Neural Text Segmentation on Podcast Transcripts

E6040.2019Spring.GGOP.report
Brian Midei bmm2172, Marko Mandic mm5305
Columbia University

Abstract

Podcasts transcripts are long form documents of conversational nature which have not yet been studied as topical text segmentation problems. This paper describes a neural network approach to segment podcasts based on topic of discussion. We model the problem as a binary classification task where each sentence is either labeled as the first sentence of a new segment or a continuation of the current segment. We embed sentences using the Universal Sentence Encoder and use an LSTM-based classification network to obtain the cutoff probabilities. Our results indicate that neural network models are indeed suitable for topical segmentation on long, conversational texts.

1. Introduction

Podcasts are an increasingly popular medium for creators and authors to express their ideas. One of the common criticisms of this medium is that listeners may only be interested in a subset of the topics discussed in one episode, but there is no easy way to locate topics within an episode. The vast majority of podcasts offer no segment information and instead release the episodes in their entirety. Providing an efficient method of segmenting these podcasts could provide benefit to both listeners and creators as it would lead to smaller, more digestible content that could drive listenership. Considerable work has been done on topical segmentation of conversational texts, but most of this work is focused on email and text message exchanges. Podcasts present a more challenging task due to the nuances of conversation such as interruptions and incomplete sentences.

We treat the problem as a binary classification task where each sentence is labeled 1 if it is the first sentence of segment and 0 otherwise. We embed sentences using the Universal Sentence Encoder and run them through two stacked bidirectional LSTMs, followed by 3 fully connected layers to obtain cutoff probabilities for each sentence.

In this report, we will first present prior work in text segmentation. Then we will overview the datasets used and the major design decisions. We then present the implementation details and results, followed by a discussion of future work and lessons learned.

2. Related Work

Earlier approaches to topical text segmentation involve assigning a predefined topic to each sentence or determining whether sentences belong to the same topic [2, 3]. These methods are more rigid and perform well for specific text types and tasks, but do not generalize well. Neural text segmentation approaches pose the problem as a binary sentence classification task for modeling section cutoffs. Most works either label the last sentence of a new segment as a positive [4] or label the first sentence of a new segment as a positive [5].

Badjatiya et al. apply an attention mechanism to a sliding window of sentences, with predictions being made on the median sentence in the window. SliceCast also uses an attention mechanism, but we consider the entire document for attention.

All previous approaches make use of word embeddings to encode meaning from the text. A key distinction is that SliceCast employs the Universal Sentence Encoder [6] to obtain the sentence embeddings directly, instead of combining word embeddings of each sentence, as other models do.

3. Data

Our data is separated into 2 main datasets: a large scale training set of Wikipedia articles, Wiki-727k, and a novel dataset of hand-labeled podcast transcriptions.

3.1. Wiki-727k

The Wiki-727k dataset is the primary source of training data. It was introduced by Koshorek et al. [4]. This dataset consists of text from 727,000 Wikipedia articles. Due to rigid structure of Wikipedia articles, each article can be easily segmented using the table of contents. This dataset was chosen due to the consistent labeling and volume of data.

Because the Wiki-727k dataset is so large, we have the luxury of pruning the dataset such that the Wiki training data more closely resembles our target podcast dataset. More details on this process can be found in section 4.1 of this report. After pruning the dataset, the resulting set contains ~150,000 Wikipedia documents.

3.2. Podcast

The Podcast dataset is used to fine tune the model after pretraining on the Wiki-727k dataset. It consists of podcast transcripts from three different shows, obtained either directly from the authors or using Google Speech to Text. The podcast dataset consists of ~100 podcasts of different granularity and structure. One show has an average of four segments per podcast, while the other two are longer and have an average of 15 segments per episode. All podcasts have an average segment length of ~40 sentences.

For segmentation labeling, we use the data provided by the authors or by listeners who have provided their own segmentations. It is important to note that the podcast labels are significantly more subjective than those of the Wiki-727k dataset. While the Wikipedia articles have a static structure, podcast segments are imprecise, with conversations flowing from one topic to the next with no clear boundary. This subjectivity certainly affects the network's ability to predict segmentation and will be discussed in more detail in the discussion of results.

4. Methodology

Here we discuss the most important design choices we made in terms of data representation and modeling.

4.1. Data Representation

In Natural Language Processing (NLP) tasks, one of the first design decisions that must be made is determining how the data will be represented. Neural networks can only interpret numerical data, and thus the information from text must be encoded into numerical form.

Our task is to predict where segment splits occur within a Wikipedia or podcast document. Many NLP tasks partition text into a set of words, called tokens. However, because the scope of the problem is very large documents, we partition each document into a set of strings where each string is a sentence.

