



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Analysis and contrasting of different english news channel shows using Artificially Intelligent Natural Language Processing methodologies

Siddharth Garg

18BCB0038

Aim

To develop a way to analyze primetime news using Artificial Intelligence. Using exploratory data analysis, sentiment analysis, and finally topic modelling using *Latent Dirichlet allocation* Algorithm.

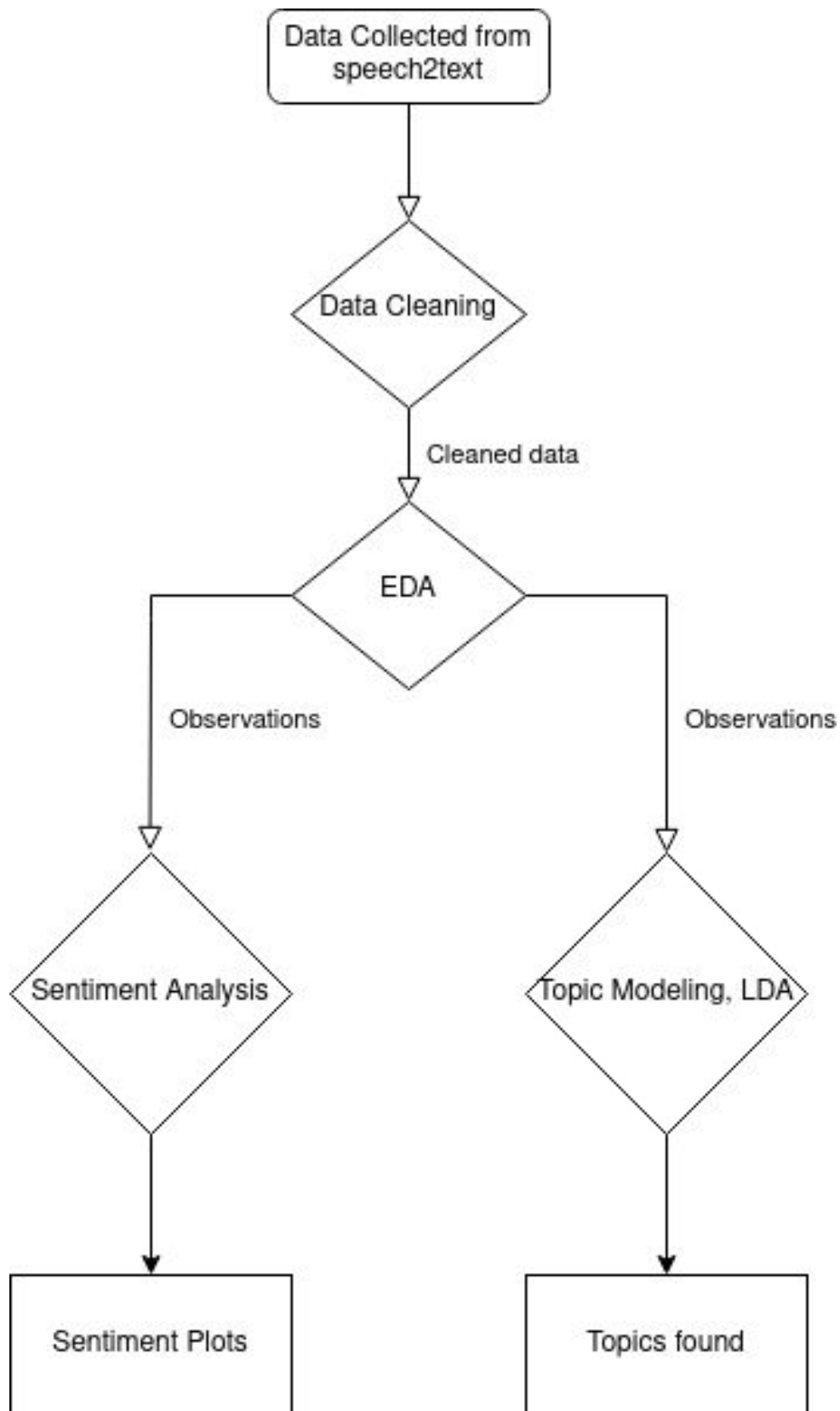
Introduction

Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.

The data collected was generated from official youtube channels of respective news channels using speech to text technology.

