**BriYF.com**

User Guide

**Preface**

Team:

Rhys Li

Alex Whelan

Kai Yan

In Collaboration with:

Professor Martin Hilbert Associate Professor Department of Communication

Audience:

{FILL HERE}

**Overview of Product**

In our mission of creating qualitative analysis of the political landscape we had two major tasks to overcome. First we needed to gather data in the form of opinion articles from various news outlets like Fox, CNN, CBS, Politico, etc. Second, we had to figure out how to use this data with a machine learning algorithm to not only make sense of our data but also make it somehow meaningful to the end-user.

Our product will allow a user to query a topic, perhaps something they are interested in, and in addition write a paragraph of their thoughts on the chosen topic. Our application will return to the user a list of similar articles, their summarizations, and a visualization of the vector space depicting the articles mined and annotated by the sources. In addition, we provide a sentiment analysis module, that lets users input text, which returns a binary classification in terms of a score from 0-100. All this encapsulated in an easy to use website where the web scrapping and analysis will be persistent.

In addition to the qualitative aspect we have developed a platform supporting multiple users that will have access to a repository of information and NLP tools. Our product allows users to subscribe to topics and we automate the web scrapping. The tools we provide such as the visualization, sentiment analysis, and opinion comparison aid the users in the qualitative aspect of information extraction. Our mission is to reduce the time consumption of gathering an filtering articles from various news outlets by providing them with automated centralized database.

DISCLAIMER:

possible use of articles from other websites?

**Technology**

1. Django 2.0 with SQLite - <https://www.djangoproject.com/>

2. Gensim - <https://radimrehurek.com/gensim/>

3. TSNE

4. Python 3.5

5. CELERY????

**Installation**

The website is hosted and available online at [https://www.bryif.com](https://www.bryif.com/). This is made in an effort to reduce the hassle of having to install certain APIs and external libraries.

Developer Installation Instructions

1. Install Conda for ease-of-use virtual environments

1. Follow the instructions at <https://conda.io/docs/user-guide/install/index.html> to install virtual environment for package management dependent on your Operating System.
2. After installation create a virtual environment via: conda create --name Environment\_Name python=3.5
3. Activate the Environment via: source activate Environment\_Name

2. Install Gensim and external libraries - <https://radimrehurek.com/gensim/install.html>

3. Install Django – follow instructions on <https://www.djangoproject.com/download/>

1. After installing Django go to the root directory of the website code and run: python manage.py runserver

**Functionality**

Step 1: Sign-up

- Register as a user

{INSERT GRAPHIC}

{ADD: Description}

Step 2: Subscribe to topic(s)

- Add a list of topics you are interested in reading about and the miner will perform the automated web scrapping task for you.

{INSERT GRAPHIC}

Description: We have targeted 4 news websites, which are CNN, Foxnews, CBS, and Politico. For the web scraper, user can input the keywords they want to the web scraper and the scraper will automatically find the articles from each news websites' opinion section. We used the build-in function urllib3 to get the data from news websites and and process the html data by regex. Then we written the data into text file in an arranged order, including keyword, author, title, url, date, source, and content,  and make 2 json file to stored the keyword information and loaded urls. These data can will be stored in database and used for Doc2Vec and summarization.

Step 3: Find similar article(s)

- Write a brief paragraph between 70-200 words describing our opinion or thoughts of a particular topic. Gensim’s Doc2Vec will return three candidate articles that match closest to your opinion.

Description:

Instead of building a recurrent neural network(RNN) from scratch we decided to use Gensim (generate similar) framework for handling data analysis. Gensim in one of the leading NLP tools used in text classification for generating similarity indices between articles. The similarity score is assigned by computing cosine similarity between the input opinion (after it has been converted into a vector embedding) across all the articles within our data corpus. Unlike the bag-of-words approach which is simply records frequency of words in a document, gensim uses a hierarchical system of using word vectors, which are shared across all documents, to generate paragraph vectors.

This particular doc2vec model is called distributed memory. It generates a unique paragraph vector embedding by using a sliding window which iterates across a fixed length of text making a prediction on the next word given the current context. Once a prediction is made and recorded this prediction is appended to the paragraph vector. In the end, after iterating over the entire length of the text N epochs, the sum of all the inferences made, the paragraph vector, can be thought of as the topic of the paragraph. There are many advantages in using this model , one of which is the ordering of words is preserved thus it can capture semantics of the paragraph much better than say bag-of-words.

The algorithm operates in 2 stages. First the training stage in which we use Googles pre-trained word vectors ([https://code.google.com/archive/p/word2vec](https://code.google.com/archive/p/word2vec/)/) to get weights for our RNN and generate an embedding for a specific paragraph. The training task involves adjusting training parameters such as context window size, removing infrequent words, settings number of epochs (iterations) with an end result of receiving weights for our model which are used in the inference stage. Second, the inference stage where we introduce a new paragraph and feed in through the RNN to generate a new embedding which we will compare to our existing embeddings using cosine similarity.

In-Depth Explanation Distributed Representations of Sentences and Documents

<https://arxiv.org/pdf/1405.4053v2.pdf>

-Step 4: Browse the scrapped articles

**Troubleshooting**

**FAQ**

How accurate is the model?

- The nature of classifying opinions is intrinsically biased. In other words, some may agree with the returned opinion and some may not. One option in terms of testing would be to use a holdout dataset of opinions which we don’t have or use the existing corpus and subtract a portion of them to be used as the holdout set. However, manual checking is an arduous task which is extremely time consuming as it involves reading not only the opinion, understanding the theme and authors intent but also having to do that across a large corpus of documents. Given that each article is approximately 600 words testing on a minimum of 100 articles would involve reading over 400 pages of text as we would want to look at least the top 3 results. However, we do test for convergence which gave us an approximation of how many words we must enter before the algorithm classifies article correctly. Taking an increasingly larger subset of text we increment the sample size until the algorithm is able to correctly classify the article. In addition, we discovered that using Google News pre-trained word vectors decreased the amount of words needed for the algorithm to make the correct prediction.

How large is the data corpus?

- Currently we have support for four news sources being FoxNews, CNN, CBS, and Politico. Other sources such as Wall Street Journal have a paywall that require a subscription to use their services. On the topic of tax reform we are able to gather 1000+ articles across these four sources, however this data set still small as most commercial Machine Learning applications require much larger datasets on the order of tens or hundred of thousands.

Are there other models that perform better or have higher accuracy?

- This is a difficult question to answer as it involves doing extensive research. There are various hyper-parameters that can be adjusted such as the size of the context window, using distributed memory versus bag of words, the number of epochs in the training phase, the dimensions of the vectors, the options to concatenate versus taking the average of the paragraph vectors and many others options that may or may not improve the quality of the returned articles.

What is the difference between doc2vec and word2vec?

Simply put the doc2vec model is hierarchical in that it uses word vectors to determine paragraph vectors. By default the word vectors are initialized randomly, but there are options to include pre-trained word vectors which we use in our model. Word vectors are embeddings used on individual words while document vectors are embeddings used on variable length text.

**Contact Information**

**Yuhao Li –** [**zjkli@ucdavis.edu**](mailto:zjkli@ucdavis.edu)

**Alex Whelan –** [**awhelan@ucdavis.edu**](mailto:awhelan@ucdavis.edu)

**Kai Yan –** [**kaiyan@ucdavis.edu**](mailto:kaiyan@ucdavis.edu)

**Glossary**

**1.**

**2.**

**Appendix**

**{INSERT REQUIREMENT DOC}**