A Neural Model for Text Segmentation

Stanford CS224N Custom Project

Ana Sofia Nicholls
Department of Computer Science
Stanford University
anasnich@stanford.edu

Gene Tanaka
Department of Computer Science
Stanford University
gtanaka@stanford.edu

Abstract

Text segmentation is the task of dividing a document of text into coherent and semantically meaningful segments which are contiguous. This task is important for other Natural Language Processing (NLP) applications like summarization, context understanding, and question-answering. The goal of this project is to successfully implement a text segmentation algorithm. We take a supervised learning approach to text segmentation and propose a neural model for this task. We aim to extend this task to podcasts by using existing transcription services. Our model obtained a Pk score (described below) of 6.54 on the Wiki-50 dataset, which was an improvement over our baseline score of 69.23. In this paper we will discuss experiments with self-attention as a modification to our model.

1 Key Information to include

- Mentor: Professor Manning
- External Collaborators (if you have any): N/A
- Sharing project: CS210 potentially

2 Introduction

Text segmentation is defined in this paper as the task of dividing text into coherent, contiguous segments that each represent a change in topic. This task is an important stepping stone to more complex tasks such as text summarizing and information extraction. One primary issue in this field of research has been the scarcity of labeled data. In addition to being scarce, existing data tends to be synthesized automatically, which makes them unreliable metrics for evaluating models. As a result, many efforts in text segmentation have attempted heuristic-based and unsupervised learning methods. Past approaches to text segmentation have used Bayesian probabilistic models in which topics impose distributions over the vocabulary, as well as GraphSeg, an unsupervised graph method where nodes are sentences connected by edges when semantically similar.

The goal of this project is to successfully implement a supervised text segmentation based on neural LSTM architecture. Our motivation for this project is two fold: text segmentation is an important task within NLP, but is also an extremely useful task for a variety of purposes. In NLP, text segmentation is important for other tasks like summarization, context understanding, and question-answering. Furthermore, segmentation in general is an extremely useful task. Our main inspiration for this project was realizing that audio content is exploding across platforms (such as podcasts, YouTube, Audible, etc) and has become a very popular medium for both learning and entertainment. While tools for transcription have been developed, audio content remains difficult to efficiently search, navigate through, and index. For example, podcast episodes can be hours long, and being able to segment an episode into relevant chunks could greatly increase a user's efficiency in learning from podcasts.

3 Related Work

Text segmentation has been explored both in a supervised learning setting and an unsupervised setting. For the supervised setting, multiple neural models have been proposed. We took inspiration from Koshorek's group [1] and treated the task as a supervised learning. We attempted to recreate their neural model and adopted their labelled Wikipedia dataset (more details about this dataset in the Data section) to train our model.

Other groups that have approached this task from a supervised learning standpoint have proposed different neural models along with different datasets. Badjatiya's group [2] used an attention-based neural model which included a convolutional neural network to create the sentence embeddings instead of an LSTM neural model which Koshorek's group used. Similarly to Koshorek's group, Badjatiya et al used a bidirectional LSTM followed by a fully connected layer for the output. The approaches have many similarities except for a few important differences. Badjatiya's group emphasized their use of attention in their neural models, whereas Koshorek's group did not consider attention in their model. We drew inspiration from Badjatiya's use of attention to include incorporating self-attention as one of our experiments to be able to combine the two approaches and provide an extension of Koshorek's model. Badjatiya's dataset combined data from different domains, such as clinical data, fiction novels, and Wikipedia, whereas Koshorek used only Wikipedia data.

Unsupervised approaches have also been explored for this task. Purver's group [3] used an unsupervised approach which involved using MCMC and Gibbs Sampling. Their experiments focused on spoken language, so they used a meeting corpus as their dataset. However, their model did not perform as well when compared to a supervised model. All of these papers used either window diff or P_k as their evaluation method, and we followed suit so that we could compare our results to the algorithms described.

4 Approach

For this text segmentation task, we are using a neural model with two LSTM networks. LSTMs keep longer term memories to be able to encode long range dependencies. They do so using gates which control what information can flow to a memory cell state. A bidirectional LSTM helps capture information from both directions into embeddings.

Our first subnetwork, which we will refer to as the Sentence Representation Network, is an LSTM that generates sentence representations. The Sentence Representation Network is a bidirectional LSTM with two layers. The input to this network is a tensor of sentences representing a single document. Each sentence contains a sub tensor of stacked word embeddings pertaining to each word in the sentence. After running the input tensor through the network, we take the mean of all the word embeddings pertaining to a sentence so that the output of the LSTM is a single vector embedding for each sentence.

