A Deep Learning Solution for Precise Subtitle Segmentation

Anush Veeranala, Xinyu Zhang, Lilin Jin

May 6, 2024

Abstract

Despite increased adoption, effective subtitle segmentation poses a challenge, particularly in libre-software contexts. Our research addresses this gap by proposing a reliable subtitle segmentation solution, leveraging deep learning techniques to advance comprehension, streamline academic processes, and reduce dependence on proprietary solutions and volunteer efforts.

1 Introduction

Subtitles play a crucial role in fostering the inclusive nature of videos, ensuring accessibility for individuals with hearing impairments and those unfamiliar with various languages. Recognizing this importance, many conferences, lectures, and entertainment media have increasingly embraced embedded subtitles. Despite the progress made in generating subtitles, effective subtitle segmentation remains a critical aspect for enhancing content comprehension with minimal cognitive stress.

Currently, there is limited research and a scarcity of open-source libraries addressing subtitle segmentation. While proprietary solutions prevail in the entertainment industry, academic institutions, schools, and conferences often face challenges due to financial constraints. Unfortunately, this results in reliance on proprietary software or, in the case of conferences like Emacsconf, the dedication of valuable volunteer hours.

Our aim is to contribute to this field by developing a reliable subtitle segmentation solution. Various approaches have been attempted in the past; however, our current focus involves delving into the realm of deep learning techniques to further advance our efforts. By doing so, we not only streamline the process for academic institutions but also alleviate the need for volunteers, leading to significant time and cost savings.

2 Related Work

2.1 Sentence Transformers

Sentence Transformers are a modification of the standard Transformer models (like BERT, RoBERTa, XLNet, etc.)[5]. It is very useful for understanding and segmenting text into coherent subtitle units. It allows for easy generation of sentence embeddings that could be utilized for segmenting texts based on semantic similarity or thematic breaks. With it's deep understanding of sentence semantics, we can identify natural breaks in the text where the topic or context shifts, ensuring that each segment is internally coherent.

Documentations: https://www.sbert.net/docs/installation.html

https://github.com/UKPLab/sentence-transformers

2.2 Attention-based Neural Text Segmentation

This paper introduces a supervised method for text segmentation, builds an attention-based bidirectional LSTM model[1]. The model utilizes sentence embeddings learned through CNNs and predicts segments based on contextual information.

It uses CNN to extract rich feature representations from each sentence, learning embeddings of the sentence. Then it uses BiLSTM for analyzing embeddings. It also adds an attention mechanism, allows the model to focus on the most relevant parts of the context when determining segment boundaries.

This paper has a GitHub repo attached:

https://github.com/pinkeshbadjatiya/neuralTextSegmentation/

2.3 Text Segmentation as a Supervised Learning Task

The paper proposed a hierarchical model, consisting of two levels of LSTM networks[3]. The lower level generates sentence representations from word tokens using a bidirectional LSTM. The higher-level LSTM then processes these sentence representations to label each sentence based on whether it signifies the beginning of a new segment. This hierarchical design allows for capturing both the word-level semantics and the broader contextual cues necessary for effective segmentation.

Since we have a sufficiently large and labeled dataset similar to their dataset, framing subtitle segmentation as a supervised learning task could lead to significant improvements in segmentation quality.

This paper has a GitHub repo attached:

https://github.com/koomri/text-segmentation/tree/master

3 Datasets

Data License: CC BY-NC-ND 4.0.

In the raw data, “eol” and “eob” are tokens or placeholders used to represent specific points or breaks in the text. Here’s what they signify:

“eob”: This stands for “end of block”. In the context of subtitles, “eob” indicates the end of a complete subtitle that should be displayed on a single frame of a video. For instance, if a subtitle reads “Hello, how are you?” and this sentence is displayed in its entirety on one screen, the “eob” token would mark the end of this subtitle before the next subtitle appears.

“eol”: This stands for “end of line”. Within a subtitle, “eol” marks the end of a line or segment of text. In a multi-line subtitle, “eol” would be used to indicate where one line ends and the next begins, ensuring the text is displayed in a readable format across the screen.

In the raw transcription, these tokens “eol” and “eob” are used to segment the spoken content into distinct lines or blocks, to facilitate further processing

This data is sourced from the “MuST-Cinema: a Speech-to-Subtitles corpus” paper[2], which focuses on developing automatic solutions for subtitling through Neural Machine Translation (NMT). The corpus, MuST-Cinema, is a multilingual speech translation dataset constructed from TED subtitles. It comprises audio, transcription, and translation triplets, with special symbols inserted to preserve subtitle breaks. The paper emphasizes the need for high-quality, large, task-specific training data for efficient automatic subtitling and proposes a method for annotating existing subtitling corpora with subtitle breaks.

