Topic Segmentation using Supervised Methods

Pinkesh Badjatiya - 201402002
November 25, 2016

1 ABSTRACT

We present a supervised methods to detect topic segments in a chunk of text. This method uses a combination of encoder + classifier to detect topical segments. Our initial experiments show decent accuracy, though the accuracy providing more possibilities for furthur exploration.

2 INTRODUCTION

The problem of Topic Segmentation requires segmenting a chunk of text into meaningful segments, trying to separate them based on Topical Structure. A lot of unsupervised approaches exist which work on the Bayes Formulation of the problem taking into account some of the manually extracted features like Cue Words, Lexical Cohesion, Topic Modeling, etc. My aim was to work on a supervised approach and creating some baseline methods.

2.1 MOTIVATION

- Large dataset available for training, about 51.8 million documents. This translates to more than **200 million samples** for training.
- Not much work done on supervised methods. The closest work in the area of topic segmentation aimed at finding topic segments from an email conversation. Other methods rely heavily on the unsupervised methods.
- The problem of topic segmentation finds its uses in a lot of places. One of them is, instead of a search engines/bots indexing pages based on topic can index topic segments derived using the system. This provides better access to required content.

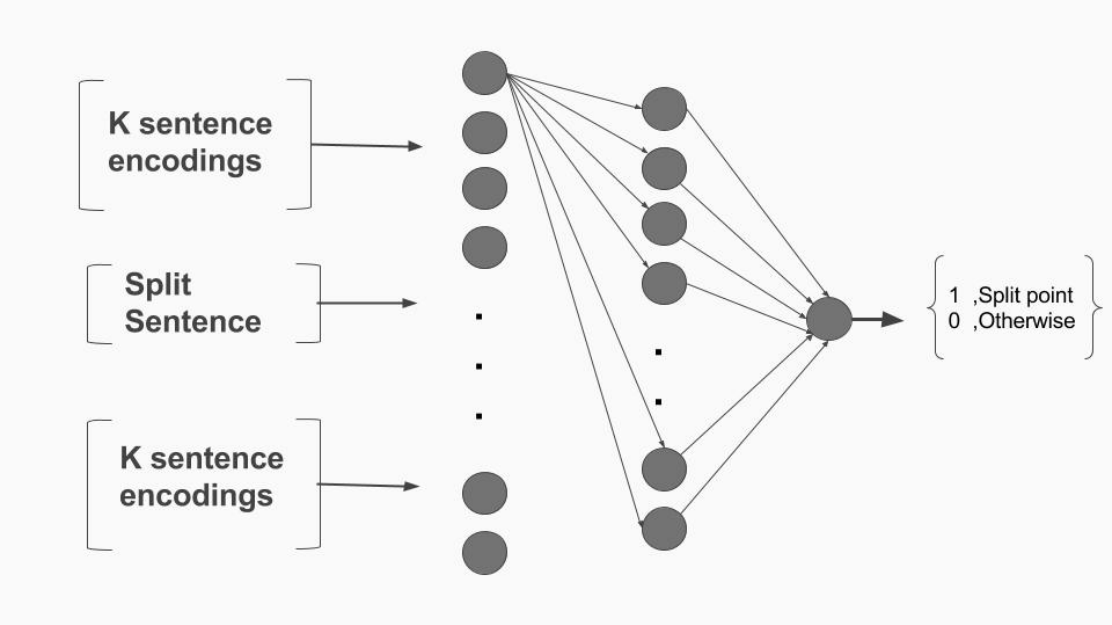

Figure 3.1: Structure of the classifier

- All existing systems use unsupervised model making the whole process cumbersome and it relies on the human feature extraction which is prone to a lot of errors. Proposing a supervised model will take care of most of the work.

3 Approach to the problem

The whole process of creating a supervised requires a pipeline of 3 almost independent systems, eventually we plan on backpropogating results to improve the system. The following are the details of each of them in the same order.