The project consisted of four phases:

1. Data Cleaning
2. Exploratory Data Analysis
3. Sentiment Analysis
4. Topic Modelling



Abstract

I. Data Cleaning

As a data scientist, we may use NLP for sentiment analysis (classifying words to have positive or negative connotation) or to make predictions in classification models, among other things. Typically, whether we're given the data or have to scrape it, the text will be in its natural human format of sentences, paragraphs, tweets, etc. From there, before we can dig into analyzing, we will have to do some cleaning to break the text down into a format the computer can easily understand.

1. Remove HTML
2. Tokenization + Remove punctuation
3. Remove stop words
4. Lemmatization

II. EDA

In statistics, exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

Exploratory data analysis is one of the most important parts of any machine learning workflow and Natural Language Processing is no different.



(Word Cloud depicting frequency of tokens occurring in each channel debate)

III. Sentiment Analysis

Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

Sentiment analysis refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

IV. Topic Modelling

Topic modeling is an unsupervised machine learning technique that's capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize a set of documents.

Latent Dirichlet allocation

In natural language processing, the latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a topic model and belongs to the machine learning toolbox and in wider sense to the artificial intelligence toolbox.

Literature Review

An overview of topic modeling and its current applications in bioinformatics.

Liu, L., Tang, L., Dong, W. *et al.* *SpringerPlus* **5**, 1608 (2016).

<https://doi.org/10.1186/s40064-016-3252-8>

This paper starts with the description of a topic model, with a focus on the understanding of topic modeling. A general outline is provided on how to build an application in a topic model and how to develop a topic model. Meanwhile, the literature on application of topic models to biological data was searched and analyzed in depth. According to the types of models and the analogy between the concept of document-topic-word and a biological object (as well as the tasks of a topic model), we categorized the related studies and provided an outlook on the use of topic models for the development of bioinformatics applications.

Online News Media Bias Analysis using an LDA-NLP Approach

Sarjoun Doumit and Ali Minai School of Electronic & Computing Systems, College of Engineering, University of Cincinnati, Ohio 45221-0030, U.S.A. (email: doumitss@mail.uc.edu & ali.minai@uc.edu).

It is widely recognized that every media outlet has its own "spin" on news, and this bias has been described in many ways and at many levels. In political news for example, the bias can be liberal, conservative, moderate, corporate, etc. In addition, recent research has focused on the 'sentiment dimension' to further identify and categorize news bias. This is achieved through analysis of the adjective and adverb terms found in the news texts. The accuracy and generality of these models depend on the evaluation methods used to appraise the intensity and emotional weights of the adjectives and adverbs, thus rendering the results open to controversy. In this paper we propose a unifying system to extract information from political news texts and analyze it within a cognitive network. We view the different news media sources as agents with unique personalities, which we assume are latent within their texts. We use a combination of Latent Dirichlet Allocation (LDA) and natural language processing (NLP) methods to identify the different agents' personality traits with respect to various topics or concepts. An agent's personality traits affect its inclination to word a certain event in a specific way. Using the common concepts stored in the cognitive network, our system can compare the different agents on a unified and normalized platform.



Opinion Mining and Sentiment Analysis.

Bo Pang, Yahoo! Research, USA, bopang@yahoo-inc.com Lillian Lee, Computer Science Department, Cornell University, USA, llee@cs.cornell.edu

An important part of our information-gathering behavior has always been to find out what other people think. With the growing availability and popularity of opinion-rich resources such as online review sites and personal blogs, new opportunities and challenges arise as people now can, and do, actively use information technologies to seek out and understand the opinions of others. The sudden eruption of activity in the area of opinion mining and sentiment analysis, which deals with the computational treatment of opinion, sentiment, and subjectivity in text, has thus occurred at least in part as a direct response to the surge of interest in new systems that deal directly with opinions as a first-class object.

This survey covers techniques and approaches that promise to directly enable opinion-oriented information-seeking systems. Our focus is on methods that seek to address the new challenges raised by sentiment-aware applications, as compared to those that are already present in more traditional fact-based analysis. We include material on summarization of evaluative text and on broader issues regarding privacy, manipulation, and economic impact that the development of opinion-oriented information-access services gives rise to. To facilitate future work, a discussion of available resources, benchmark datasets, and evaluation campaigns is also provided.

