Deep Learning YouTube Sponsor Blocker

Justin Shin

George Mason University

ABSTRACT

SponsorBlock is a browser extension that allows the user to automatically skip YouTube sponsor ads, self-promotions and more. This is accomplished by the help of the crowdsourced database where volunteer contributors have submitted hand-labeled time-stamped segments using the extension. The motivation for this project was to automate the labeling by fine-tuning a pretrained transformer model trained on YouTube video transcript data. The architecture is composed of two specialized models: one that performs Topic Segmentation on the YouTube transcripts and one that performs Classification on the resulting segments. The Matthews Correlation Coefficient (MCC) was the metric used to quantify the quality of the classification. Table 1 shows that the RoBERTa model yielded the best MCC score for Topic Segmentation which was also used to yield the high accuracy of the Text Classification model shown in Table 2.

Table 1. Results (Segmentation)

	BERT	RoBERTa	XLNet	XLM_RoBERTa
MCC	0.5826	0.6131	0.5472	0.5947
TP	1720	1750	1535	1703
TN	1755	1792	1847	1797
FP	423	386	331	381
FN	495	465	680	512

Table 2. Results (Classification)

MCC	0.8785

Deep Learning YouTube Sponsor Blocker

Justin Shin

George Mason University

1. Introduction

SponsorBlock is a web browser extension that allows the user to automatically skip YouTube sponsor ads in realtime during video playback. It is a crowd-sourced effort to accurately label sponsor ad segments to allow the extension to skip over these segments. Users can vote on the accuracy of the sponsor segment on whether they were placed accurately or not, in which case, with sufficient downvotes, the segment becomes deactivated. Over the years, SponsorBlock has extended its segmenting feature that allows labeling not only sponsor ads but also self-promotions, call-to-actions (where the YouTuber asks for likes and subscriptions), intros and end credits, and more. This process of labeling the segments have been done by volunteers contributors since the project's inception totalling over 14 million entries, but with ready access to the SponsorBlock database through their free API, as well as access to YouTube transcripts (though limited), this process can be automated using Deep Learning Neural Networks. Coincidentally, with the recent popularity of highly performant large language models from such examples as GPT4 and Llama, this process of automating the labeling can be greatly improved. The goal of this project is to build a model that can predict those sponsor segments as accurately or more accurately than a human contributor.

2. Motivation

The idea for this project came from my daily personal use of the SponsorBlock browser extension. Over the years, I have become increasingly intolerant of any type of advertisement especially of the type that are sneakily embedded in your average internet content, namely sponsor ads. I use uBlock Origin to block pre-roll (plays before a video) and mid-roll (plays in the middle of a video) video ads and SponsorBlock for sponsor ads. To date, the extension has skipped 1,853 segments for me, which totals nearly 16 hours of time saved. While working on this project, I have also contributed to the crowd-sourced database which has helped users save a total of 2.5 hours. I realize the powerful utility of this browser extension but there are a few shortcomings from relying solely on human contributors.

Firstly, the majority of YouTube videos have zero segment contributions. This is obvious because most YouTube viewers are not SponsorBlock viewers and therefore would not have contributors to label them. But this affects a SponsorBlock user

Secondly, when a new YouTube video is released, a SponsorBlock user must wait for a contributor to find those segments before they can enjoy the video sponsor-free. Though, SponsorBlock contributors are very quick to label the newly released videos of the most popular YouTube channels, smaller channels often take days.

And finally, oftentimes, a contributor will only label some of the segments and will miss others. This may be the case because the contributor was only motivated to label one segment only. It is also likely that they simply did not come upon the segment and have completely missed them.

A common problem among the above-mentioned shortcomings is that SponsorBlock is completely dependent on human contributors and utterly fragile to human error. The obvious solution is to run an automated labeling process before watching any YouTube. This was the motivation for this project.

3. Related Work

There have already been a few attempts to solve this problem with Deep Learning. Most of these projects were not very serious projects and were just one-offs. Four of the most notable projects were looked into to aid in the development and provide inspiration to this project. most of these don't have performance metrics since they weren't done for research but rather as simply a solution to the problem.