Our second neural network is a bidirectional LSTM with two layers. We input a sentence embedding into this LSTM, and obtain a distribution as output. Lastly, we run the output of the second LSTM through a fully connected layer to obtain a sequence of n vectors in $\mathbb{R}^2$, where n is the number of sentences. During validation/testing, we then apply a softmax function to obtain segmentation probabilities for each of those vectors. For training, we are using cross entropy loss with an Adam optimizer.

For one of our experiments, we also implemented a self-attention mechanism into the Sentence Representation Network. Attention allows the network to better determine what parts of the input are important. Self-attention helps relate different positions within a single sequence to produce a representation of the same sequence.

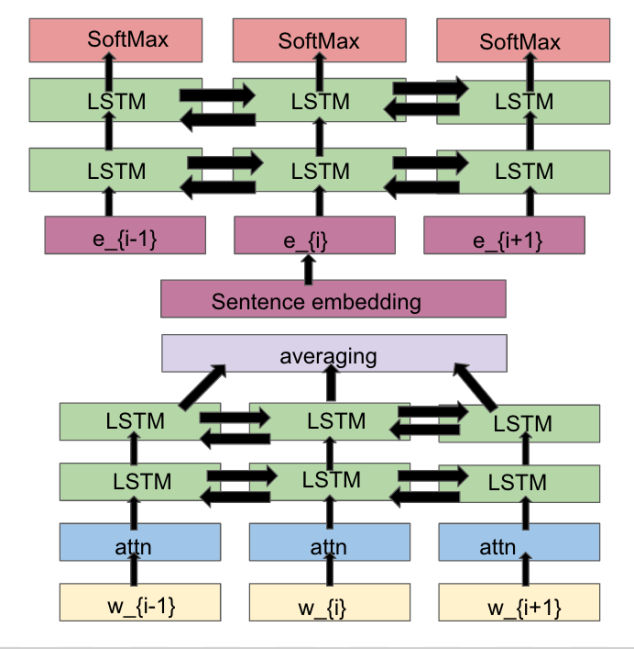

Figure 1. Diagram of our neural network model with attention.

Originally, we planned to use the same baseline as Koshorek's group, which involved calculating the average segment length (k) in the dataset and labelling every k^{th} sentence as the end of a segment. However, after further inspection, we decided to change our baseline to the following. We take the average of all of the word embeddings in each sentence. Looking at two consecutive sentences, we set a threshold for the difference between them. If the difference between the two sentence embeddings is greater than the threshold, then we label the first sentence as ending a segment. The reasoning behind this baseline is that we would expect the average of the word embeddings in a sentence to encode some semantic information about a sentence. Furthermore, we would expect sentences belonging to the same segment would have similarities in semantic meaning.

We adapted the code for loading FastText word embeddings from the FastText website [4]. We also based our code for our evaluation metrics on the Pk and WindowDiff implementations in the NLTK library, though with slight modifications to be compatible with NumPy arrays instead of Python lists.

5 Experiments

5.1 Data

For this task, we will be using the "Wiki-727K" dataset. It was created by the authors of the paper "Text Segmentation as a Supervised Learning Task" [1]. As described in the paper, the dataset contains 727,746 Wikipedia documents with segmentations as described by their table of contents. The documents have had all non-text elements and single sentence segments removed. We wrote a script that parses each document through the following steps:

- Create tensor of target labels by reading through document sentence by sentence and marking sentence as a border (1) if it is followed by the string "===", which denotes the start of a new section. Otherwise mark the sentence with a 0.
- Pad tensor of target labels with 0s to match length of longest document in dataset.
- Create tensor of word embeddings by looping through words in each sentence and looking up FastText word embedding.
- Pad each sentence with zero-vectors to match length of longest sentence in dataset.

We trained our model on 561 randomly selected Wikipedia documents, and evaluated on the same 50 documents ("Wiki-50") used in the Koshorek [1] paper. There is no overlap between these two sets of documents.

5.2 Evaluation method

Similar to the paper we referenced in our proposal, we are using the P_k metric [5] for our quantitative evaluation. $P_k(ref, hyp)$ is the probability that a pair of chosen sentences with a distance of k are inconsistently classified. The possible classifications at each step are:

(a)/(d) "okay": true (ref) break and hyp break both present/both absent within length k window

(b) "miss": true break present but hypothesized break not present within length k window

(c) "false alarm": hypothesized break present but true break not present within length k window

Each of the classifications are illustrated in the image below [5]:

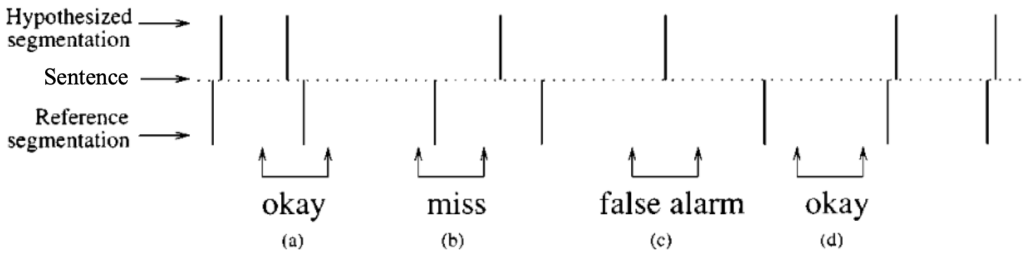

Figure 2. Diagram of possible P_k classifications, adapted from Beeferman, et al [5]