In summary, the provided text serves as a real-world example of the content included in the MuST-Cinema corpus, showcasing natural speech, expressions, and the need to accurately segment sentences into subtitles based on speech duration and content.

4 Approach

4.1 Dataset Cleaning

The MuST-Cinema corpus data exhibited high quality based on the initial manual analytic tests conducted by our team. Initially, when we attempted to pass the tokens into the model, we found that tokenizing using UDPipe yielded less satisfactory results. However, when we combined our model with the tokenization of the pre-trained model, the results showed significant improvement.

In our approach, where we treated the task as token classification, we expanded our understanding beyond individual words and a few breaks, such as ¡eol¿ and ¡eob¿. We assumed that words could exist in two states, namely S1 and S2. The ¡eol¿ tag is only followed by S2, whereas ¡eob¿ is only followed by S1.

We divided the raw data, consisting of 275,086 documents, into training, testing, and validation sets, with proportions of 80%, 10%, and 10%, respectively.

4.2 Video Preprocessing

The video preprocessing stage is crucial in our subtitle segmentation project. It prepares the groundwork for later steps like subtitle creation and automatic segmentation. During this process, the video file is split into two main streams: audio and text.

Audio Extraction

First, we extract the audio track from the video file. This audio is crucial not only for syncing subtitles accurately but also for ensuring that our segmentation matches the spoken content precisely. We use a dependable video processing library to perform this extraction, which helps maintain the quality and integrity of the audio.

Speech-to-Text Conversion

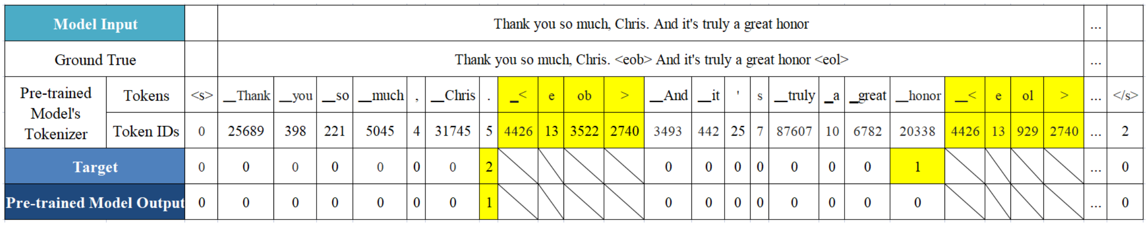
At the same time, we use OpenAI's Whisper model[4] to convert the extracted audio into text. Known for its strong performance across various languages and accents, Whisper provides a high-quality transcription essential for our subtitle segmentation's accuracy. The model processes the audio stream and delivers a textual representation of the spoken content. This transcription is then used directly in the auto-segmentation phase, where it is analyzed and segmented into distinct subtitle units.

4.3 Fine-tuning

We selected the igorsterner/xlmr-multilingual-sentence-segmentation model, a fine-tuned variant of XLM-RoBERTa for multilingual sentence segmentation, as our base. To tailor it for subtitle segmentation, we further fine-tuned the model on the MuST-Cinema dataset to ensure that the placement of <eob> and <eol> markers conforms to the syntactic structure and improves readability.

Data Preprocessing

To fine-tune the model, we adapted the target text containing <eol> and <eob> markers to fit the model's output format, as shown in Figure 1. Using the pre-trained model's tokenizer, we converted the target text into token IDs and then removed the token IDs for <eol> $[4426, 13, 3522, 2740]$ and <eob> $[4426, 13, 929, 2740]$. For the remaining tokens, we labeled the token preceding <eol> as 1, the token preceding <eob> as 2, and all other tokens as 0.



Model Input			Thank you so much, Chris. And it's truly a great honor																							...	
Ground True			Thank you so much, Chris. <eob> And it's truly a great honor <eol>																							...	
Pre-trained Model's Tokenizer	Tokens	<s>	__Thank	__you	__so	__much	,	__Chris	.	_<	e	ob	>	__And	__it	'	s	__truly	_a	_great	__honor	__<	e	ol	>	...	</s>
	Token IDs	0	25689	398	221	5045	4	31745	5	4426	13	3522	2740	3493	442	25	7	87607	10	6782	20338	4426	13	929	2740	...	2
Target		0	0	0	0	0	0	0	2					0	0	0	0	0	0	0	1					...	0
Pre-trained Model Output		0	0	0	0	0	0	0	1					0	0	0	0	0	0	0	0					...	0

Figure 1: Token Labeling

Statistically, the longest sentence in the training set has 300 tokens. To enhance the model's adaptability, we pad all sentences to 330 tokens and generate attention masks for each sentence.

