Youtube Podcast Topic Segmentation

John Calzaretta, Matt Pribadi, Ricardo Marin

Instructors: Peter Grabowski, Daniel Cer

Abstract

This project aims to use natural language processing to divide podcast episodes into topic segments with timestamps. We primarily reference two previous works: one unsupervised topic segmentation approach on a corpus of meeting transcripts, and one supervised approach on a corpus of Wikipedia articles. This paper discusses the implementation of both unsupervised and supervised topic segmentation methods on a corpus of podcast transcripts from Youtube videos. All model variants are evaluated using the WindowDiff and Pk error metrics. Ultimately, the supervised learning method using transformer encoders produced the best results and, in many cases, produced practically useful podcast topic segments.

1 Introduction

With a vast and expanding library of podcasts, topic timestamps allow users to quickly identify which sections of each episode are worth listening to and which are worth skipping, saving the user time and increasing the quality of average content consumed. Despite the value provided to consumers, most of the podcast library does not provide them - likely due to the time required to manually parse an episode. This research aims to provide a tool for podcasters to quickly input their podcast transcript and receive a set of reliable topic segments for users to navigate across.

1.1 Data

Our dataset consists of transcripts of over 3,500 podcasts from YouTube videos. Our team created this dataset using the YouTube Data API v3 from Google to scrape the auto-generated transcripts and topic segment timestamps from videos in the YouTube Top 100 podcast channels page (YouTube Data API). A key challenge is that most YouTube transcripts are auto generated using automatic speech recognition (ASR). Transcription errors reduce signal and add noise to the dataset, and the lack of punctuation makes defining sentences difficult. We explore a set of heuristics in which each slice of W words is defined as one sentence (where W = 10, 30, 50). The below analysis uses W = 30 as the other splitting methods were not effective in modeling - too few words create a highly sparse target sequence, too many words obscures the meaning of each sentence. SpaCy tools were also explored as a sentence splitter but proved too noisy when used in modeling. For both modeling approaches, the inputs, **S,** are at the episode level and include a list of M utterances. The target labels, **Y**, are binary digits of length, **M**, where 1 indicates the beginning of a new topic.

$$S = \{S_1, \ldots, S_m\}, Y = \{Y_1, \ldots, Y_m\}, Y \in [0,1]$$

Supervised methods are less common in the speech segmentation space due to a lack of abundant, labeled datasets. To effectively utilize deep learning tools, we used our scraped dataset to create a much larger dataset of "synthetic" podcast episodes. A synthetic episode is created by slicing each actual episode into its actual topic segments, then repeatedly shuffling across the corpus of topic segments, and concatenating topics from different episodes until a max sequence length is achieved. We believe that the model should be able to learn where topic transitions more easily since the topics of different episodes should be more semantically different than two consecutive topics in the same episode.

With a total corpus of about 3,500 scraped podcast episodes across about 20 YouTube channels, we randomly split 50% of our data into train and test sets. This was a necessary step for the supervised model training but was also used to tune parameters of the unsupervised approach. The train set alone (around 1,700 records) was used to generate over 50,000 new synthetic episodes.

1.2 Related Work

Topic segmentation is a fundamental natural language processing task that has been explored across many domains, beginning with written text and recently expanding to more complex spoken language tasks. Spoken language has proven to be a greater challenge than segmenting written text due to both the reliance on automatic speech recognition (ASR), which introduces complexity and noise to datasets, and also due to the dynamic nature of dialogue compared to the more structured written form.

Unsupervised methods are commonly utilized in the text segmentation domain due to a lack of abundant labeled data. These methods typically fall into the categories of similarity-based methods, or probabilistic methods. An example of a similarity-based segmentation method utilizes BERT sentence embeddings and cosine similarity to segment meeting dialogue in topics (Solbiati et al., 2021). A key assumption with such similarity-based methods is that sentences belonging to the same topic segment are more like each other than to sentences in other segments.

Supervised approaches often result in strong performance but are reliant on a large amount of labeled data. A leading supervised segmentation method was proposed by Koshorek et al. (2018), who implement a bi-directional LSTM to segment wikipedia documents.