Proposed Method

Latent Dirichlet allocation

There has been great interest in Latent Dirichlet Allocation or LDA ever since the publication of the seminal paper by Blei, Ng and Jordan[2]. It is a machine learning technique (shown in extended version in Fig.1), that evolved from a previous model called Probabilistic Latent Semantic Analysis[6] (pLSA) for reducing the dimensionality of a certain textual corpus while preserving its inherent statistical characteristics. LDA assumes that each document in a corpus can be described as a mixture of multiple latent topics which, in turn, are distributions over words found in the documents of the corpus. LDA assumes that documents are made of a list of words where the order of the words is not important i.e. a bag-of-words approach. LDA is a generative model in that it can generate a document from a set of topics, but it can also be used as an inference tool to extract topics from a corpus of documents. To generate a corpus of D documents, where each document has N_d words, and for a total of T topics, LDA's generative algorithm is:

1. Pick a document size $N_d \geq 253$
2. Pick a set of topics $\theta \sim \text{Dirichlet}(\alpha)$
 - (a) Draw a topic $t_w \sim \text{Multinomial}(\theta)$
 - (b) Draw a word $w_n \sim \text{Multinomial}(\phi_{t_w})$, where $\phi \sim \text{Dirichlet}(\beta)$.

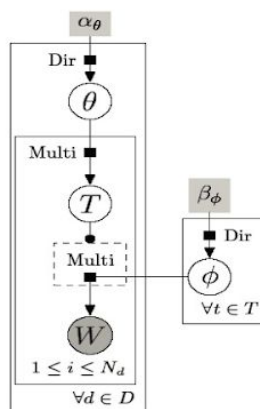


Figure 1: Directed factor graph for LDA. ϕ is the words-topic distribution, θ represents the topics-document distribution, α is the *Dirichlet* hyperparameter for θ , and finally β is the *Dirichlet* hyperparameter for ϕ

We start by defining how we view the process of political news story generation and how bias gets embedded into the very fabric of the original factual news. Let E represent a factual event, and \bar{E} the smallest set of factual clauses, c_k , that can summarize event E , where $k = 1, \dots, K$. A factual clause contains the basic semantic elements to describe the event or parts of the event without any use of adjectives or adverbs which carry sentiment, so \bar{E} represents the most unbiased text report of the event. Let O represent the existing media outlets as a population of agents such that $O = \bigcup_{i=1}^n O_i$, where O_i is an identifiable media outlet with identity i and n is the total number of media outlets in the environment. Each O_i is characterized by a set of subjects S_i , where each subject S_{ij} has a bias B_{ij} represented as a set L_{ij} of K dependent clauses d_k , which could be either an adjective clause or an adverb clause. Thus, each dependent clause carries a sentiment, and the pattern of sentiment over L_{ij} defines agent i 's bias towards subject j . We assume that every agent has a bias for every subject in its repertoire, though agents may vary in their subjects. When reporters who work for a specific media outlet O_i come across a factual event E , they first determine what subject S_{ij} this event is relevant to. Then, armed with the specific bias B_{ij} for S_{ij} , they initiate the process of creating a news story using the bias in conjunction with the unbiased description \bar{E} to create a biased version of E . This is denoted by E_i , i.e., source i 's (biased) report of event E . The average news reader receives this biased story as the final product. Our aim is to analyze the biased news story and try to isolate the biases or the L_{ij} which includes the adjectives and adverbs in order to compare them on a deeper semantic/cognitive level instead of making a purely lexical-weighted comparison. We visualize the process of weaving the bias into the factual event as a combination of three sub-processes which we call 'Actor-assignment', 'Action-assignment' and 'Sentiment-assignment'. These processes work together as single coordinated dynamical units. We call these meme-synergies (Fig. 2) in analogy with the muscle synergies that underlie motor control in animals. Muscle synergies are coordinated musculoskeletal degrees of freedom that are jointly controlled as a unit, and form the primitives from which more complex movements are constructed. Just as muscle synergies combine to move one's arm to hit a tennis ball in a particular style, meme-synergies combine effortlessly to generate coherent textual news products with a certain bent. In our model, we consider a media's biased clauses or embedded memes to be preconfigured to trigger or produce text-generation in accordance with those of the original event E , especially amongst those that share a similar construct such as a concept or sentiment. This correspondence of thought and action represents a deeply embodied view of human cognition.