Model Architecture Modifications

We modified the classifier layer of the pre-trained model, originally designed for binary classification (labels 0 and 1), to support a three-class classification task.

Training Hyperparameters

- Seed: 42
- Train Batch Size: 16
- Eval Batch Size: 16
- Learning Rate: 2×10^{-5}
- Optimizer: AdamW with weight decay of 0.0001
- Number of Epochs: 4
- Learning Rate Scheduler: Linear decrease without warmup
- Optimization Steps: Gradient clipping at 4.0, performed every 2 steps
- Class Weights Adjustment: [1.0, 2.5, 2] for labels [0, 1, 2]
- Loss Function: Cross-Entropy Loss, ignoring padded tokens

Training and Testing Results

Figure 2 shows the training and validation loss over four epochs. The training loss decreases steadily from about 0.22 to 0.18, while the validation loss remains stable around 0.17, indicating effective learning without significant overfitting.

Table 1 shows the performance of the model on the test dataset. From the confusion matrix, the model performs best at recognizing class 0, followed by class 2, but faces some

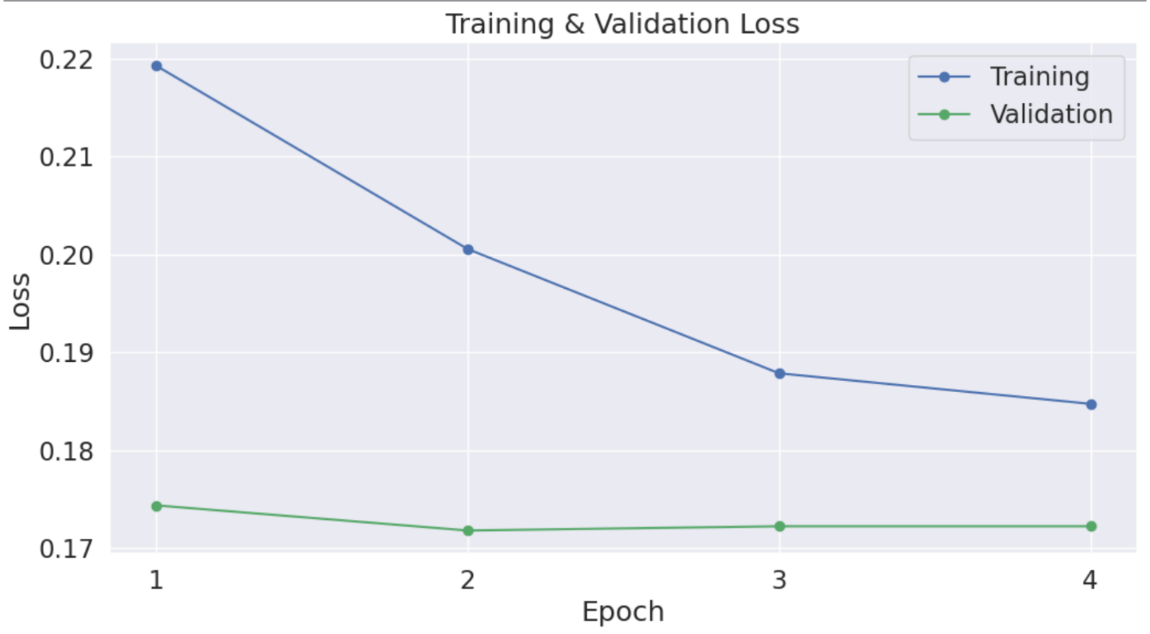


Figure 2: Training and Validation Loss

challenges with class 1, occasionally mislabeling it as class 0.

Table 1: Test Dataset Performance

Metric	**Macro Average**	**Weighted Average**
Accuracy	0.9453	0.9453
Precision	0.7847	0.9492
Recall	0.8196	0.9453
F1 Score	0.8008	0.9470

4.4 Model Application

First, we processed the text generated by Whisper by removing the introductory line ”WEBVTT”, empty lines, and timestamps. Then, we split the remaining text into sentence fragments based on periods. Each complete sentence, ending with a period, served as input for the model. Next, following the same data preprocessing as the training set, we input the data into the model to obtain labels for each token. Finally, we converted the results into text with end-of-line (EOL) and end-of-block (EOB) markers.