2 Methods

All models built rely on pre-trained transformer models to extract sentence embeddings from podcast transcripts. The unsupervised approach uses the sequence of embedded sentences and for each timestep, it calculates the semantic similarity with the previous sentence. In development of the unsupervised approach, we explored different pretrained embedding models, similarity formulas, and thresholding logic to optimally segment podcast topics. The supervised approach attempts to use a recurrent neural network to automatically derive features across the sequence and classify topic transitions.

2.1 Sentence Embeddings

Developing an effective topic segmentation model requires converting a podcast transcript of words into meaningful numeric features that capture contextual information. The standard approach in text segmentation is to use sentence embeddings as the unit of analysis rather than word embeddings, which are computationally expensive. In our analysis, we explored the use of Sentence BERT and the Universal Sentence Encoder, which have produced effective results in other text segmentation domains (Henderson et al., 2019, #). See **Table 1** in **Appendix A** for a summary of each embedding method explored.

2.2 Control Model

To serve as minimum performance benchmarks, we are using four main control models to help prove the value of our models: None, Even, and Random. The None baseline returns a predicted array of all 0s. The Even baseline takes the average length of topic segments, **N** in a corpus and sets a topic boundary every **n** index within the array. The Random baseline sets a boundary at random index positions in an array up to the number of boundary changes in the reference topic.

2.3 Unsupervised Model

We based our unsupervised approach on the work of (Solbiati et al., 2021), who use cosine similarities between neighboring sentences to infer transitions. First, each sentence is compared to the preceding timestep using cosine similarity. For each input transcript, the classification threshold t is determined by the difference of the mean (mucs) and Z standard deviations (sigmacs) of each input's cosine similarity with its neighbor. Any sentence whose cosine similarity is less than t is classified as a topic transition.

$$t = mu_{cs} - (Z * sigma_{cs})$$

We first implemented this approach on the podcast corpus to little success. Conversations are dynamic and full of tangential comments so instead of comparing only two sentences, we decided to measure how similar each sentence is to its several preceding neighbors, and one following sentence. If the given sentence is truly a topic transition, one would expect that it should be semantically different from each of its several preceding neighbors, and like the following neighbor.

Another problem was that the model initially predicted far more topics than the reference sequence. To tune the classification threshold to better suit the podcast domain, we tuned the Z parameter in the equation above. Whereas Solbiata et al. use Z=1, we determined the optimal Z value for each episode in the train dataset. The mean of the train Z values served as our new Z value when creating the classification threshold for each test example. For the YouTube podcast dataset, the mean optimal Z value for the train episodes was 1.8. The distribution of Z-values in the train set is shown in **Figure 1** in Appendix A

2.4 Supervised Model

The creation of a larger dataset consisting of synthetic podcast episodes enables the supervised model, which unlocks two key advantages. First, the use of RNNs enables the model to put each sentence in proper context by looking across the entire sequence of sentences in the episode, rather than a small window of sentences in the unsupervised method. Second, whereas the unsupervised method looks backward by one sentence in the cosine similarity calculation, RNNs can look forward and backward in the episode when deriving the context for each timestep. The set of RNN models developed and evaluated include the LSTM, Bi-directional LSTM, and Transformer encoder models. Though the Bi-LSTM models are common approaches highlighted in previous literature, the Transformer model proved to be the strongest performer in terms of PK and WindowDiff metrics. **Figure 2** details the architecture specifications and training parameters of the final transformer model. Given the sparsity of transition labels, we used a weighted sigmoid cross entropy loss function, which allows for a positive class weight pw to be passed to penalize false negatives in training.