Codebase

Data Cleaning

```
#!/usr/bin/env python
import pickle

channels = ['ndtv', 'indiatoday', 'republic']
data = {}
for c in (channels):
    with open("transcripts/" + c + ".txt", "rb") as file:
        data[c] = [file.read().decode("utf-8") ]

import pandas as pd
pd.set_option('max_colwidth',150)

data_df = pd.DataFrame.from_dict(data).transpose()
data_df.columns = ['transcript']
data_df = data_df.sort_index()
Data_df
data_df.transcript.loc['ndtv']
import re
import string

def clean_text_round1(text):
    '''Make text lowercase, remove text in square brackets,
    remove punctuation and remove words containing numbers.'''
    text = text.lower()
```

```
text = re.sub('\[.*?\]', '', text)
text = re.sub('[%s]' % re.escape(string.punctuation), '',
text)
text = re.sub('\w*\d\w*', '', text)
return text

round1 = lambda x: clean_text_round1(x)
data_clean = pd.DataFrame(data_df.transcript.apply(round1))
def clean_text_round2(text):
    '''Get rid of some additional punctuation and non-sensical
text that was missed the first time around.'''
    text = re.sub('[\'\"“”…]', '', text)
    text = re.sub('\n', '', text)
    return text

round2 = lambda x: clean_text_round2(x)
data_clean = pd.DataFrame(data_clean.transcript.apply(round2))
data_df.to_pickle("corpus.pkl")
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(stop_words='english')
data_cv = cv.fit_transform(data_clean.transcript)
data_dtm = pd.DataFrame(data_cv.toarray(),
columns=cv.get_feature_names())
data_dtm.index = data_clean.index
data_dtm
data_dtm.to_pickle("dtm.pkl")
data_clean.to_pickle('data_clean.pkl')
pickle.dump(cv, open("cv.pkl", "wb"))
```

EDA

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
import pandas as pd
```

```
data = pd.read_pickle('dtm.pkl')
```

```
data = data.transpose()
```

```
data.head()
```

```
# In[2]:
```

```
top_dict = {}
```

```
for c in data.columns:
```

```
    top = data[c].sort_values(ascending=False).head(30)
```

```
    top_dict[c]= list(zip(top.index, top.values))
```

```
top_dict
```

```
# In[3]:
```

```
for channel, top_words in top_dict.items():
```

```
print(channel)
print(', '.join([word for word, count in top_words[0:14]]))
print('---')
```

```
# In[4]:
```

```
from collections import Counter
```

```
# Let's first pull out the top 30 words for each comedian
```

```
words = []
```

```
for comedian in data.columns:
```

```
    top = [word for (word, count) in top_dict[comedian]]
```

```
    for t in top:
```

```
        words.append(t)
```

```
words
```

```
# In[5]:
```

```
Counter(words).most_common()
```

```
# In[6]:
```

```
add_stop_words = [word for word, count in Counter(words).most_common() if count > 6]
add_stop_words
```

```
# In[16]:
```

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
# In[17]:
```

```
add_stop_words=stop_words
add_stop_words
```

```
# In[18]:
```

```
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer

# Read in cleaned data
data_clean = pd.read_pickle('data_clean.pkl')
```

```
# Add new stop words
stop_words = text.ENGLISH_STOP_WORDS.union(add_stop_words)

# Recreate document-term matrix
cv = CountVectorizer(stop_words=stop_words)
data_cv = cv.fit_transform(data_clean.transcript)
data_stop = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())
data_stop.index = data_clean.index

# Pickle it for later use
import pickle
pickle.dump(cv, open("cv_stop.pkl", "wb"))
data_stop.to_pickle("dtm_stop.pkl")

# In[19]:

# Let's update our document-term matrix with the new list of stop words
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer

# Read in cleaned data
data_clean = pd.read_pickle('data_clean.pkl')

# Add new stop words
stop_words = text.ENGLISH_STOP_WORDS.union(add_stop_words)

# Recreate document-term matrix
```

```
cv = CountVectorizer(stop_words=stop_words)
data_cv = cv.fit_transform(data_clean.transcript)
data_stop = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())
data_stop.index = data_clean.index

# Pickle it for later use
import pickle
pickle.dump(cv, open("cv_stop.pkl", "wb"))
data_stop.to_pickle("dtm_stop.pkl")

# In[20]:

# Let's make some word clouds!
# Terminal / Anaconda Prompt: conda install -c conda-forge wordcloud
from wordcloud import WordCloud

wc = WordCloud(stopwords=stop_words, background_color="white", colormap="Dark2",
               max_font_size=150, random_state=42)

# In[22]:

# Reset the output dimensions
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = [16, 6]
```



```
full_names = ['ndtv', 'indiatoday', 'republic']

# Create subplots for each comedian
for index, comedian in enumerate(data.columns):
    wc.generate(data_clean.transcript[comedian])

    plt.subplot(3, 4, index+1)
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.title(full_names[index])

plt.show()

# There isnt much to conclude from here so lets jump to Sentiment analysis
```

