APPLICATION OF DEEP LEARNING TECHNIQUES ON POLITICAL COMMENTARY

A Project Presented to

The Department of Electronics and Computer Science

At the University of Missouri



In Partial Fulfilment

of the Requirement for the Degree

Master of Science



by

ADITYA PARASHAR

Dr. Jeffrey Uhlmann, Project Supervisor

Dec 2017

Contents

[ABSTRACT 3](#_Toc498556526)

[Chapter 1 4](#_Toc498556527)

[1. Introduction 4](#_Toc498556528)

[1.1. Motivation 4](#_Toc498556529)

[Chapter 2 4](#_Toc498556530)

[1. Atmosphere - Cyverse Resources 4](#_Toc498556531)

[2. Background Work 5](#_Toc498556532)

[Chapter 3 7](#_Toc498556533)

[1. Question to solve 7](#_Toc498556534)

[2. Experiments conducted 7](#_Toc498556535)

[2.1 Environment Setup 7](#_Toc498556536)

[2.2 Collecting Data 8](#_Toc498556537)

[2.3 Document Statistics 8](#_Toc498556538)

[2.4 Tf-idf Algorithm 8](#_Toc498556539)

[Chapter 4 10](#_Toc498556540)

[Results 10](#_Toc498556541)

[Chapter 5 10](#_Toc498556542)

[Summary 10](#_Toc498556543)

[Further work 10](#_Toc498556544)

[Bibliography 10](#_Toc498556545)

[Appendix 11](#_Toc498556546)

# ABSTRACT

Natural language processing (NLP) has been used across different disciplines to process large amounts of data but using NLP alone is not sufficient to understand, analyze, and distinguish textual data. Specifically, in instances of political discourse, there are a lot of similarities and subtle differences in text. This project describes one such algorithm that can be applied to two distinctly different political commentary narratives.

Implementing term frequency-inverse document frequency (tf-idf) and using deep learning, this project views different data fields and finds similarity scores from the commentaries. Deep learning has been used across multiple different areas to learn and work on tagged datasets. Based on this, deep learning models were used to determine the similarity of testing corpora on a trained model.

The focus of this project is to create a model trained on political commentary and to identify whether this model can pinpoint the similarity of non-trained data with political discourses to aid understanding of such discourses. Additionally, an attempt to determine if specific implementations of deep learning to further help the current understanding of political thought processes. The deep learning tool called Gensim was used to understand, evaluate, and create topics from political commentaries.

# Chapter 1

## Introduction

## Motivation

Machine Learning is being used across multiple areas to find patterns and train models. The fields in which it can be applied seems to be steadily increasing. Machine Learning algorithms are also being used to provide insight and help understand text, words, and documents.

The most widely used methods in text analyses include word to vector methods, such as latent Dirichlet allocation (LDA) and latent semantic index (LSI). Both of these methods use a procedure where the words present in documents are converted to vectors and the algorithm determines the relative distance or proximity of those words. The texts are then compared to measure the similarity.

For this project, a tool called Gensim was used to work on text data. Gensim is a tool that is used to realize unsupervised semantic modeling from plain text. Gensim, as an API, provides the ability to use LDA to measure similarity distances.

# Chapter 2

## Atmosphere - Cyverse Resources

Atmosphere is a cloud-hosted environment specifically designed for use by research and doctoral students. For this project, a cloud-hosted Unix environment which had 8 GB of RAM and a single core processor was used. The python library was ported to an x64-based processor and the hosted environment provided pre-defined images which could be started and suspended as needed.

Also, the installed images of Ubuntu were present on multiple servers. This configuration provided a heterogeneous environment which had multiple general and specialized machines.

## Background Work

1. Distributed Representations of Sentences and Documents - Quoc Le, Tomas Mikolov

This paper discusses using paragraph vectors instead of word or document vectors. As a matter of fact, this method can be applied to variable-length pieces of texts ranging from a phrase or sentence to a large document. This method will eventually be used to predict the next word in the document.

The algorithm uses unlabeled data and thus can be used for tasks which do not involve labeled information. The datasets used are a Stanford Sentiment Dataset and an IMDB Dataset. Though this method involved using structured datasets, the computation for it is deemed high since it is an unsupervised algorithm.

1. A Hybrid Document Feature Extraction Method Using Latent Dirichlet Allocation and Word2Vec - Zhibo Wang, Long Ma, and Yanqing Zhang

This paper combines the approaches of LDA and word2vec to generate relationships between documents and topics. The ability to classify documents and obtain features that create the relationships is useful for performance and for word prediction. Using hybrid methods seems to perform better when it is with regard to classification and discrimination. Newsgroup datasets were used to train LDA and word2vec to extract latent topics and word vectors. Performance and error rates were calculated via benchmarking and to contrast other methods, such as SVMs and tf-idf.

1. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links – Davi

Milne, Ian H. Witten

This paper evaluates a new method to find similarity. Wikipedia Link-based Measure uses hyperlinks on Wikipedia to create connections and identifies similarity across articles. Using heuristic approaches and statistical commonness, articles are identified and earmarked for relatedness. Also, using benchmarking against tf-idf or google similarity distance as opposed to using word similarity, such as Wikipedia Link-based Measure(WLM) is less computationally expensive.