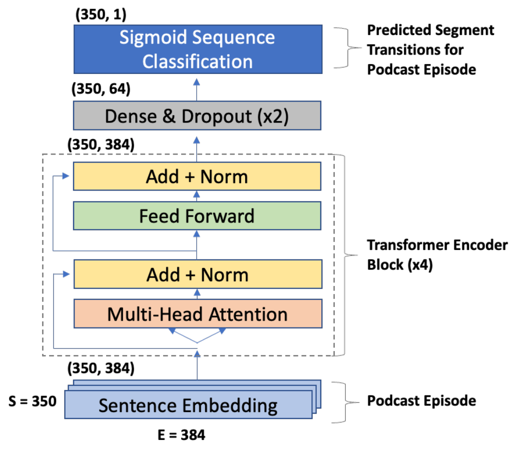

Figure 2 Transformer Model Architecture Diagram. The transformer sequence-to-sequence model ingests a sequence of sentence embeddings for one podcast episode and outputs a binary sequence where each sentence is classified as either a transition (1) or belonging to the same segment as the preceding sentence (0)

The final model architecture uses a **pw** value of about 30, meaning a false negative is penalized 30 times more than a false positive. An additional rule was applied to the output predictions such that no two neighboring timesteps are predicted to be transitions. It is very uncommon in podcasts to have a one sentence long topic, and making this adjustment to predictions improved performance.

2.5 Evaluation

Speech segmentation problems require more flexibility in performance evaluation due to language ambiguity. If a predicted topic transition is one sentence away from the actual, this should be measured as a partial-hit, and not a complete miss as accuracy measures would indicate. Two evaluation metrics are common in related work and allow for near misses: error-probability (Pk) and Window-Difference (WinDiff). In addition to the window-based methods, we also evaluate the mean absolute error between the predicted count of topics and reference count of topics for each episode.

2.6 Error-Probability (Pk) Score

The error-probability score is based on a sliding-window method that compares actual and predicted transition labels for each window. The sliding window is of length k, which is best calculated as the average number of sentences per topic divided by two (Doug et al., 1999). As the window slides across each episode, a counter is incremented when there is any predicted topic transition but no reference transition (false positive), and vice versa (false negative). The final metric measures the rate of error per window, and is weighted to penalize false negatives more than false positives due to the sparsity of topic transition labels. A key challenge of the Pk score is that it is insensitive to multiple boundary changes that could occur within a single window (Doug et al., 1999).

2.7 Window-Diff (WD) Score

The WD score is like Pk except it also aims to solve the multiple boundaries challenge. It uses the window-sliding methodology as described above, but now compares the number of reference and predicted topic transitions occurring within the window (Pevzner & Hearst, 2002). Like Pk, the lower the WD score the closer the predicted model topic transitions are to the reference labels. The calculation can be described through the equation below

$$WindowDiff(ref, hyp) = \frac{1}{N-k}\sum_{i=1}^{N-k}(|b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k})| > 0)$$

where b is the count of the boundaries within a window of length k, and N is the total number of windows per episode (Pevzner & Hearst, 2002).

3 Results and Discussion

In this section, we first summarize the results of top performing supervised and unsupervised models in comparison to the control models in **Table 2** below. Next, we dive into each modeling approach and investigate the errors in greater detail. In the end, the supervised models proved more effective for the podcast topic segmentation problem. The improved performance is likely due to the ability of the transformer encoder model to understand the context of a large sequence of sentences and derive features that are more useful for classification than cosine similarity alone. Cosine similarities provide some signal for unsupervsied models, but they are limited in their ability to understand context and are not highly correlated with transitions

3.1 Discussion of Supervised Model Results

Initial tuning runs were conducted to find the optimal positive class weight **pw** (around 30) and learning rate (0.001). From here, the different model architectures were evaluated, and some variations of architecture sizes were compared. **Table 3** compares the performance of the various supervised models evaluated.

We speculate that the larger Transformer model was better able to extract meaningful features from the sequence of sentences in each episode. The LSTM models struggled in training due to exploding gradients and failed to learn much when the training parameters were weakened in order to avoid the gradient explosion. The transformer model seemed to make targeted predictions of transitions, rather than blanket predictions in the LSTM models.

Figures 2 to 4 in **Appendix A** display the distributions of episode-level performance of the supervised transformer model, with the test mean labeled in the title and visualized with an orange bar.