Since its introduction in 1999, a few papers have critiqued the effectiveness of the P_k metric. Due to P_k's perceived shortcomings, we are also using the *WindowDiff* metric [6], which is a modified version of the P_k metric that penalizes near-miss errors less than "pure" false positives. The formula for *WindowDiff* is as follows:

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

where $b(i, j)$ denotes the number of boundaries between positions i and j in the text.

(As shown later in our results, P_k and *WindowDiff* can produce the same value under certain conditions.)

Nonetheless, P_k still seems to be the most common metric for text segmentation tasks today (based on number of citations of Beeferman et al). Our P_k calculations will be useful as they will allow us to eventually compare our model's results to the P_k values of the models described in Koshorek 2018. The table is shown below:

	WIKI-727K	WIKI-50	CHOI	CITIES		ELEMENTS	
P_k variant	*sentences*	*sentences*	*sentences*	*sentences*	*words*	*sentences*	*words*
(Chen et al., 2009)	-	-	-	-	22.1	-	**20.1**
GraphSeg	-	63.56	**5.6-7.2**	39.95	-	49.12	-
Our model	22.13	**18.24**	26.26[3]	**19.68**	**18.14**	**41.63**	33.82
Random baseline	53.09	52.65	49.43	47.14	44.14	50.08	42.80
Human performance	-	14.97	-	-	-	-	-

Figure 3. Table of P_k results from Koshorek, et al [1]

5.3 Experimental details

We trained our model on a training set of 561 documents for 10 epochs. Our model contains a Sentence Representation Subnetwork and a Prediction Subnetwork, each containing bidirectional

LSTMs. To start, we made both LSTMs 2 layers deep. We also experimented with a hidden layer size of 256 for the sentence encoder, and a hidden layer size of 128 for the segmentation prediction layer. We also added self-attention with an embedding size of 300, with residual dropout of 0.1 and attention dropout of 0.1. For training, we used an Adam Optimizer with a learning rate of 1×10^{-4}.

For our baseline, we experimented with a threshold of 5.0, after finding that it produced the lowest P_k score of thresholds between 1.0 and 7.0.

5.4 Results

Our P_k results are compared to our baseline in the table below.

Evaluation on Wiki-50 Dataset		
Model	P_k	*WindowDiff*
GraphSeg	63.56	-
Koshorek	18.24	-
Model w/ Attention	7.62	7.52
Model w/o Attention	6.54	6.54
Baseline	69.23	69.23

Figure 4. Table of P_k results from our experiments (GraphSeg, Koshorek, adapted from [1])

At first glance, these results are very promising, as our model seems to outperform GraphSeg [7] as well as Koshorek. However, there are a few important things to note about these results, as discussed in the following section.

6 Analysis

To start, GraphSeg appears to only do about as well as our baseline, making it a very weak model. One possible reason is that the GraphSeg model was not trained on the Wiki-727 dataset. This seems to be evidence that text segmentation models trained on one dataset do not generalize well to other datasets. Additionally, the GraphSeg model uses an unsupervised graph method, which Koshorek et al proposed is less effective than supervised methods for text segmentation tasks.

Second, P_k values fluctuate greatly depending on the value of k. For example, consider a document of 10 sentences, which contains 1 break (border). For example, if $k = 10$, $P_k = 1.0$ if the prediction predicts 0 breaks. On the other hand, if $k = 1$, $P_k = \frac{1}{10}$, as only the window containing the 1 true break would be classified as incorrect. Since we do not know the k value used by Koshorek et al, it is very difficult for us to conclusively state that our model outperformed theirs. For our evaluation, we calculated k to be the average segment length within the entire set of documents.

Surprisingly, the model performed better without attention. The P_k with attention was 7.62, whereas without attention the corresponding value was 6.54. This could be because the attention altered the sentence embeddings in a way where certain signals that were actually useful for segmentation got lost.

Additionally, we were surprised to learn that our baseline performed worse than the baseline proposed by Koshorek et al. Their proposed baseline starts a new segment after every sentence with probability $\frac{1}{k}$, where k is the average segment size in the dataset. It seems very weak and naive, yet it achieved a P_k of 52.65, while our baseline achieved a P_k of 69.23. One possible reason for this is that we did not have the optimal threshold for difference between two consecutive sentence embeddings.