Using sentence-level representations, the modeling tasks becomes a task of predicting which sentence is a segment split. Sentence-level representations provide 2 benefits. The first benefit is that each document has fewer timesteps. Modeling long-term dependencies in RNNs can be a difficult task, so shortening the number of timesteps mitigates this concern. The second benefit is that training time is greatly reduced, as the numerical representation for the entire document is much smaller.

4.2. Loss: Classification vs Regression

There are two methods available to model the loss of the network: classification and regression.

In classification, each sentence has a corresponding label of either 0 (non-segment split) or 1 (segment split). In this case, the loss function is a binary cross-entropy loss between the output classes of the network:

$$BCELoss = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log\left(Pr[f(x_i) = y_i)\right]\right]$$

Equation 1: Binary Cross-entropy Loss

In regression, each sentence has a real number label between 0 and 1 indicating its proximity to a segment split. For instance, a label of 1 indicates that a sentence is located precisely on a segment split. A label less than 1 indicates that a sentence is some distance away from a segment split. The benefit of regression is that there is an improved notion of "closeness" encoded in the label representation. A common loss function for a regression model is the mean squared error loss:

$$MSELoss = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - f(x_i)\right]^2$$

Equation 2: Mean Squared Error Loss

We ultimately chose classification as it is the simplest and most common method for segmentation tasks. It is important to note that we also introduce a third 'padding' class, which takes the value of an empty string, with a label of 2.

4.3. Class Imbalance

An important observation from preliminary data analysis is that the classes for classification are quite imbalanced. There are many more "cutoff" sentences than "non-cutoff" sentences. As such, it is necessary to account for this difference in the loss function to ensure that one class does not dominate over another.

To this end, we introduce a categorical cross entropy loss with weighting. This allows for proper emphasis on underrepresented classes. The following modified loss function is used for final training of the network:

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} w_{y_i}\left[y_i log\left(Pr[f(x_i) = y_i)\right]\right]$$

Equation 3: Weighted Categorical Cross-entropy Loss

In this loss function, the weight vector *w* has a value for each class. Each sample is weighted according to the class label of that sample. The class weights varied according to the statistics of the training data. Below are the class weights used for the different datasets:

	Wiki	Podcast
Pad	0.2	0.2
Cutoff	8.0	50.0
Non-Cutoff	1.0	1.0

Table 1: Categorical Cross-entropy Loss Class Weights

4.4. Metrics for success

A final consideration for system design is how to measure performance of the model. Due to the aforementioned class imbalance, accuracy is an insufficient metric. A network that predicts all '0' values would achieve a high accuracy but yield no value to the end user.

An alternative approach to accuracy is Pk-score. This metric, proposed by Beeferman et al. [7], is a probabilistic error metric in which a fixed-length sliding window is moved across the entire document. The metric represents the probability that 2 sentences *k* sentences apart from one another will be incorrectly classified. Due to this formulation, a lower Pk-score is better. An image from the original proposal illustrates this concept in more detail:

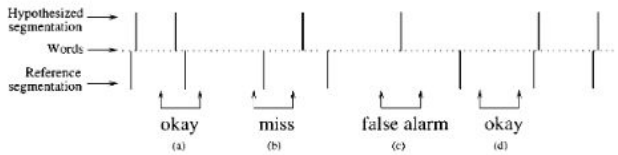

Figure 1: Pk Metric

By using the Pk metric, we preserve the notion of "closeness" in our analysis of the model's performance. Models that reliably predict close to the correct segmentation will score lower (better) than those that predict further away from the correct segmentation.

5. Implementation

This section will overview the implementation of preprocessing and model construction.

5.1. Preprocessing

In an effort to make the Wiki dataset more similar to the podcast dataset, we performed a few preprocessing steps to prune the Wiki dataset of articles with undesirable characteristics.

Each Wikipedia article begins with a summary paragraph which covers a range of topics. We remove the first segment from each article in the dataset to preserve the purity of segments. Next, from exploratory data analysis of the podcast and Wiki datasets, we learned that there is a large discrepancy in average segment length between the two datasets. To combat this characteristic, we eliminated segments shorter than 5 sentences from the Wiki dataset during preprocessing. Additionally, we eliminated all articles with fewer than 3 segments. With these steps, we preserve only the Wikipedia articles that are most structurally similar to the podcasts, which are the target data.

5.2. Deep Learning Network

The architecture of our model is shown in Figure 2.