Table 2 Comparison of top performing unsupervised and supervised methods

Model Configuration				**Test Metrics**		
Model Name	**Method**	**Embeddings**	**Z-Value**	**PK**	**WD**	**MAE**
S_Transformer	Supervised	SBERT-Mini	NA	**0.329**	**0.369**	**0.519**
US_SBN	Unsupervised	SBERT-Mini	1.67	**0.4586**	**0.4855**	**0.576**
None	Control	NA	NA	**0.4338**	**0.4249**	**-**
Even	Control	NA	NA	**0.4856**	**0.4991**	**0.1335**
Random	Control	NA	NA	**0.4848**	**0.524**	**-**

Table 3 Comparison of supervised models on the test dataset

Model Configuration			Test Metrics		
Model Name	**Layers**	**# Params**	**PK**	**WD**	**MAE**
Transformer Large	- Transformer Encoders x4 - Dense & Dropout x2 - Sigmoid Binary Classification Layer	4,801, 549	**0.346**	**0.404**	**0.72**
Transformer Small	- Transformer Encoders x2 - Dense & Dropout x1 - Sigmoid Binary Classification Layer	856,329	**0.394**	**0.421**	**0.91**
BiLSTM	- Bi-LSTM Layers X2 - Dense & Dropout X2 - Sigmoid Binary Classification Layer	663,860	**0.562**	**0.513**	**2.43**
LSTM	- LSTM Layers X2 - Dense & Dropout X2 - Sigmoid Binary Classification Layer	414,913	**0.489**	**0.582**	**1.90**

3.2 Discussion of Unsupervised Model Results

Tuning of the unsupervised approach involved evaluating several pre-trained embeddings models, tuning the Z parameter for threshold derivation, and developing a windowed similarity method. Each model was compared against a baseline model which attempts to recreate the model described in (Solbiati et al., 2021). In **Table 4** we present the most performant results in each of the pre-trained models we used.

Compared to the baseline model, the tuning of Z values and the introduction of the windowed similarity method result in improvements in the WD and MAE metrics. The USE4 and USE5 models perform better than the SBert Large embedding model in terms of Pk and WD and are faster to train thanks to a smaller architecture. However, the SBERT-Mini (SBN) was the strongest performing model due to its low mean absolute error of predicted topic counts, as well as low Pk and WD.

3.3 Episode Error Analysis

To understand the practical usefulness of our models, we selected two episodes to investigate in detail: one that the unsupervised model classified well, and one that it classified poorly. The "Good Evaluation Example" involves a podcast from `r/AmITheA**hole` (see **Figure 5**, more zoomed in versions in **Appendix A**). Our predicted model resulted in a Pk of 0.248 and a Wd of 0.320, indicating strong matches in predicted topic changes. This is likely because the podcast discusses Reddit content heavily and the embedding model (Sentence BERT) was partly trained on Reddit comments (Henderson et al., 2019) and thus was exposed to similar content in training.

The Bad Evaluation Example involves a podcast from `Fantasy Football 2017` that utilized author annotated timestamps for topic changes. Our predicted model resulted in a Pk of 0.620 and a Wd of 0.717, indicating a poor match between the predicted topic transitions and the reference transitions. A difference between this podcast and the previous one is that it is heavily centered around the topic of football (less variance in embeddings), has three distinct speakers, and the speech pattern is less of a narrative and more of a discussion. The lack of variance in discussion content makes the cosine similarity method less meaningful, and the larger number of topics presented more opportunities for error.

Table 4 Comparison of unsupervised models on the test dataset

Model Configuration			**Test Metrics**		
Model Name	**Description**	**Z**	**PK**	**WD**	**MAE**
Baseline	SBERT Large, cosine similarity	1.0	0.359	0.9278	5.0353
SBL	SBERT Large, windowed similarity	1.55	0.4579	0.5405	0.7175
USE5	Universal Sentence Encoder 5 (large), windowed similarity	1.54	0.4561	0.5404	0.7459
USE4	Universal Sentence Encoder 4, windowed similarity	1.28	0.4551	0.536	0.718
SBN	**SBERT Mini, windowed similarity**	**1.67**	**0.4586**	**0.4855**	**0.576**

4 Future Work

The sections below describe plans improvement plants to our modeling as well as additional clean-up on the dataset we are using in order to achieve better results.