Sentiment Analysis

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:

import pandas as pd

data = pd.read_pickle('corpus.pkl')
data
```

```
# In[2]:
```

```
from textblob import TextBlob
```

```
pol = lambda x: TextBlob(x).sentiment.polarity
```

```
sub = lambda x: TextBlob(x).sentiment.subjectivity
```

```
data['polarity'] = data['transcript'].apply(pol)
```

```
data['subjectivity'] = data['transcript'].apply(sub)
```

```
data
```

```
# In[9]:
```

```
[i for i in data.index]
```

```
# In[11]:
```

```
# Let's plot the results
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
import matplotlib.pyplot as plt
```

```
plt.rcParams['figure.figsize'] = [10, 8]
```

```
for index, channel in enumerate(data.index):
    x = data.polarity.loc[channel]
    y = data.subjectivity.loc[channel]
    plt.scatter(x, y, color='blue')
    plt.text(x+.001, y+.001, channel, fontsize=10)
    plt.xlim(-.01, .12)

plt.title('Sentiment Analysis', fontsize=20)
plt.xlabel('<-- Negative ----- Positive -->', fontsize=15)
plt.ylabel('<-- Facts ----- Opinions -->', fontsize=15)

plt.show()

# In[12]:

import numpy as np
import math

def split_text(text, n=10):
    """Takes in a string of text and splits into n equal parts, with a default of 10 equal
    parts."""

    # Calculate length of text, the size of each chunk of text and the starting points of
    each chunk of text
    length = len(text)
    size = math.floor(length / n)
    start = np.arange(0, length, size)
```

```
# Pull out equally sized pieces of text and put it into a list
split_list = []
for piece in range(n):
    split_list.append(text[start[piece]:start[piece]+size])
return split_list
```

```
# In[13]:
```

```
data
```

```
# In[14]:
```

```
list_pieces = []
for t in data.transcript:
    split = split_text(t)
    list_pieces.append(split)
```

```
list_pieces
```

```
# In[15]:
```

```
len(list_pieces)
```



```
# In[16]:
```

```
len(list_pieces[0])
```

```
# In[18]:
```

```
polarity_transcript = []
```

```
for lp in list_pieces:
```

```
    polarity_piece = []
```

```
    for p in lp:
```

```
        polarity_piece.append(TextBlob(p).sentiment.polarity)
```

```
    polarity_transcript.append(polarity_piece)
```

```
polarity_transcript
```

```
# In[22]:
```

```
plt.plot(polarity_transcript[0])
```

```
plt.title(data.index[0])
```

```
plt.show()
```

```
# In[29]:
```

```
# Show the plot for all comedians
plt.rcParams['figure.figsize'] = [16, 12]

for index, channel in enumerate(data.index):
    plt.subplot(3, 4, index+1)
    plt.plot(polarity_transcript[index])
    plt.plot(np.arange(0,10), np.zeros(10))
    plt.title(channel)
    plt.ylim(ymin=-.2, ymax=.3)

plt.show()