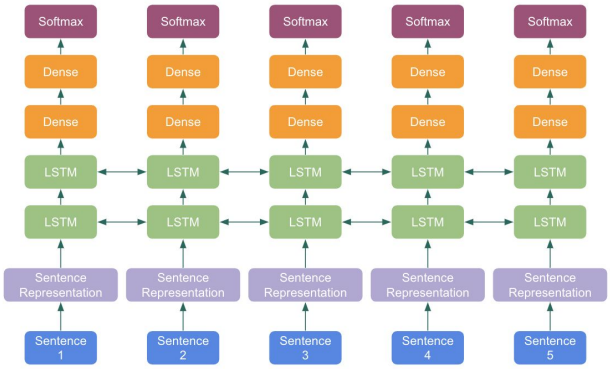

Figure 2: Model Architecture

Typical NLP tasks tokenize documents into word tokens. However, our task is to predict cutoff probabilities for sentences. Therefore, we decide to avoid word embeddings and instead embed sentences directly using the Universal Sentence Encoder [6]. This encoder, provided by TensorFlow Hub, converts a string representation of a sentence to a vector of size 512. This significantly simplifies the learning process by abstracting the encoding process. This embedding layer is the first layer of the network implemented as a keras Lambda layer.

The sentence embeddings are then processed by two stacked bidirectional LSTMs with 256 units each. We then apply 3 fully connected layers of shapes 256, 64, and 3, sequentially. The resulting 3 class output represents the output classes "non-cutoff", "cutoff", and "pad".

We implemented this network using TensorFlow 1.13 and Keras. The training was performed on Google Cloud Compute with a Tesla P100 GPU.

5.3. Transfer Learning

A key challenge is that there is no publicly available dataset of segmented podcasts. We tackle this issue through transfer learning. We train on the Wiki-727k

dataset and validate on on our small podcast set. When the validation accuracy flatlines, we conclude that all the shared features between podcasts and Wikipedia articles have been learned. We then train directly on the podcast dataset using pretrained weights from Wiki training.

5.4. Self Attention Mechanism

Self attention mechanisms have proven their effectiveness in sequence to sequence models at low computational cost. We employ a self attention mechanism to the output of the Universal Sentence Encoder using a single, time-distributed dense layer. The purpose of this layer is to selectively focus on specific sentences and learn an importance distribution across the sentences. As shown in section 6.1 of this report, addition of this attention mechanism provides a modest improvement in Pk score.

6. Results

In this section, we will first discuss the performance of our model with both the Wiki-727k dataset and podcast datasets. We will then compare these results with those obtained by prior work and discuss the insights gained from this work.

6.1. Project Results

Our best model achieved an average Pk score of 25.3 on the Wiki-727k test set. Figure 3 shows the predictions and the ground truth for a single Wikipedia article.

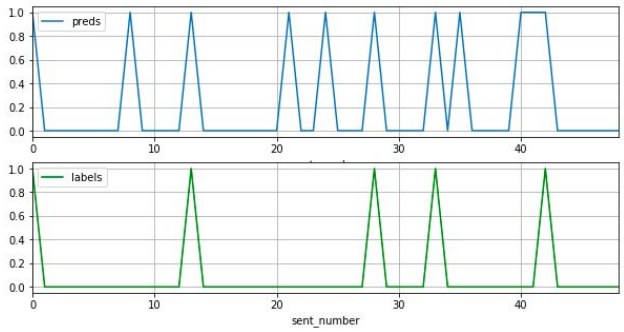

Figure 3: Wiki Predictions (top) and ground truth (bottom)

We are indeed over-predicting section breaks, however these incorrect labelings largely occur at paragraph splits within the document. This indicates that the model has a higher sensitivity to topic changes than is desired.

One important note is that Wikipedia articles are crowdsourced documents with many contributors for each document. This means that paragraphs, even within the same segment, can have different style and voice that could lead the network to predict a segment change.

Our main goal is to achieve good results on the podcast dataset, so we prioritize accuracy on this task.

The best podcast model achieved a Pk score of 38.7 on the podcast dataset. Figures 4 and 5 show the predictions and ground truth for a single podcast, before and after transfer learning is applied respectively.