3.1. NeuralBlock

NeuralBlock [1] was written by Andrew Lee and was the first project to be recognized by the creator of SponsorBlock. It was used as a moderation tool that verified the correct timestamps of user submissions. It is currently not in use due to YouTube's rate-limiting the server from downloading too many transcripts.

NeuralBlock used a bidirectional-LSTM RNN that trained on the transcripts downloaded from YouTube. It tokenized the top 10,000 words found in the sponsor segments to train the model. The RNN was able to predict the start and end timestamps of a sponsor segment. This project is not in active development.

3.2. DeepSponsorBlock

DeepSponsorBlock [2] was a Stanford C230 project that used the novel approach of parsing the frames of the YouTube video to predict the sponsor segments instead of using the transcript. All previous methods were based on training a model on the transcripts only. The creators mentioned that the advantage to this method was that it can predict sponsor segments regardless of language as it is not dependent on natural language. Though this would be very effective for non-English users, the vast majority of the SponsorBlock submissions are from English YouTube videos.

The architecture consists of a CNN that extracts features from the video frames and an RNN that predicts the start and end timestamps for the sponsor ad. One limitation is that the dataset only consisted of videos that had exactly one sponsor ad, where the average YouTube video may have two or more sponsor ads. The resulting model suffered from overfitting with an accuracy of 79%.

3.3. reBlock

reBlock [3] was the winner of a hackathon competition and thus has not been worked on since the win. It was built by fine-tuning a pretrained RoBERTa [4] model that performed token classification on YouTube transcripts. The model was trained for 7 hours on 4 Geforce GTX 1080 Ti's on over 31,000 video transcripts.

3.4. SponsorBlock-ML

SponsorBlock-ML [5] is currently the best performing solution and is currently used to assist in labeling. Unlike the models previously mentioned above, this model is the first to support multi-classification supporting more than just classification of sponsor ads. This model is composed of two fine-tuned transformer models. The first model is a pretrained t5 [6] model that takes a YouTube transcript as input and performs a summarization task which compresses it down to only the sponsor segment in the transcript. Then the segment is fed into the second model which classifies the segment as one of four possible types: sponsor, call-to-action (where the YouTuber asks for likes and subscriptions), self-promotions, or none of the above. The models were trained on a dataset with over 400,000 rows of data generated from YouTube transcripts.

4. Method

After the preliminary research on the current existing solutions, it was decided that this project's model also needed to be able to identify more than just sponsor ads. SponsorBlock supports labels for sponsor ads, call-to-action, self-promotions, and more. Like SponsorBlock-ML this project needed to also support identifying the three mentioned labels above. With ready access to the SponsorBlock API and a YouTube transcript API, the dataset was generated and used to train the models.

4.1. Prototype

The motive for the first version was to understand how to build and train a transformer using available tools. It was also an exercise to learn about how to structure the dataset for producing a performant model. The python package called "SimpleTransformer" was used to build and train these models. It is a wrapper for the popular Hugging Face "Transformer" package that allows for simpler and quicker development of a fine-tuned transformer.

4.1.1. Prototype Dataset

The dataset evolved through the stages of the project. To create the first version of the model, a small test dataset was generated from the SponsorBlock and YouTube transcripts API's. This process involved first sorting the SponsorBlock API by most voted entries, then downloading the appropriate transcripts of said entries, and finally, identifying the corresponding timestamps from those downloaded transcripts. One note of caution for those attempting to replicate this project: YouTube does not take kindly to excessive downloads and may throttle or ban the IP address used during the download of the transcripts. This was the fate of NeuralBlock, mentioned above, and has effectively been put out of service because of the excessive transcript downloads.

The prototype dataset was generated from 1,154 transcripts. This dataset consisted of 5,769 rows of individual transcript lines with a label of 0 or 1 denoting whether the line belonged in a sponsor ad or not, respectively.

4.1.2. Prototype Model

The first version of the model was a pretrained RoBERTa model that was fine-tuned to perform Text Classification on the transcripts. It was designed to parse a transcript one line at a time and classify it as either being part of a sponsor ad or not. It did not support other categories of labels like call-to-actions or self-promotions and had a smaller dataset to train on. There were very clear problems with this design, for example, it is favorable for the model to minimize the number of identified segments in a transcript. Because the model parsed the transcript at the line level, it was possible for the model to identify every other line as a sponsor ad, resulting in a large number of sponsor segment predictions. This problem could easily be alleviate by parsing the transcript by multiple lines at once, hence, moving to have the model perform Topic Segmentation forcing it to segregate the full transcript to a more manageable minimum number of segments.