Consider the following document, where bolded sentences represent segment boundaries:

> 0. **IceRocket is an Internet search engine which specializes in real-time search.**

1. Based in Dallas, Texas, it launched in 2004 hoping to market itself solely through word of mouth.
2. IceRocket is backed by Mark Cuban and headquartered in Dallas, Texas.
3. The company has received angel funding from Mr. Cuban.
4. **Icerocket launched in 2004.**
5. The search engine originally launched with features designed to make web searches on a PDA much easier, for instance allowing users to email a query to the engine and receive their results back in response.
6. Icerocket had an early licensing deal with Gofish.com In August 2011, it was announced that IceRocket had been acquired by the Meltwater Group.
7. **IceRocket is generally for blog searches but has expanded into searching the popular social networking websites Twitter and Facebook as well as allowing searching of news and the world wide web.**
8. IceRocket's Big Buzz feature allows users to search Blogs, Tweets, news, images etc. all from one page.
9. The IceRocket site is a free resource for people looking to monitor their brand, it is ad supported. IceRocket has an API that it licenses to social media monitoring firms as well as PR agencies.

The target for this document is $\begin{bmatrix}1\\0\\0\\0\\1\\0\\0\\1\\0\\0\end{bmatrix}$ our baseline predicts $\begin{bmatrix}0\\1\\1\\0\\0\\0\\0\\0\\1\\1\end{bmatrix}$, and our model predicts $\begin{bmatrix}1\\0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix}$.

If we start by looking at our baseline, we notice that the predictions are somewhat close to the 1st and 3rd boundaries in the target, but there is a major issue where it predicts consecutive sentences to be boundaries. Ultimately, it is likely that our technique of averaging word embeddings to create a sentence representation makes a strong and irrational assumption that all words in the sentence have relatively the same semantic importance. While this method is useful for being computationally fast, we would likely retain more semantic information by calculating a weighted average.

For this document, our model correctly predicted the 1st boundary, was a near miss for the 2nd boundary, and missed the 3rd boundary. Our model was commonly able to correctly predict a boundary on the first sentence of the document. This was expected, as many documents in our training set had a boundary at the first sentence. As shown by our model's inability to identify the boundary at the 8th sentence, our model tends to overly predict 0s, especially further into a document. (Even going so far as predicting all 0s for certain documents.) This is likely due to the fact that a wide majority of sentences in each document are labeled 0. This is exacerbated by the fact that our training data is padded with 0s such that all of our target sizes match the length of the longest document in the training set.

7 Conclusion

Text segmentation, the task of dividing a document into contiguous sections that are semantically and contextually meaningful, is a field of research that will benefit from advancements in sentence representation. In turn, text segmentation has great potential to aid the NLP tasks of information extraction and summarization. While previous work in this domain has primarily investigated unsupervised methods, there appears to be potential for improvement through supervised methods.

For our model, we focused on a supervised LSTM-based model to predict segmentation. We successfully incorporated self-attention into the model, although it did not improve our results as we would have expected. Through trial and error, we learned the importance of a having strong baseline and consistent evaluation metric. We also recognized that models must be trained and evaluated on similar data in order to make valid comparisons between results.

As for limitations, despite our model doing very well, we worry that it leans toward predicting 0 for all examples (specifically, classifying each segment as not ending a segment). Although this leads to good evaluation scores, this output is not helpful to obtain accurate segmenations of a particular document.

For future work, we would like to address the limitations of our baseline by attempting stronger alternatives for creating sentence representations. One interesting alternative by Arora et al [8] takes a weighted average of word vectors, then modifies them using Principal Component Analysis/Singular Value Decomposition. Another potentially interesting method would be to use deep averaging networks to obtain sentence representations. In addition to improving our baseline, we would like to experiment with attention in other layers in the neural network.

References

[1] Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Beranty. Text segmentation as a supervised learning task. In *Association for Computational Linguistics (ACL)*, 2018.

[2] Pinkesh Badjatiya, Litton Kurisinkel, Manish Gupta, and Vasudeva Varma. Attention-based neural text segmentation. In *IIIT-H*, 2018.

[3] Matthew Purver, Condrad P. Kording, Thomas L. Griffiths, and Joshua B. Tenebaum. Unsupervised topic modelling for multi-party spoken discourse. In *Association for Computational Linguistics (ACL)*, 2006.

[4] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[5] Doug Beeferman, Berger Adam, and John Lafferty. Statistical models for text segmentation. In *Machine learning 34(1):177-210*, 1999.

[6] Lev Pevzner and Marti A. Hearst. A critique and improvement of an evaluation metric for text segmentation. In *Association for Computational Linguistics (ACL)*, 2002.

[7] Goran Glavas, Federico Nanni, and Simone Paolo Ponzetto. Unsupervised text segmentation using semantic relatedness graphs. In *Association for Computational Linguistics (ACL)*, 2016.

[8] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.