3.1 Data Processing

The main abjective of this module was to process the 13GB wiki dump extracting about 51 million documents. This huge amout of data needed to be filtered furthur to get quality documents with each section/paragraph containing MIN_SENTENCES_IN_SECTION sentences and other filtering. These samples can then be passed on to the encoder.

3.2 Sample Encoder

This module encodes the samples obtained into multiple encoding schemes which are then used for classification. The initial baselines were:

- **TF-IDF Encoding**
  The TF-IDF model training was done on the whole wiki corpus. Each sample was

converted into a matrix of (2xVOCAB_SIZE), which is a concatenation of 2 consecutive paragraph across a split point.

- **Mean Word2Vec Encoding**
  The Word2Vec model was pre-trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. Each sample was then converted into a vector of size (2x300) where each (1x300) dimensional vector is text across a split point.

- **TF-IDF weighted Word2Vec Encoding**
  TF-IDF provides a score for the relevance of a word in using the frequency measure of tokens across documents. This score can be used to create a weighted word2vec instead of a simple mean of the encodings. I am still in the process of finalizing this method but we hope it will provide better accuracy than the above method.

3.3 Classification Model

This consists of a simple Neural net trying to classify if there is a split point in the given input. My initial tests were done on a simple 1-hidden layered net with about 32 neurons.

4 Experiment

For my experiment i extracted a total of about 80k processed samples that satisfy the filtering criteria. Then splitting the data into test and train samples with ratio 0.2/0.8 resulted in about 64k training samples. I tested my pipeline for the Mean Word2Vec encoding with a simple neural net with 1 hidden layer and 32 neurons and no hyper parameter tuning.

The classification resulted in about **78% accuracy** on the test dataset. This accuracy measure is not a good measure to check the system as the test metric used in the papers for evaluation of the model is WindowDiff Metric which i will be working on next.

5 Interface

The web-interface for the API allows user to upload the document and get the text segmented into paragraphs. Currently it uses Affinity Propogation to predict the split points, but once the system is ready, it can be plugged into the API and can be used as a black box for segmentation. The live demo is available **here**.

6 Code

All the code related to this project is hosted in a private repository on Github. To access the repository, the person needs to be added as a collaborator first.
Link to **Repository**

```
X(train)= 63933.6
X(test)= 15983.4
Epoch 1/10
63933/63933 [==============================] - 24s - loss: 0.5238 - acc: 0.7514
Epoch 2/10
63933/63933 [==============================] - 25s - loss: 0.4845 - acc: 0.7745
Epoch 3/10
63933/63933 [==============================] - 24s - loss: 0.4601 - acc: 0.7876
Epoch 4/10
63933/63933 [==============================] - 24s - loss: 0.4366 - acc: 0.8013
Epoch 5/10
63933/63933 [==============================] - 23s - loss: 0.4105 - acc: 0.8150
Epoch 6/10
63933/63933 [==============================] - 23s - loss: 0.3841 - acc: 0.8296
Epoch 7/10
63933/63933 [==============================] - 24s - loss: 0.3550 - acc: 0.8456
Epoch 8/10
63933/63933 [==============================] - 23s - loss: 0.3238 - acc: 0.8594
Epoch 9/10
63933/63933 [==============================] - 24s - loss: 0.2931 - acc: 0.8742
Epoch 10/10
63933/63933 [==============================] - 23s - loss: 0.2632 - acc: 0.8901
15968/15984 [============================>.] - ETA: 0sacc: 78.00%
```

Figure 4.1: Experiment results showing 89% as training accuracy while 78% as test accuracy.