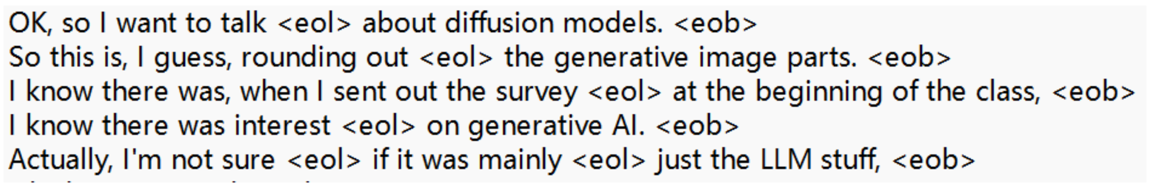

OK, so I want to talk <eol> about diffusion models. <eob>
So this is, I guess, rounding out <eol> the generative image parts. <eob>
I know there was, when I sent out the survey <eol> at the beginning of the class, <eob>
I know there was interest <eol> on generative AI. <eob>
Actually, I'm not sure <eol> if it was mainly <eol> just the LLM stuff, <eob>

Figure 3: Samples of Model Output Results

4.5 Generate Subtitle File

To match the transcribed text with the audio and ensure accurate subtitle timing, we use the Aeneas library. This process involves creating a synchronization map that associates segments of the transcribed text with specific timestamps from the audio file.

Aeneas automatically synchronizes the transcribed text with the corresponding audio segments. This synchronization is crucial for aligning the subtitles accurately with spoken words in the video.

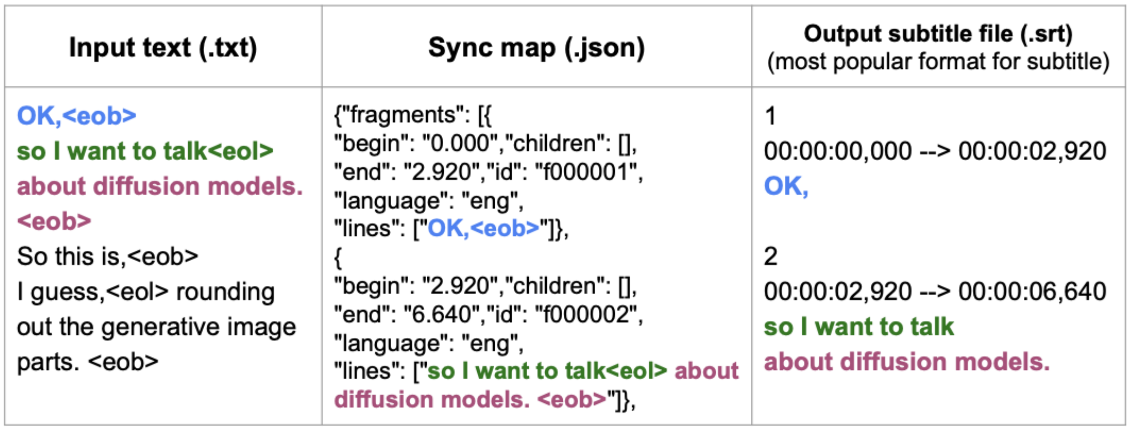
We begin by preparing two inputs: First is a plain text file containing the transcribed text, segmented by <eob> and <eol>, which indicate the breaks for subtitle blocks and lines. The second one is extracted audio file corresponding to the video.

Using the Aeneas tool, we generate a synchronization map. This map is a JSON file detailing the exact start and end times for each segment of text, ensuring each subtitle appears and disappears at the correct moments in the video. Output Generation: The final output is an .srt file, which is the standard format for subtitles. This file integrates the synchronization data with the text, formatting each segment with its corresponding start and end times, as specified in the synchronization map. Figure 1 shows how this process works.

4.6 Evaluation Method

Given that subtitles can potentially have multiple ground truth segmentations, it is not ideal to compare a subtitle with any one given ground truth. Currently, all present models evaluation metrics has this draw back of performing comparisons with a single ground truth. Thus, we have chosen to utilize the simple and straightforward method of evaluation, the F1 score.