4.1 Dataset

Moving forward, it would be beneficial to collect a larger dataset of podcast episodes to gain diversity and also to rely less on synthetic episodes in supervised training. Additionally, incorporating speaker diarization from the audio files of each episode might prove an effective way of splitting sentences by knowing when one person stops talking and another begins.

4.2 Supervised Modeling

To improve on the performance of the transformer-encoder model, we would like to build an encoder-decoder transformer mode. Also, the model would be optimized further if it were trained with a loss

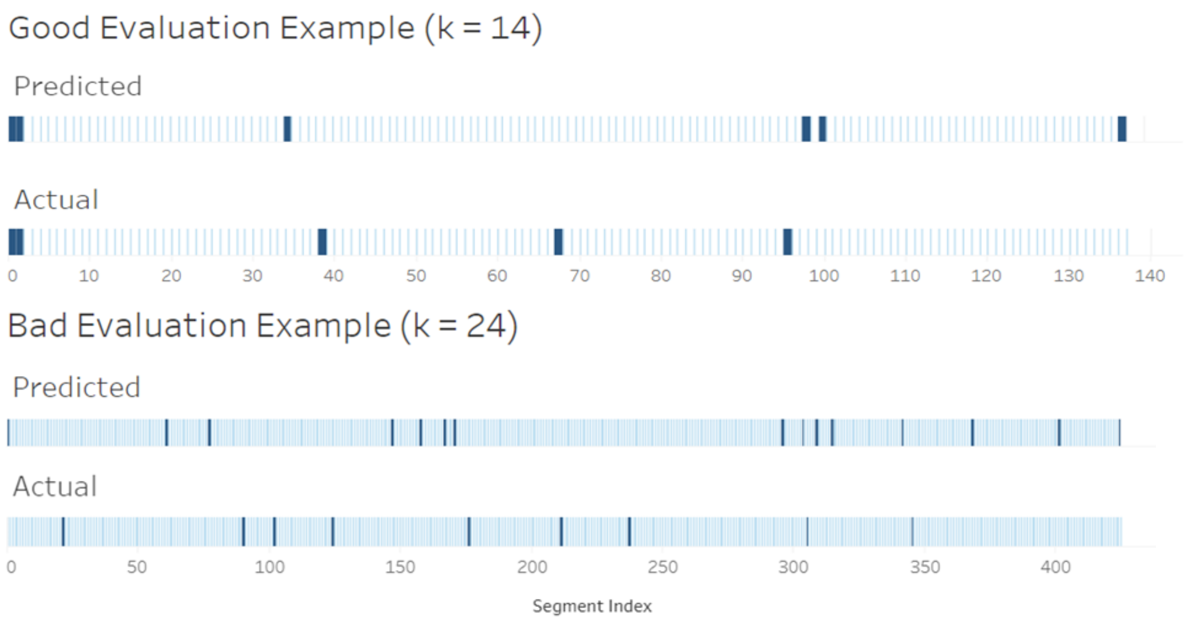

Figure 5 Sample Model Evaluation Visualization. Comparison of predicted segmentation by our model and reference segmentation by YouTube content creators as the mark topic changes within their YouTube videos. Where the dark blue lines match within similar segment indexes (within the window size, k) these can be counted as similar topic transitions between the reference and predicted models. Both predicted models utilized SBERT and the all-MiniLM-L6-v2 pretrained model with a Z value of 1.785 on the YouTube N3 dataset.

function that more closely aligns with the Pk and WD performance metrics, rather than a cross entropy loss function which quickly learns to predict mostly zeros.

4.3 Unsupervised Modeling

The unsupervised learning methods were primarily limited by the signal provided by cosine similarities. In future work, we would attempt to build new methods for comparing each sentence to its neighbors to better identify transitions that don't rely on cosine similarity alone.

