentation by Embedding-Enhanced TextTiling. *CoRR* abs/1610.03955 (2016). arXiv:1610.03955 http://arxiv.org/abs/1610.03955

[21] Siang Yun Yoong, Yao-Chung Fan, and Fang-Yie Leu. 2021. On Text Tiling for Documents: A Neural-Network Approach. In *Advances on Broad-Band Wireless Computing, Communication and Applications*, Leonard Barolli, Makoto Takizawa, Tomoya Enokido, Hsing-Chung Chen, and Keita Matsuo (Eds.). Springer International Publishing, 265–274.

[22] L. Zhang and Q. Zhou. 2019. Topic Segmentation for Dialogue Stream. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 1036–1043.