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](#_Toc498939947)

[Chapter 1 4](#_Toc498939948)

[1. Introduction 4](#_Toc498939949)

[1.1. Motivation 4](#_Toc498939950)

[Chapter 2 5](#_Toc498939951)

[1. Atmosphere - Cyverse Resources 5](#_Toc498939952)

[Comparison of Cyverse with non-hosted environments 5](#_Toc498939953)

[2. Related Work 6](#_Toc498939954)

[Chapter 3 11](#_Toc498939955)

[1. Question to solve 11](#_Toc498939956)

[2. Experiments conducted 11](#_Toc498939957)

[2.1 Environment Setup 11](#_Toc498939958)

[2.2 Collecting Data 12](#_Toc498939959)

[2.3 Document Statistics 13](#_Toc498939960)

[2.4 Tf-idf Algorithm 13](#_Toc498939961)

[Chapter 4 15](#_Toc498939962)

[Results 15](#_Toc498939963)

[Chapter 5 15](#_Toc498939964)

[Summary 15](#_Toc498939965)

[Further work 15](#_Toc498939966)

[Bibliography 15](#_Toc498939967)

[Appendix 16](#_Toc498939968)

# 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(ML) is used across multiple areas and fields to find patterns and train models. The trained models are eventually used to understand and derive high quality information. ML algorithms can be used across images or text. Images are inherently ambiguous, words however are semi-structured and they also contain information about itself. Which makes working on text much easier since the metadata That said, textual analysis has presented many challenges for machine learning. Laborious, manual feature extraction is the main disadvantage.

The most widely used methods in text analyses include word to vector methods, such as latent Dirichlet allocation (LDA) and latent semantic indexing (LSI). All of these methods use a procedure where each document is considered to have a set of topics that are assigned to it via LDA and the algorithm determines the relative distance or proximity of those documents. The texts are then compared to measure similarity.

On my current project I used political commentaries to find out how similar they were. This was done using a baseline of two distinctly disparate corpora. Finding out similarity scores against ground truth helped identifying what the parameters were in identifying the likeness [4].

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.

With the proliferation of online, social media the ability to identify and associate based on political affiliation helps in understanding sources. <<Add more here>>

# 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.

## Comparison of Cyverse with non-hosted environments

|  |  |  |
| --- | --- | --- |
| Config/Parameter | Non-hosted Environments | Cyverse |
| Memory Management |  |  |
| Virtual Memory |  |  |
| Processors |  |  |
| Interrupts |  |  |

<<Table showing how significant and what the advantages are..>>

## Related Work

The main goal of this project was to determine how similar political commentaries were. Semantic similarity is a widely studied and analyzed research problem. The previously studied methods predominantly start out with a generic model and become tailored to suit specific contexts.

The semantic usage of words and how words present in a document determine the importance of each word. Word Maturity Model[1] expresses the maturity of a word as the function of two

Parameters: word “w” and age “e”. Given a word w and an age e, the function estimates the maturity of the word at that age, and also establishes when that word has a mature representation. This model computationally models meaning, maturity and quantitatively evaluates the quality of the text on the basis of the model. This methodology, called Word Maturity (WM), models the maturity of the lexicon by analyzing the trajectory of a word's representation at several points in time. The WM model was based on the premise that word meaning varies with age in a continuous manner. This model makes no sense to talk about the meaning of a term as something static and absolute, but rather it would develop as it is exposed to new lexical contexts.

To add to lexical contexts there has been a significant amount of work done which involves taking the order and the structure of the words present in the corpora. To add to classifiers there is also research on learning based on structure. A well-known( ??) model which involved training the classifier to pick up on indexed positions of words predicts the next word in a sentence. In this framework, every word is mapped to a unique vector, represented by a column in a matrix W. The column is indexed by position of the word in the vocabulary. The

Concatenation or sum of the vectors is then used as features for prediction of the next word in a sentence.

Classifier

on

Average / Concatenate

Word Matrix

W

W

W

W

sat

cat

the

Some of the biggest drawbacks is that techniques like word vectors only work on sentences, but not paragraphs/documents with several sentences. It is unclear how to combine the representations over many sentences. Such techniques therefore are restricted to work on sentences but not paragraphs or documents.

On the flip side there is also significant research and results that come out of models like Diffusion Theory [3] which looks describes the spread of product opinions in social systems. According to this theory, not only the product characteristics (e.g. complexity) are an important factor in the formation of opinions but also the structure of the social networks. This theory communities comprises four steps. In the first step the users’ opinions on the product are extracted by text mining. In the second step, the communication relationships among users are identified by text based relationship mining methods. The extracted users, opinions, and relationships form a social network which is represented as graph. Nodes and edges of the graph can be characterized by attributes. The nodes represent the users of a forum and the edges their communication relationships. Models like these help explain the flow of information and thought across social networks.