References

Carletta, J., & et. al. (2006). The AMI Meeting Corpus: A Pre-announcement. *Machine Learning for Multimodal Interaction*, 28-39. https://link.springer.com/chapter/10.1007/11677482_3

Doug, B., Berger, A., & Lafferty, J. (1999). Statistical Models for Text Segmentation. *Machine learning*, *34*(1-3), 177-210. https://link.springer.com/content/pdf/10.1023/A:1007506220214.pdf

Henderson, M., Budzianowski, P., Casanueva, I., Coope, S., Gerz, D., Kumar, G., Mrkšić, N., Spithourakis, G., Su, P.-H., Vulić, I., & Wen, T.-H. (2019). A Repository of Conversational Datasets. *arXiv preprint*, (arXiv:1904.06472). https://arxiv.org/abs/1904.06472

Koshorek, O., Cohen, A., Mor, N., Rotman, M., & Berant, J. (2018). Text Segmentation as a Supervised Learning Task. *arXiv preprint*, *arXiv:1803.09337*.

Pevzner, L., & Hearst, M. (2002). A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, *28*(1), 19-36. https://aclanthology.org/J02-1002.pdf

Riedl, M., & Biemann, C. (2012). TopicTiling: A Text Segmentation Algorithm based on LDA. *In Proceedings of ACL 2012 Student Research Workshop*, 37–42. https://aclanthology.org/W12-3307.pdf

Solbiati, A., Heffernan, K., Damaskinos, G., Poddar, S., Modi, S., & Cali, J. (2021). Unsupervised Topic Segmentation of Meetings with BERT Embeddings. *ArXiv.org*, *arXiv preprint*(arXiv:2106.12978). https://arxiv.org/pdf/2106.12978.pdf

YouTube Data API. (n.d.). Google Developers. Retrieved December 3, 2022, from https://developers.google.com/youtube/v3

Zhang, L., & Zhou, Q. (2019). Topic Segmentation for Dialogue Stream. *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 1036-1043. https://ieeexplore.ieee.org/document/9023126

A Appendices

A.1 Tables

Table 1 Sentence Embedding Methods

Name	Architecture	Description
SBERT-Large	Transformer	All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs.
SBERT-Mini	Transformer	General purpose model pre-trained with the SNLI and the MultiNLI dataset.
Universal Sentence Encoder (USE)	Transformer	Trained on a variety of data sources and a variety of tasks. Two versions DAN and transformer based.