The learning rate was set to 4e-5, the optimizer used was AdamW, and the model was trained at 5 epochs; any higher resulted in incredible overfitting.

4.2. Final Architecture

The final architecture including two models: one for Topic Segmentation and the other for Text Classification. The Topic Segmentation allowed to divide the whole transcript into smaller topic segments (sponsor, call-to-action, etc.). The Text Classification predicted whether these segments were classified with the correct topic. The motive for this final version was to improve on the problems found in the prototype model.

4.2.1. Final Datasets

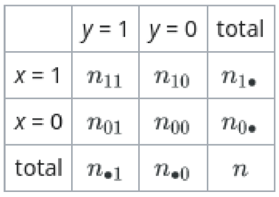
The datasets used to train the final iteration of this project's model increased nearly 5 times which consisted of over 5,000 YouTube transcripts. The Topic Segmentation model was trained on concatenating two transcript lines and determining whether the topic change fell between these two lines. Seeing as there were more non-topic changes than topic changes, the dataset required severe undersampling, where the number of topic changing lines matched the number of non-changing lines. The original dataset for Topic Segmentation consisted of more than 2 million rows but after undersampling, the total number of samples was brought down to just 4,393. The Text Classification model required segments of multiple lines in a transcript labeled correctly with one of four labels: sponsor, call-to-action, self-promotion, or none of the above. This effectively makes this a Multiclass Classification Model. The dataset consisted of 10,992 segments.

4.2.2. Final Models

For the final version, the architecture consisted of two models, one transformer that performs Topic Segmentation and classifies a group of lines from a transcript from one of the four labels: sponsor ad, call-to-action, self-promotion, or none of the above. The Topic Segmentation model was tried by fine-tuning four pretrained transformer models: BERT, RoBERTa, XLNet, and XLM-RoBERTa. The BERT model was set as the baseline model as 2 out of 3 tried models were derivative improvements on the original BERT model. RoBERTa was chosen as it is commonly praised as a better performing model but a more costly one as well. XLNet was suggested by the authors that provided some inspiration for using Topic Segmentation for this project [7]. XLM-RoBERTa was trained on a multilingual dataset and was tried to see how well it's English computations performed. The Text Classification model was created by fine-tuning a pretrained RoBERTa model. RoBERTa yielded promising results from the previous Prototype study. It did not have the same rigorous ablation study though, due to time constraints as well as the Topic Segmentation model requiring more attention from its lacking performance.

5. Results

5.1. Evaluation Metric



	$y = 1$	$y = 0$	total
$x = 1$	n_{11}	n_{10}	$n_{1\bullet}$
$x = 0$	n_{01}	n_{00}	$n_{0\bullet}$
total	$n_{\bullet 1}$	$n_{\bullet 0}$	n

Figure 1. MCC Table

The model performance was evaluated using the Matthews Correlation Coefficient (MCC) which is a form of Precision and Recall metric that takes the True Positive, True Negative, False Positive, and False Negative as input as shown in Figure 1. The method to calculate the score is shown in Figure 2. The score value falls between a range from -1 to 1 where 1 means perfect prediction, 0 means as good as a random prediction, and -1 means an inverse prediction. The purpose is to get a score as close to 1 as possible. Both the Text Classification models from the Prototype and Final architectures as well as the Topic Segmentation model use the MCC as a performance metric.

$$\phi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1\bullet}n_{0\bullet}n_{\bullet 0}n_{\bullet 1}}}.$$

Figure 2. MCC Equation

5.2. Preliminary Results

As shown in Table 1, the prototype performed with a lacking score of 0.4625 which falls below the halfway point in the MCC between perfect prediction and random prediction. This score means that the model performs better than a random guess only half the time. While it is better than a score of 0, there was much to improve on this model.