During our class, we conducted manual evaluations, which yielded positive results. Our model was evaluated against the popular video editor CapCut, specifically focusing on the subtitles generated for the DS598 lectures. The comparisons were centered around three critical aspects of subtitle quality: timing accuracy, readability, and content accuracy. Both subtitle has a very high accuracy, but our subtitle model demonstrates superior



Input text (.txt)	Sync map (.json)	Output subtitle file (.srt) (most popular format for subtitle)
OK,<eob> so I want to talk<eol> about diffusion models. <eob> So this is,<eob> I guess,<eol> rounding out the generative image parts. <eob>	{"fragments": [{ "begin": "0.000","children": [], "end": "2.920","id": "f000001", "language": "eng", "lines": ["OK,<eob>"]}, { "begin": "2.920","children": [], "end": "6.640","id": "f000002", "language": "eng", "lines": ["so I want to talk<eol> about diffusion models. <eob>"]},	1 00:00:00,000 --> 00:00:02,920 OK, 2 00:00:02,920 --> 00:00:06,640 so I want to talk about diffusion models.

Figure 4: Matching Process

timing accuracy in synchronizing subtitles with the corresponding audio. The readability of subtitles produced by our model was found to be significantly higher than those generated by CapCut. This improvement is attributed to our model's ability to optimally break sentences and phrases into coherent units that fit comfortably on the screen, respecting both linguistic guidelines and user interface constraints.

Looking ahead, as part of our future work, we aim to explore evaluation methods like SubER[6], which do not require ground truth. Additionally, we plan to investigate the possibility of building a framework for human evaluation.

5 Evaluation Results

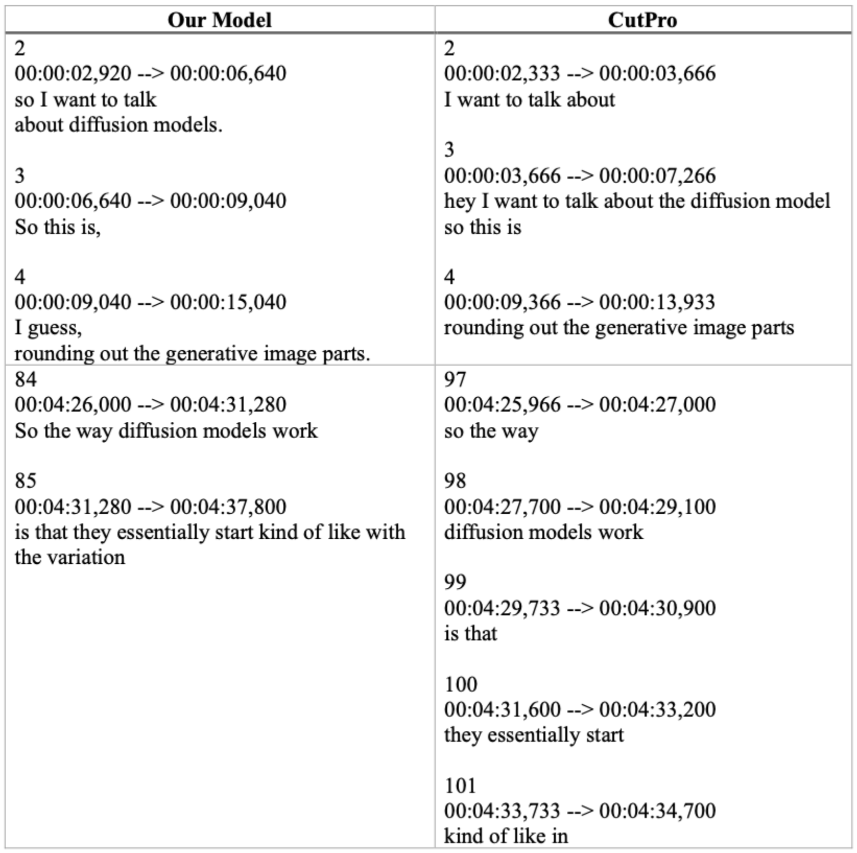
Considering both <eol> and <eob> as a single category break and evaluating whether a break occurs after each word, the resulting F1 score was 0.544.

For the subtitle demo we generated for lectures, we received positive feedback during our manual evaluation phase in a presentation where we compared the results of our model with those of auto-generated by CutPro.

In Figure 5, we compared a few select lines to demonstrate the readability and cohesiveness of each approach.