1. Evaluating WordNet-based Measures of Lexical Semantic Relatedness - Alexander Budanitsky,

Graeme Hirst

This paper highlights three types of approaches that measure semantic relatedness.

The first is a theoretical examination of a proposed measure for those mathematical properties thought desirable, such as whether it is a metric, whether it has singularities or whether its parameter-projections are smooth functions.

The second kind of evaluation is comparison with human judgments. Insofar as human judgments of similarity and relatedness are deemed to be correct by definition, this clearly gives the best assessment of the “goodness” of a measure.

The third approach is to evaluate the measures with respect to their performance in the framework of a particular application. If some particular NLP system requires a measure of semantic relatedness, we can compare different measures by seeing which one the system is most effective with, while holding all other aspects of the system constant.

# Chapter 3

### Question to solve

The use of Natural Language Processing to understand human language has been used across multiple areas. Specifically, this includes an understanding of how the transcripts and commentary involve some context in a large number of corpora. This is especially true in the case of political commentary where NLP can be used to understand disparate views. There are very few applications that specifically look at what differentiates the opposing views on political reports.

<<This question needs to be expanded upon>>

## Experiments conducted

On my project, I started out by creating an environment on atmosphere. The following sections describe the implementation processes.

## Environment Setup

Atmosphere provided an Ubuntu server computing resource. Since Gensim, as a Natural Language processing package, requires python 2.7, all the packages installed were for python 2.7. The dependent packages that were also installed include the following:

GENSIM DEPENDENT PACKAGES

|  |  |
| --- | --- |
| Package Name | Version |
| gfortran | 7.2.0 |
| libopenblas-dev | 0.2.19-3 |
| liblapack-dev | 3.2 |
| scipy | 1.0.0 |
| numpy | 1.13.3 |

Gensim was used as the package since it is the most widely used topic modeling API. As a library, it has multiple deep learning models which can be used and implemented. Gensim and tensor flow as libraries are the most advanced and extensively used libraries. The wide adoption of the modeling API also provides support and maintenance of the implementation.

## Collecting Data

The data for this project was taken from transcripts that were freely available on news websites. The data was extracted as text files into labeled corpora. To extract just the raw text, the text files were processed using a data cleansing operation. Since the algorithm takes in documents as strings there were a few pre-processing steps that had to be done to ensure that the documents were indeed ready to be analyzed.

1. Saving the commentary as “.txt” files to ensure they could be read into UTF-8 encoded files
2. Removing carriage returns and newlines
3. Removing special characters – asterisk(\*), single quotes (‘), double quotes (“) and ticks (`)

From the two websites, the above four characters were the only ones which demarcated the text as invalid corpora.

## Document Statistics

All the commentary which was analyzed was obtained from the month of August 2017. The average size of each commentary was 1600 words. The total size of all the corpora was 25,176 words. <Add here>

## Tf-idf Algorithm

The most commonly used algorithm to analyze textual corpora is the Term Frequency – Inverse Document Frequency algorithm. Tf-idf is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The tf-idf value increases proportionally to the number of times a word appears in the document and is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Tf-idf is one of the most popular term-weighting schemes. Tf-idf involves the product of two statistics:

* + 1. Term frequency: In the case of the term frequency tf(t,d), the simplest choice is to use the raw count of a term in a document, i.e. the number of times that term t occurs in document d.



2.4.2 Inverse document frequency idf is a measure of how much information the word provides, that is, whether the term is common or rare in all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term and then taking the logarithm of that quotient.



<< explain the terms>>

The above algorithm was implemented using the gensim API. Following were the steps followed to identify similarity in the corpora:

1. The ground truth for similarity was one corpus of Rachel Maddow and Sean Hannity each.
2. The baselines were filtered and the pre-processing steps were performed on each of the ground truths.

The model was trained on the ground truth.

1. After which the corpora from MSNBC and Fox were preprocessed.
2. The final step involved using the above pre-processed test corpora to evaluate the similarity index.

Execution time and infrastructure

Results : how fox contributors are pretty much the same. But the msnbc ones are not as similar. And also how contributors dictate the similarity more than the host.

# Chapter 4

## Results

# Chapter 5

## Summary

## Further work

# Bibliography

1. Distributed Representations of Sentences and Documents - Quoc Le, Tomas Mikolov
2. A Hybrid Document Feature Extraction Method Using Latent Dirichlet Allocation and Word2Vec - Zhibo Wang, Long Ma, and Yanqing Zhang
3. An Effective, Low-Cost Measure of Semantic Relatedness obtained from Wikipedia Links - David Milne, Ian H. Witten
4. Evaluating WordNet-based Measures of Lexical Semantic Relatedness - Alexander Budanitsky, Graeme Hirst
5. Semantic diversity: A measure of semantic ambiguity based on variability in the contextual usage of words - Paul Hoffman & Matthew A. Lambon Ralph & Timothy T. Rogers
6. Detecting Opinion Leaders and Trends in Online Social Networks - Freimut Bodendorf and Carolin Kaiser
7. Broadly speaking: Vocabulary in semantic dementia shifts towards general, semantically diverse words - Paul Hoffman, Lotte Meteyard, Karalyn Patterson
8. Visualizing the Signatures of Social Roles in Online Discussion Groups – JoSS Article: Volume 8, Howard T. Welser, Eric Gleave, Danyel Fisher and Marc Smith