A.2 Figures

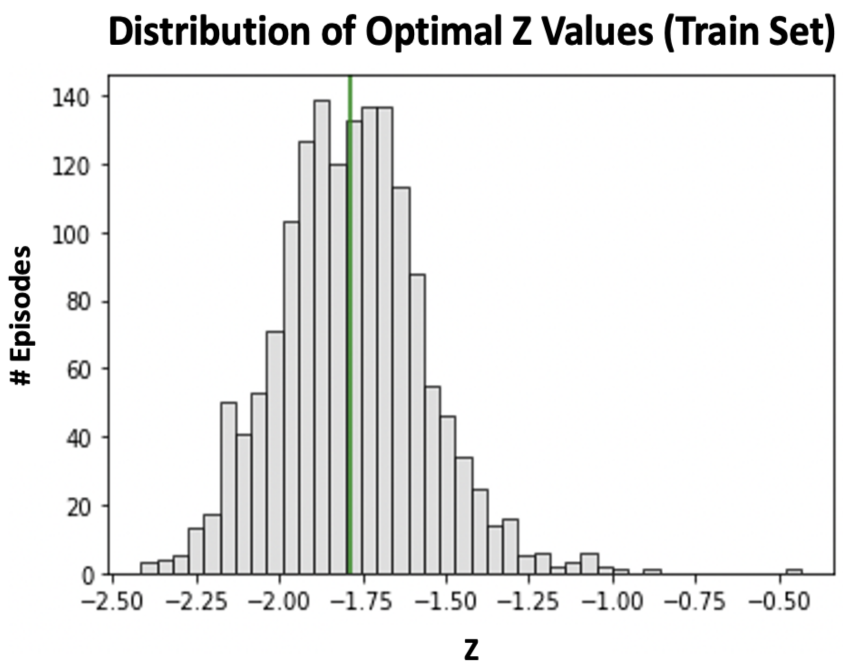

Figure 1 Optimal Z values used in the tuning of our models for training data

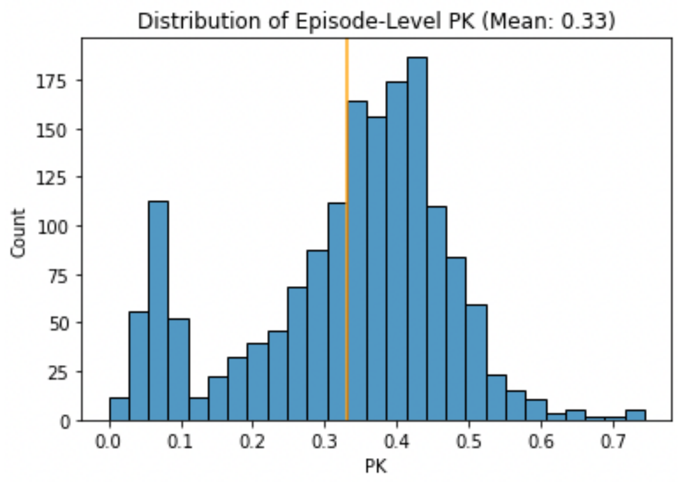

Figure 2 Error Probability (Pk) distribution for top performing models

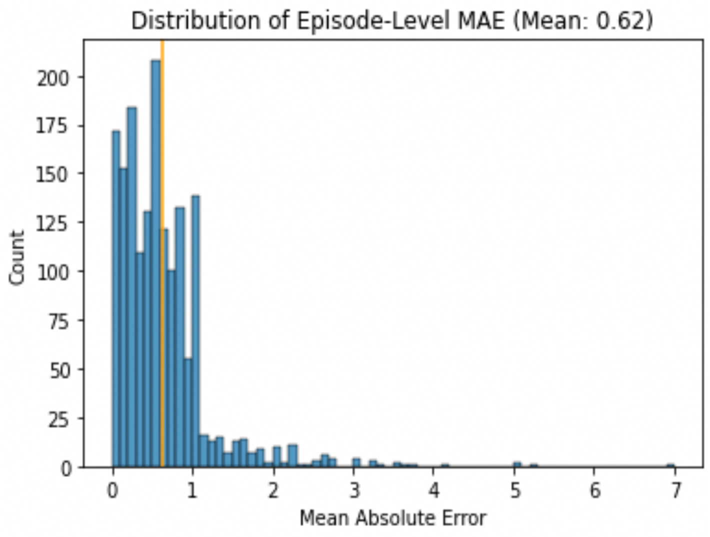

Figure 3 WinDiff (WD) distribution for top performing models

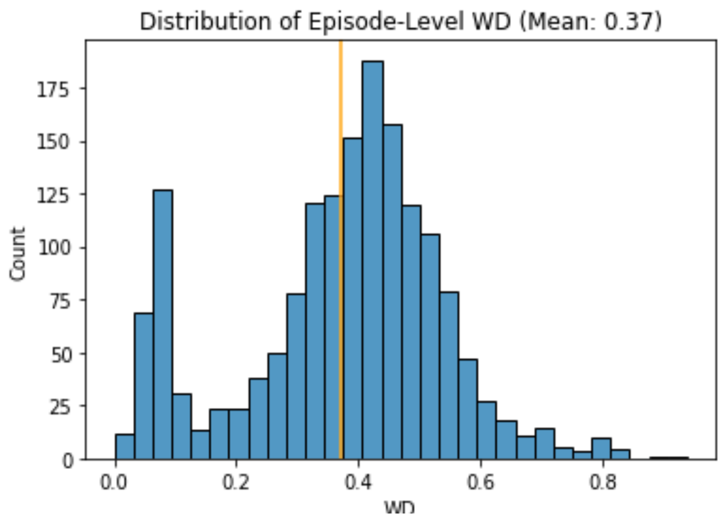

Figure 4 Mean Absolute Error (MAE) distribution for top performing models