Table 1. Preliminary Results

MCC	0.4625
TP	434
TN	410
FP	156
FN	154

5.3. Final Results

The final results from the Topic Segmentation model is show in Table 2 and Text Classification in Table 3. The Topic Segmentation performed best with the pretrained RoBERTa model. RoBERTa is known to perform better than BERT but at the cost of being more expensive to train. The XLNet model did not perform as well perhaps because it was not as sensitive to the training dataset as RoBERTa. XLM-RoBERTa is simply RoBERTa with multilingual support so it may not have performed as well as base RoBERTa because it was not as sensitive to English. The Text Classification model performed well with an MCC score of 0.8785. No further ablation studies were performed on the Text Classification model since reaching this acceptable score.

Table 2. Final Results (Segmentation)

	BERT	RoBERTa	XLNet	XLM_RoBERTa
MCC	0.5826	0.6131	0.5472	0.5947
TP	1720	1750	1535	1703
TN	1755	1792	1847	1797
FP	423	386	331	381
FN	495	465	680	512

Table 3. Final Results (Classification)

MCC	0.8785

6. Future Work

6.1. Pk Metric

The evaluation metric used for this project was the Matthews Correlation Coefficient which works as a type of Precision and Recall metric. The problem with using this metric is that it is not appropriate for imperfect classification tasks like Topic Segmentation. Figure 3 shows two example models called A-0 and A-1 each showing their Topic Segmentation predictions. When compared to the ground truth, Ref, A-0 performs better than A-1 but if MCC is used to compare their performances, they would equally be penalized for an imperfect classification despite A-0 only being minimally off from the ground truth.

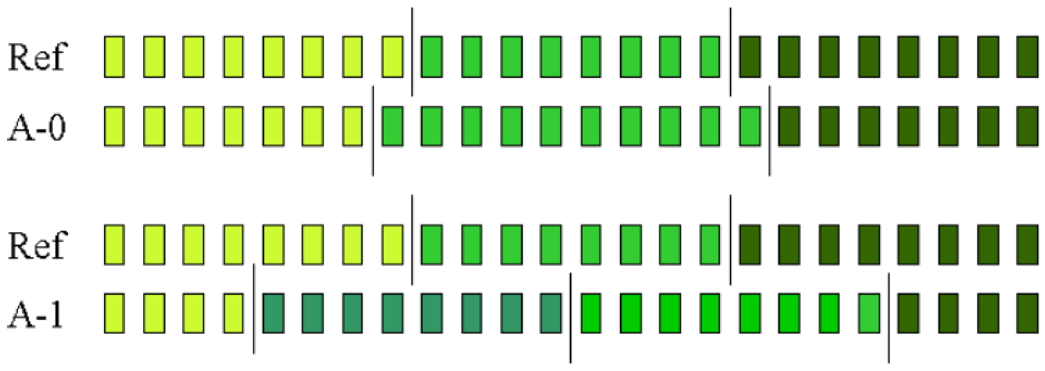


Figure 3. Matthews Correlation Coefficient

The most appropriate metric to use against Topic Segmentation is the Pk score. First introduced by Beeferman et al [8], the Pk score is computed by using sliding window to compare the first and last index of both the model and the ground truth. With each mismatching indices, the model is penalized and incremented 1 point where the lower the score the better the performance. The example in Figure 4 demonstrates how this gives a more accurate score than MCC. The sliding window starts at index 2 with a window size of 5 and slides 5 times. With the exception of the first comparison, the Hyp model is only penalized the 1 point. Evidently, if this sliding window was used to compare A-0 and A-1 from Figure 3, A-0 would have a much lower score than A-1.