# Appendix

*# -\*- coding: utf-8 -\*-*

import gensim

from gensim import corpora, models, similarities

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from datetime import datetime

import re

startTime = datetime.now()

raw\_documents = [""]

gen\_docs = [[w for w in word\_tokenize(text)] for text in raw\_documents]

*#print(gen\_docs)*

dictionary = gensim.corpora.Dictionary(gen\_docs)

corpus = [dictionary.doc2bow(gen\_doc) for gen\_doc in gen\_docs]

tf\_idf = gensim.models.TfidfModel(corpus, normalize=False)

sims = gensim.similarities.Similarity("~/Documents/bworkingdoc.txt", tf\_idf[corpus], num\_features=**len**(dictionary))

*#print(sims)*

query\_doc = [w for w in word\_tokenize("")]

query\_doc\_bow = dictionary.doc2bow(query\_doc)

*#print(" this is query doc bow", query\_doc\_bow)*

query\_doc\_tf\_idf = tf\_idf[query\_doc\_bow]

*#print(query\_doc\_tf\_idf)*

**print**(list(**enumerate**(sims[query\_doc\_tf\_idf])))

**print** datetime.now() – startTime

|  |
| --- |
|  |
|  | from flask import Flask  from flask\_restplus import Resource, Api, fields, marshal\_with |
|  | import newspaper,json |
|  | from newspaper import Article |
|  | import nltk |
|  | app = Flask(\_\_name\_\_) |
|  | api = Api(app) |
|  |  |
|  | @api.route('/api/v1/get-articles/<path:site>') |
|  | @api.doc(params={'site':"Site name with http prefix Ex:http://ndtv.com"}) |
|  | class ArticleList(Resource): |
|  | def get(self,site): |
|  | print site |
|  | paper=newspaper.build(site) |
|  | articles={} |
|  | i=0 |
|  | for article in paper.articles: |
|  | articles[i]={} |
|  | articles[i]['url']=article.url |
|  | i=i+1 |
|  | return {'size':i,'articles':articles} |
|  |  |
|  | @api.route('/api/v1/feed-url/<path:site>') |
|  | class FeedList(Resource): |
|  | def get(self,site): |
|  | paper=newspaper.build(site) |
|  | feed\_urls={} |
|  | i=0 |
|  | for feed in paper.feed\_urls(): |
|  | print feed |
|  | print type(feed) |
|  | feed\_urls[i]=feed |
|  | i=i+1 |
|  | return {'size':i,'feed\_urls':feed\_urls} |
|  |  |
|  |  |
|  | @api.route('/api/v1/scrape-article/<path:url>/<string:name>/<string:profession>') |
|  | class ArticleInfo(Resource): |
|  | def get(self,url,name,profession): |
|  | article = Article(url) |
|  | article.download() |
|  | article.parse() |
|  | article.nlp() |
|  | article\_data = {} |
|  | article\_data['url']=url |
|  | article\_data['title']=article.title |
|  | article\_data['keywords']=article.keywords |
|  | article\_data['summary']=article.summary |
|  | article\_data['text']=article.text |
|  | article\_data['top\_image']=article.top\_image |
|  | article\_data['publish\_date']=str(article.publish\_date) |
|  | article\_data['authors']=article.authors |
|  | article\_data['movies']=article.movies |
|  | article\_data['html']=article.html |
|  | ################################### |
|  | ###### ToDo: NLP checks goes here |
|  | for sent in nltk.sent\_tokenize(article.text): |
|  | for chunk in nltk.ne\_chunk(nltk.pos\_tag(nltk.word\_tokenize(sent))): |
|  | if hasattr(chunk, 'node'): |
|  | if chunk.node=="PERSON": |
|  | print chunk.node, ' '.join(c[0] for c in chunk.leaves()) |
|  | nameHit=False |
|  | profHit=False |
|  | firstNameHit=False |
|  | lastNameHit=False |
|  | nameList=name.split() |
|  | for x in article.text.split(): |
|  | if x.lower()==nameList[0].lower(): |
|  | firstNameHit=True |
|  | if x.lower()==nameList[1].lower(): |
|  | lastNameHit=True |
|  | if x.lower()==profession.lower(): |
|  | profHit=True |
|  | if firstNameHit==True and lastNameHit==True: |
|  | nameHit=True |
|  | return {'article':article\_data,'name\_hit':nameHit,'profession\_hit':profHit} |
|  | @api.route('/index') |
|  | class Home(Resource): |
|  | def get(self): |
|  | return {'hello': 'world'} |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | app.run(debug=True) |