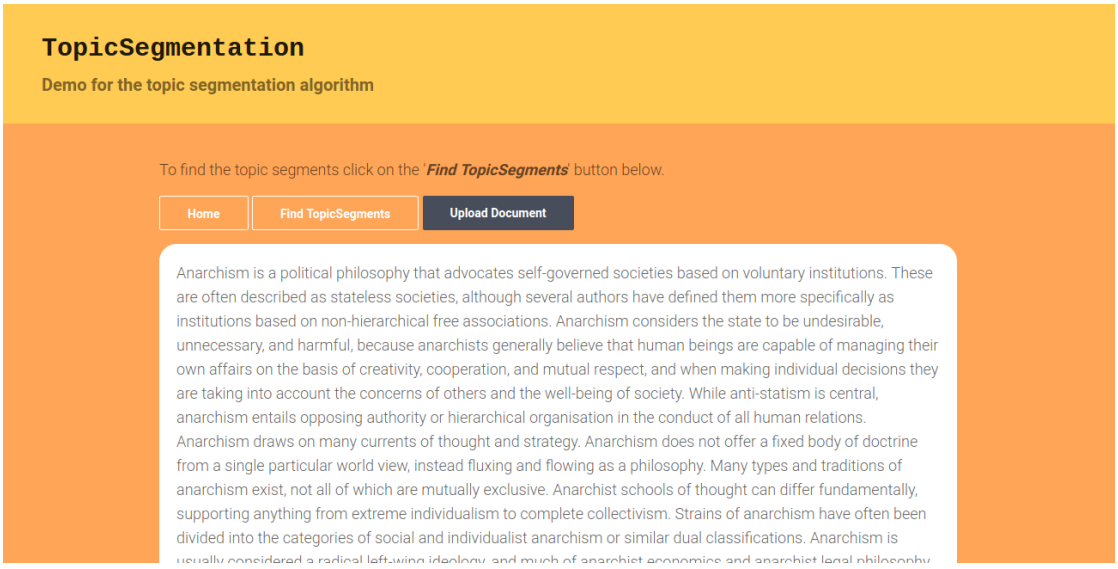

Figure 4.2: Web-Interface for the system

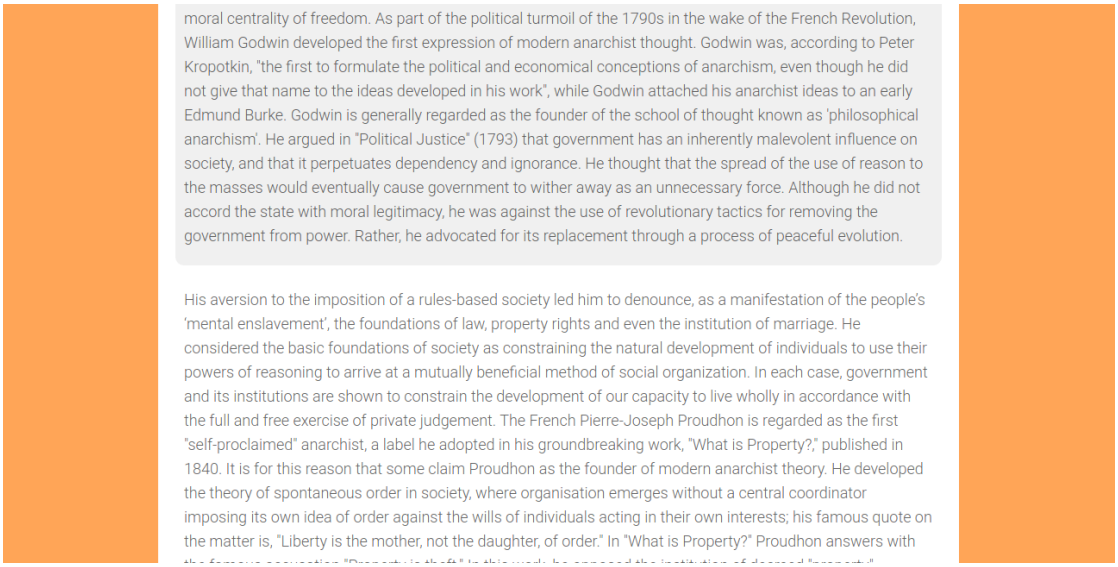

Figure 4.3: After the segments have been detected