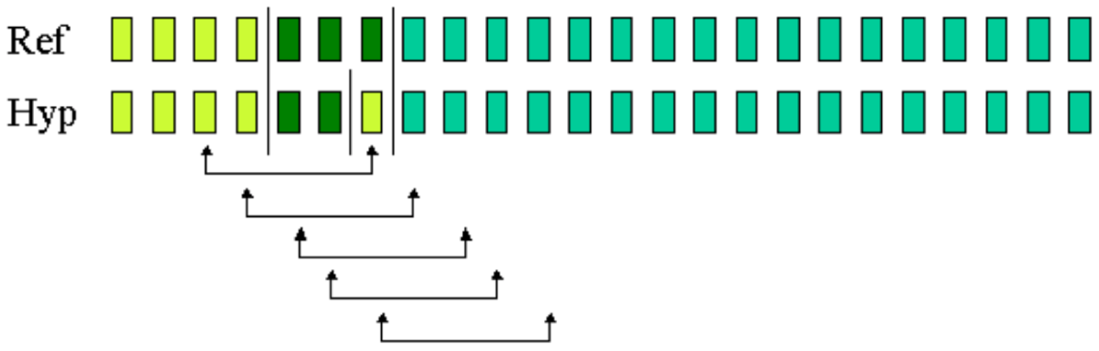


Figure 4. Pk

6.2. Sentence-BERT

Sentence-BERT (SBERT) [9] is a modified BERT model which uses siamese and triplet BERT networks to derive meaningful sentence embeddings. This results in a more accurate and context aware model which is achieved by comparing cosine-similarity between two sentences. This also yields faster computations where it would take 65 hours for BERT but only 5 seconds for SBERT while still maintaining similar accuracy rates. Seeing as the current implementation of Topic Segmentation is highly dependent on the model's understanding at the sentence level, experimenting with SBERT may yield exponential performance improvements.

6.3. Training on Additional Data Sources

This model can also be improved on by introducing more varying data points for the model. For instance, it can follow in the steps of DeepSponsorBlock and incorporate the training of video frame data. This would result in a model that is more sensitive to portions of a video that contains no spoken words especially during music playback. The model would be able to differentiate between music that belongs in a sponsor segment or not. The obvious to including video frame data is how much more costly and difficult it is to acquire the data. YouTube's rate limitations may be stricter with downloading image data compared to just downloading text transcripts. Nevertheless, image data would greatly improve the accuracy the Topic Segmentation model.

The audio can also be used to train the model. The advantage of audio data is the ability for the model to capture volume changes as well as silences to accurately classify topic transitions. Incorporating both the video frames and audio would drastically improve the Topic Segmentation allowing the model to greatly improve its discernment against topic changes.

7. Conclusion

The purpose of this project was to automate the labeling of sponsor segments of YouTube videos. This was successfully accomplished by generating a dataset from scratch from 5000 YouTube transcripts to train a Topic Segmentation and Text Classification model. The role of the Topic Segmentation model is to compartmentalize a full YouTube transcript into segments. These segments are then fed into the Text Classification model to determine the correct category so as to identify sponsor ads. It is currently not as performant as a human contributor but by extending it's capabilities with the above-mentioned improvements, it can perform better than a human contributor by labeling more videos at unparalleled speeds and with high accuracy. One drawback with all the mentioned related works, with the exception of SponsorBlock-ML, is that none are in active development, and many have been abandoned by its first git commit. Deep Learning YouTube Sponsor Blocker will not only be improved on by the aforementioned future work but also by maintaining the project longer than any of its predecessors. This will successfully yield improvements unlike the past attempts and may someday perhaps replace NeuralBlock and SponsorBlock-ML as the official SponsorBlock moderator.

References

1. Andrew Lee, *NeuralBlock* (2021). https://github.com/andrewzlee/NeuralBlock.
2. Nikhil Athreya, Cem Gokmen, and Jennie Yang, *DeepSponsorBlock* (2021). https://github.com/DeepSponsorBlock/DeepSponsorBlock.
3. David Li and Jonathan Li, *reBlock* (2022). https://github.com/MonliH/reBlock.
4. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *CoRR* (2019).

5. Joshua Lochner, *SponsorBlock-ML* (2023). https://github.com/xenova/sponsorblock-ml.
6. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *CoRR* (2019).
7. Kelvin Lo, Yuan Jin, Weicong Tan, Ming Liu, Lan Du, and Wray Buntine, "Transformer over Pre-trained Transformer for Neural Text Segmentation with Enhanced Topic Coherence," *Findings of the Association for Computational Linguistics: EMNLP 2021* (2021).
8. Doug Beeferman, Adam Berger, and John Lafferty, *Machine learning,* Springer (1999).
9. Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks},," *CoRR* (2019).