The subtitles generated by our model are more cohesive and structured, leading to better readability. Key differences include:

- **Consolidation of Text:** Our model combines related phrases into fewer, longer segments (e.g., combining the initiation of the topic into one subtitle rather than



Our Model	CutPro
2 00:00:02,920 --> 00:00:06,640 so I want to talk about diffusion models. 3 00:00:06,640 --> 00:00:09,040 So this is, 4 00:00:09,040 --> 00:00:15,040 I guess, rounding out the generative image parts.	2 00:00:02,333 --> 00:00:03,666 I want to talk about 3 00:00:03,666 --> 00:00:07,266 hey I want to talk about the diffusion model so this is 4 00:00:09,366 --> 00:00:13,933 rounding out the generative image parts
84 00:04:26,000 --> 00:04:31,280 So the way diffusion models work 85 00:04:31,280 --> 00:04:37,800 is that they essentially start kind of like with the variation	97 00:04:25,966 --> 00:04:27,000 so the way 98 00:04:27,700 --> 00:04:29,100 diffusion models work 99 00:04:29,733 --> 00:04:30,900 is that 100 00:04:31,600 --> 00:04:33,200 they essentially start 101 00:04:33,733 --> 00:04:34,700 kind of like in

Figure 5: Comparison between Our Model and CutPro

splitting it across multiple short and fragmented pieces). This reduces cognitive load on viewers, who have to process less visual interruption.

- **Punctuation and Clarity:** Our model uses punctuation effectively to separate ideas, which helps in understanding the structure of sentences and enhances the clarity of the spoken content.
- **Elimination of Redundancies:** Our model streamlines the text to remove unnecessary interjections and redundancies that can distract from the main content.

6 Conclusion

Our project is designed to address the issue of accurately segmenting subtitles, trying to enhance both the readability and accessibility in environments where specialized tools are not available. We aim to deploy deep learning techniques such as BERT and LSTM within a framework of unsupervised learning. Our final vision is to offer a user-friendly tool where individuals can upload video files or YouTube links and receive accurately segmented subtitles. With additional features like subtitle-video merging and speech quality evaluation available through command-line options, we aspire to make our deliverables widely accessible. Committed to the open-source license, we ensure the use of open-source code and libraries, aligning with our goal to make this project freely available to all.

6.1 Advantage and Disadvantages

Our subtitle segmentation model has demonstrated significant advancements in the field of automated subtitle generation, particularly in terms of processing speed, timing accuracy, readability, and content accuracy. The system greatly outperforms traditional video editors like CapCut, offering a much faster processing time — 50 seconds total compared to three minutes. This efficiency can be a major advantage in scenarios where quick turnaround of video content is critical, such as academic references or lecture content creation.

However, our tool currently presents a steep learning curve due to its reliance on coding skills. While this is manageable within a technical context, it could be a barrier to deploy for users in less technical fields or for casual users who require subtitling services. This requirement restricts the accessibility of our tool to a broader audience, which is critical for widespread adoption.

6.2 Future Improvements

Looking forward, there are several key areas where our project could evolve to not only enhance its functionality but also its user-friendliness and applicability:

1. **Web Integration:** Integrating our subtitle system into a web page or a standalone website would make our tool accessible to a wider audience, eliminating the need for coding skills and allowing users to input their video and receive subtitles directly through a user-friendly interface.

2. **Text Summarization Feature:** Adding a text summarization function would provide immense value, especially for educational or lengthy content. However, implementing this feature faces challenges due to the text window limitations of open-source large language models (LLMs). Addressing these limitations to ensure efficient summarization across longer content would be a crucial area of development.

3. **Enhanced Evaluation Techniques:** Developing a more comprehensive subtitle evaluation scheme is essential. Current methods predominantly rely on comparisons with a ground truth, which may not fully capture the qualitative aspects of subtitle usability and viewer experience. A more nuanced approach that includes user interaction and satisfaction metrics could provide deeper insights into the effectiveness of the subtitles.

References

[1] Pinkesh Badjatiya, Litton J Kurisinkel, Manish Gupta, and Vasudeva Varma. Attention-based neural text segmentation, 2018.

[2] Alina Karakanta, Matteo Negri, and Marco Turchi. Must-cinema: a speech-to-subtitles corpus. *CoRR*, abs/2002.10829, 2020.

[3] Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. Text segmentation as a supervised learning task, 2018.

[4] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[5] Nils Reimers and Iryna Gurevych. 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.

[6] Patrick Wilken, Panayota Georgakopoulou, and Evgeny Matusov. Suber: A metric for automatic evaluation of subtitle quality, 2022.