More recently there has also been hybrid approaches which involves a mix of paragraph vectors[a] and LDA. Our new method as shown in projects words, documents, and topics in a high-dimension semantic space. A document vector is considered as a single vector, which is the centroid of all words in the document as what Word2Vec does in the projection layer. In addition, each document has its individual length, thus its vector is divided by the number of words in the document to guarantee the measurements with same scale.

The Hybrid Document Feature Extraction Method[b] creates topics where a subset of high-probability words in each topic is employed to represent the topic, and then their probabilities are rescaled as the weights of words. Hence different words have different contributions to the topic.

1. Kireyev, K. & Landauer, T.K. (2011) Word Maturity: Computational Modeling of

Word Knowledge. Proceedings of the 49th Annual Meeting of the Association for

Computational Linguistics: Human Language Technologies. Portland, Oregon, USA

1. 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
2. Detecting Opinion Leaders and Trends in Online Social Networks - Freimut Bodendorf and Carolin Kaiser
3. Broadly speaking: Vocabulary in semantic dementia shifts towards general, semantically diverse words - Paul Hoffman, Lotte Meteyard, Karalyn Patterson
4. Visualizing the Signatures of Social Roles in Online Discussion Groups – JoSS Article: Volume 8, Howard T. Welser, Eric Gleave, Danyel Fisher and Marc Smith
5. 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 often opposite and disparate views on the same topic. There are very few applications that specifically look at what differentiates the opposing views on political reports. The project is further motivated by the need for determining similarity between small pieces of text across documents that potentially span different topics during multi-document summarization.

For this project, we define Similarity to be an index where the query vector is equivalent to the document corpora.

* 1. An OH-58 helicopter, carrying a crew of two, was on a routine training orientation when contact was lost at about 11:30 a.m. Saturday (9:30 p.m. EST Friday).
  2. (b) “There were two people on board,” said Bacon. “We lost radar contact with the helicopter about 9:15 EST (0215 GMT).”
  3. (c) An OH-58 U.S. military scout helicopter made an emergency landing in North Korea at about 9.15 p.m. EST Friday (0215 GMT Saturday), the Defense Department said.

We consider units (a) and (b) in Figure 1 to be similar, because they both focus on the same event (loss of contact) with the same primary participant (the helicopter). On the other hand, unit (c) in Figure 1 is not similar to either (a) or (b). Although all three refer to a helicopter, the primary focus in (c) is on the emergency landing rather than the loss of contact.

## 2 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. It is a simple transformation which takes documents represented as bag-of-words counts and applies a weighting which discounts common terms (or, equivalently, promotes rare terms). It also scales the resulting vector to unit length 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.



Where:

tf(*t*,*d*) = term frequency

*ft*,*d* = raw count of the number of terms in the document

t = term in the document

d = document

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.



Where:

N = total number of documents in the Corpus, N = |D|

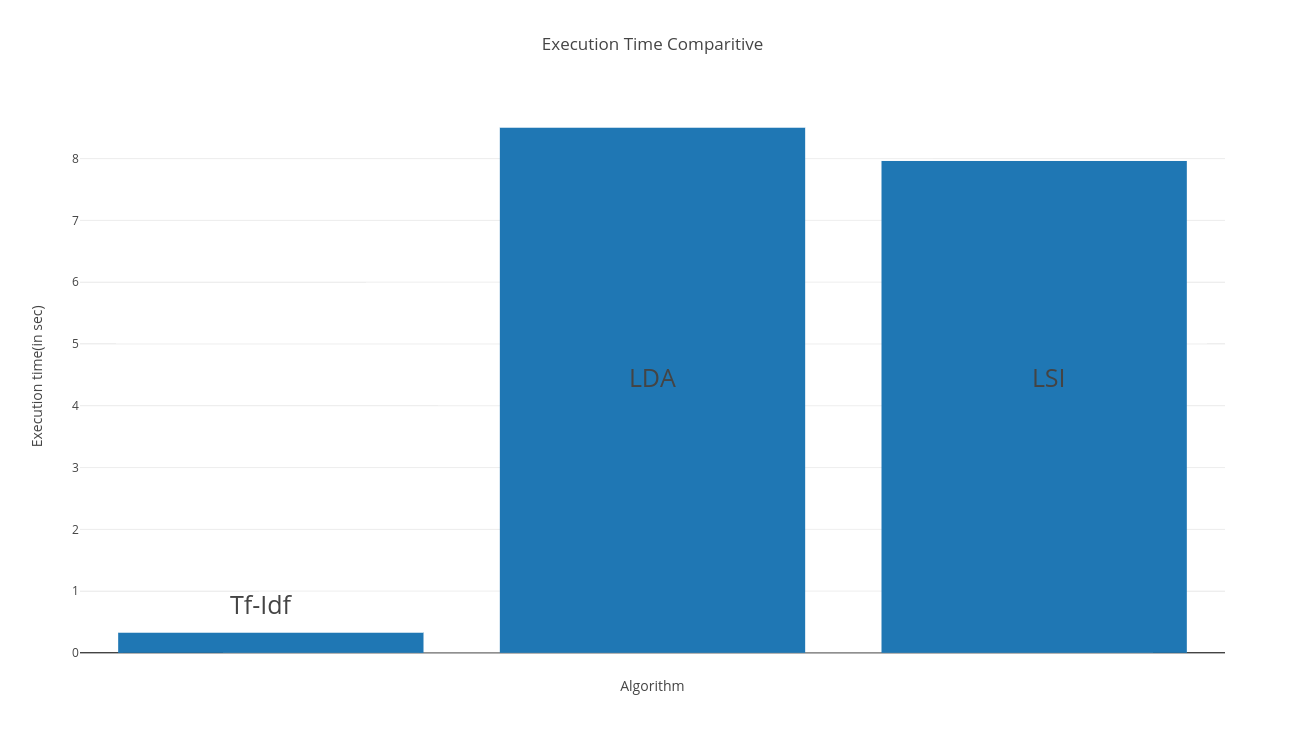
The above algorithm was implemented using the gensim API. Following were the steps followed to identify similarity in the corpora:

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.

Rationale behind Using Tf-Idf :



Execution time and infrastructure

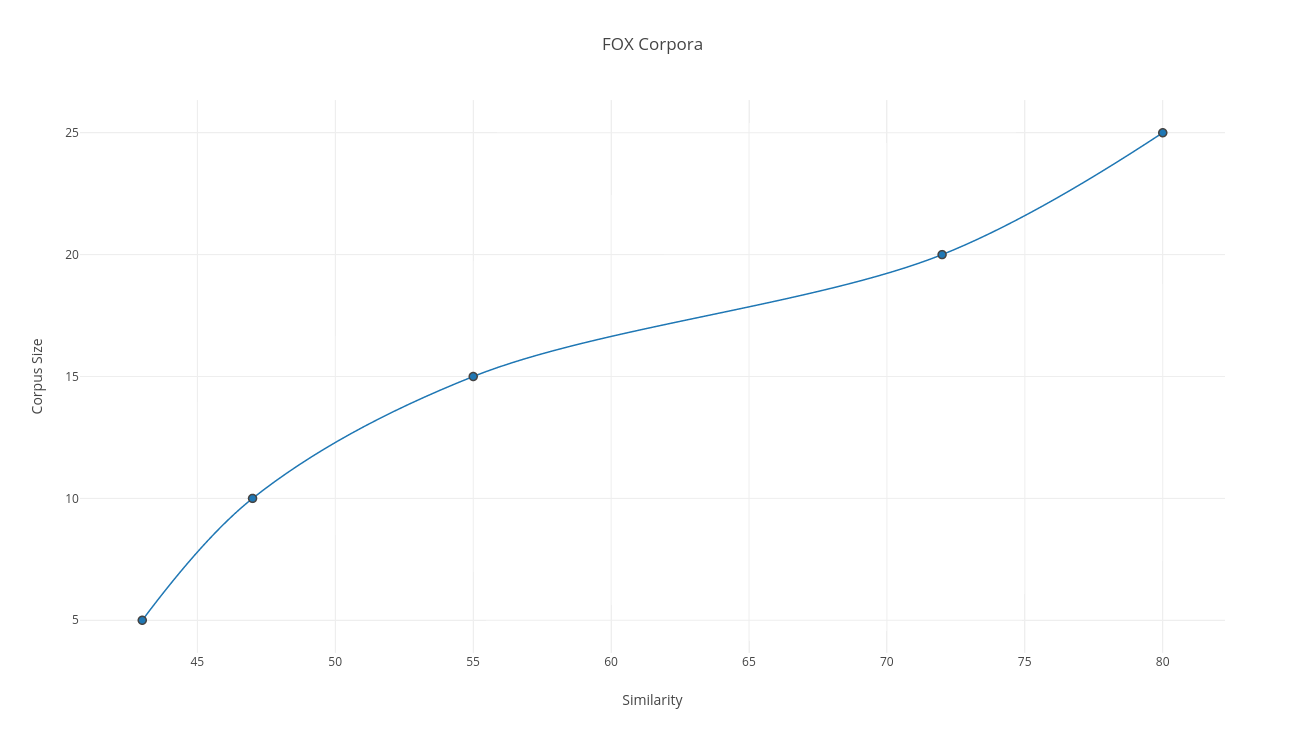
Results : how fox contributors are pretty much the same. But the msnbc ones are not as similar. And also how contributors dictate the similarity more than the host.