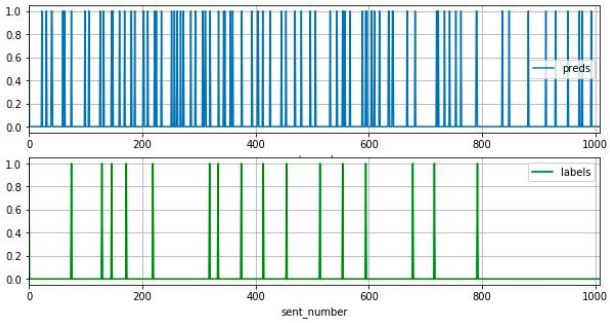

Figure 4: Podcast predictions (top) and ground truth (bottom)

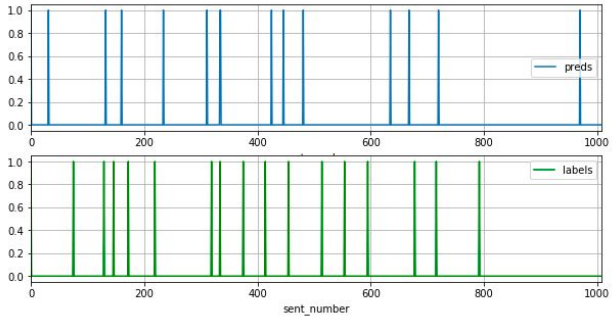

Figure 5: Podcast predictions (top) and ground truth (bottom) after transfer learning

We can see that before transfer learning, the model is unable to generalize the task to the podcast data. After retraining on a small podcast dataset, we significantly reduce the noise and improve accuracy.

Important to note is that the task of labeling podcast segments is inherently subjective and more error prone. The degree to which a topic change results in a different segment depends on the each podcast and whoever labeled the section splits.

6.2. Comparison of Results

Table 2 shows the comparisons of the results achieved by this project and other state of the art models intended for text segmentation.

Pk scores (percent)	Wiki-819k	Podcasts
Mor et. al. (2018)	24.27	NA
SliceCast - Base	27.2	39.0
SliceCast - Attention	**25.3**	**38.7**

Table 2: Pk score Results

The best results we achieve on the Wiki dataset are comparable to those of Mor et. al. Unfortunately, the podcast dataset is novel so we have no benchmark for performance. It is clear that with both datasets, the addition of an attention mechanism results in a slight performance improvement.

6.3. Discussion of Insights Gained

Transfer learning has proved incredibly valuable in this task. The predictions on podcast segmentation are significantly better after using this method, which demonstrates the utility of neural networks even when the number of examples is small. Furthermore, applying the attention mechanism resulted in a slight accuracy improvement with minimal computational complexity increase.

The Universal Sentence Encoder is the only embedding on the TensorFlow Hub capable of embedding the sentences for this task. Character-level embeddings through Elmo proved non-viable due to memory constraints on the GPU. Through this work, we show a utility for the Universal Sentence Encoder to create useful sentence-level representations. One issue with this approach, however, is that the Universal Sentence Encoder embeddings are not trainable and therefore can't be fine tuned for this specific task.

One the learning process. In order to use Keras and TensorFlow operations concurrently, we had to make sure that the TensorFlow graph is only loaded once and then used throughout training, instead of re initializing it on each batch. A small change in the way Keras is imported makes a huge difference in the training time.

7. Conclusion

This project has demonstrated a valuable use case of transfer learning for a text segmentation task. After training the model on the Wiki-727k dataset, we retrained it on a small sample of podcast transcripts and obtained significant results. The Pk score obtained on the Wiki-727k dataset is comparable to the state of the art models, and the Pk score for the podcasts is comparable, meaning that the model generalized well for this task. A larger podcast dataset, and perhaps specializing training to a specific podcast author, would provide more promising results. We hope to extend this project further and provide summarization of each section in addition to segmentation.

8. Acknowledgement

We thank Hassan Akbari for helping our design choices with regards to word and sentence embeddings.

9. References

[1] https://github.com/bmmidei/SliceCast/tree/master

[2] H. Chen, S. Branavan, R. Barzilay, & D Karger "Global models of document structure using latent permutations", Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2009

[3] F. Choi "Advances in domain independent linear text segmentation" In Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, Association for Computational Linguistics, 2000.

[4] N. Mor, O. Koshorek, A. Cohen & M. Rotman "Learning Text Segmentation Using Deep LSTM", The Blavatnik School of Computer Science, Tel-Aviv University, 2018.

[5] P. Badjatiya, L. Kurisinkel, M. Gupta, & V. Varma "Attention-based Neural Text Segmentation", Hyderabad, India, August 2018.

[6] D. Cera, Y. Yanga, S. Konga, N. Huaa, N. Limtiacob, R. St. Johna, N. Constanta, M. Guajardo-Cespedes, S. Yuanc, C. Tara, Y. Sunga, B. Stropea, & R. Kurzweil "Universal Sentence Encoder", Google Research, April 2018.

[7] D. Beeferman, A. Berger, & J. Lafferty "Statistical Models for Text Segmentation", Machine Learning. 34. 177-210. 10.1023/A:1007506220214., 1999