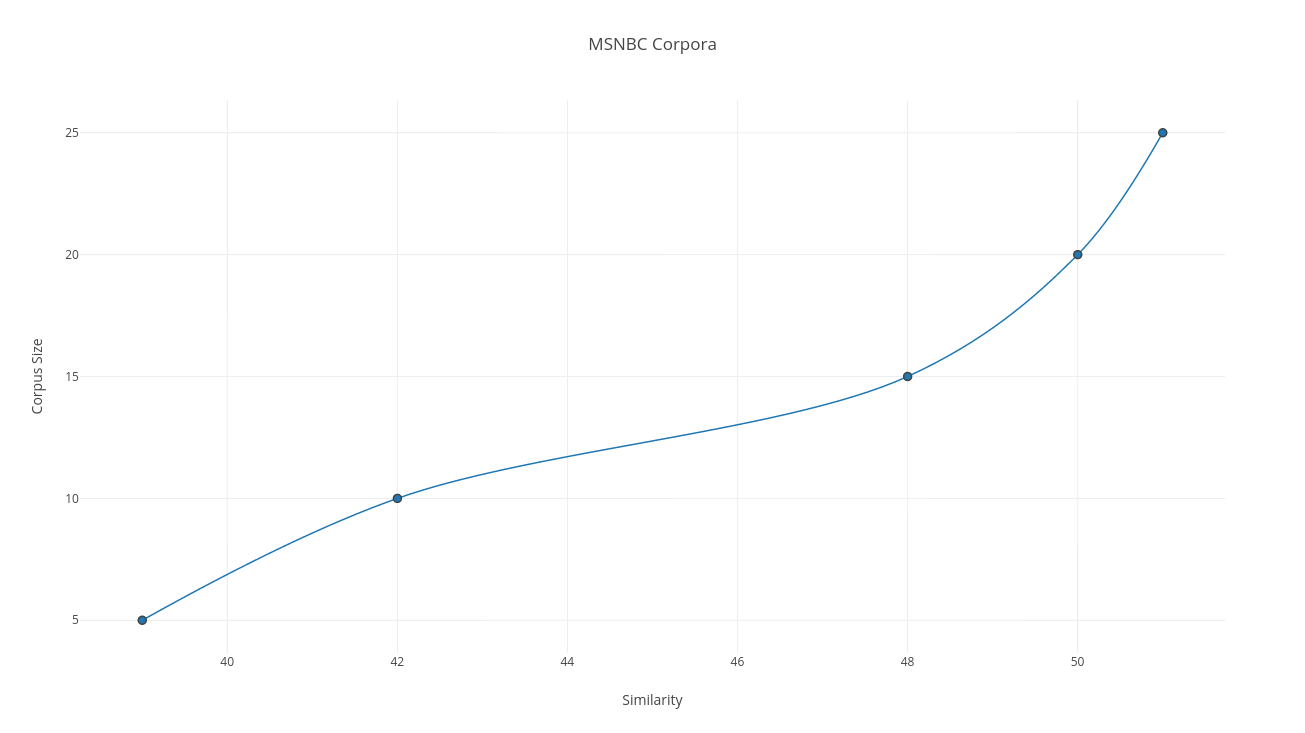
# Chapter 4

## Experiments conducted

## Results

<<table with the results>>





Using the right amount and number of samples, how sensitivity is even measured

# Chapter 5

## Summary

Automated measurement of the similarity between text documents is fundamentally a psychological modeling problem. This paper presents an assessment of keyword, n-gram and LSA approaches against human data for a small corpus of short news documents. In this project there were assumptions made which included that stopwords be removed and that the effect that it would have on the similarity would be insignificant. It is clear that when the model judges two documents from the FOX Corpus to be highly similar, it is correct. However there is only a 50% likelihood of the similarity with the CNBC corpora.

Aside from the similarity that arose out of using the Tf-Idf algorithm, it shows that as the model gets more data and more text the similarity goes up. The model itself was trained on corpora which were from the month of August for both FOX and MSNBC, which makes the model to be data agnostic.

Both the FOX and MSNBC results had the same number of training examples. Also there was no background information that was fed to the model itself.

The most obvious is that the lower the number of corpora, the similarity that results from it is also low. While the Fox news similarity increased as the size of corpus increased, the same was not the case with MSNBC. This gives empirical evidence that the greater informative the training data is, the greater the advantage in having a corpus of background knowledge available for use during classification.

## Further work

<Using topics to find out how and what is similar across fox and msnbc>

<what if there is background knowledge that is added AND/OR the model gets recursively created >

<Also use LDA, LSI to compare and contrast it with tf-idf>

More sophisticated representations might be able to identify the common features between the highly similar document pairs currently being missed.

# 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) |