# Unusual to see every channel has been pretty positive regarding the ayodhya verdict

# In[ ]:
```

Topic Modelling

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:
```



```
import pandas as pd
import pickle

data = pd.read_pickle('dtm_stop.pkl')
data

# In[2]:

from gensim import matutils, models
import scipy.sparse

# In[3]:

tdm = data.transpose()
tdm.head()

# In[5]:

sparse_counts = scipy.sparse.csr_matrix(tdm)
corpus = matutils.Sparse2Corpus(sparse_counts)
```



```
# In[6]:
```

```
cv = pickle.load(open("cv_stop.pkl", "rb"))
id2word = dict((v, k) for k, v in cv.vocabulary_.items())
```

```
# In[7]:
```

```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=2, passes=10)
lda.print_topics()
```

```
# In[8]:
```

```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=3, passes=10)
lda.print_topics()
```

```
# In[9]:
```

```
lda = models.LdaModel(corpus=corpus, id2word=id2word, num_topics=4, passes=10)
lda.print_topics()
```




```
# This approach aint working out
```

```
# ## Nouns Only
```

```
# In[22]:
```

```
from nltk import word_tokenize, pos_tag
```

```
def nouns(text):
```

```
    """Given a string of text, tokenize the text and pull out only the nouns."""
```

```
    is_noun = lambda pos: pos[:2] == 'NN'
```

```
    tokenized = word_tokenize(text)
```

```
    all_nouns = [word for (word, pos) in pos_tag(tokenized) if is_noun(pos)]
```

```
    return ' '.join(all_nouns)
```

```
# In[23]:
```

```
data_clean = pd.read_pickle('data_clean.pkl')
```

```
data_clean
```

```
# In[26]:
```

```
data_nouns = pd.DataFrame(data_clean.transcript.apply(nouns))
```

```
data_nouns
```



```
# In[28]:
```

```
from nltk.corpus import stopwords
add_stop_words = set(stopwords.words('english'))
```


```
# In[29]:
```

```
# Create a new document-term matrix using only nouns
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer

# Re-add the additional stop words since we are recreating the document-term matrix
stop_words = text.ENGLISH_STOP_WORDS.union(add_stop_words)

# Recreate a document-term matrix with only nouns
cvn = CountVectorizer(stop_words=stop_words)
data_cvn = cvn.fit_transform(data_nouns.transcript)
data_dtmn = pd.DataFrame(data_cvn.toarray(), columns=cvn.get_feature_names())
data_dtmn.index = data_nouns.index
data_dtmn
```

```
# In[30]:
```



```
# Create the gensim corpus
corpusn = matutils.Sparse2Corpus(scipy.sparse.csr_matrix(data_dtmn.transpose()))

# Create the vocabulary dictionary
id2wordn = dict((v, k) for k, v in cvn.vocabulary_.items())

# In[31]:

# Let's start with 2 topics
ldan = models.LdaModel(corpus=corpusn, num_topics=2, id2word=id2wordn, passes=10)
ldan.print_topics()

# In[32]:

# Let's try topics = 3
ldan = models.LdaModel(corpus=corpusn, num_topics=3, id2word=id2wordn, passes=10)
ldan.print_topics()

# In[33]:

# Let's try 4 topics
ldan = models.LdaModel(corpus=corpusn, num_topics=4, id2word=id2wordn, passes=10)
```

```
ldan.print_topics()
```

```
# ## Nouns And Adjectives Both
```

```
# In[35]:
```

```
# Let's create a function to pull out nouns from a string of text
```

```
def nouns_adj(text):
```

```
    """Given a string of text, tokenize the text and pull out only the nouns and  
    adjectives."""
```

```
    is_noun_adj = lambda pos: pos[:2] == 'NN' or pos[:2] == 'JJ'
```

```
    tokenized = word_tokenize(text)
```

```
    nouns_adj = [word for (word, pos) in pos_tag(tokenized) if is_noun_adj(pos)]
```

```
    return ' '.join(nouns_adj)
```

```
# In[36]:
```

```
# Apply the nouns function to the transcripts to filter only on nouns
```

```
data_nouns_adj = pd.DataFrame(data_clean.transcript.apply(nouns_adj))
```

```
data_nouns_adj
```

```
# In[37]:
```

```
# Create a new document-term matrix using only nouns and adjectives, also remove
common words with max_df
cvna = CountVectorizer(stop_words=stop_words, max_df=.8)
data_cvna = cvna.fit_transform(data_nouns_adj.transcript)
data_dtmna = pd.DataFrame(data_cvna.toarray(), columns=cvna.get_feature_names())
data_dtmna.index = data_nouns_adj.index
data_dtmna
```

```
# In[38]:
```

```
# Create the gensim corpus
corpusna = matutils.Sparse2Corpus(scipy.sparse.csr_matrix(data_dtmna.transpose()))
```

```
# Create the vocabulary dictionary
```

```
id2wordna = dict((v, k) for k, v in cvna.vocabulary_.items())
```

```
# In[39]:
```

```
# Let's start with 2 topics
```

```
ldana = models.LdaModel(corpus=corpusna, num_topics=2, id2word=id2wordna,
passes=10)
```

```
ldana.print_topics()
```

```
# In[42]:
```



```
# Let's start with 3 topics
```

```
ldana = models.LdaModel(corpus=corpusna, num_topics=3, id2word=id2wordna,  
passes=10)
```

```
ldana.print_topics()
```

```
# In[41]:
```

```
# Let's try 4 topics
```

```
ldana = models.LdaModel(corpus=corpusna, num_topics=4, id2word=id2wordna,  
passes=10)
```

```
ldana.print_topics()
```

```
# In[44]:
```

```
ldana = models.LdaModel(corpus=corpusna, num_topics=2, id2word=id2wordna,  
passes=80)
```

```
ldana.print_topics()
```

```
# ## Two Topics estimated
```

```
#
```

```
# ##### 1. Closure of issues, relief among community
```

```
#
```

```
# ##### 2. Party Politics and Media revolving around the case
```



```
# In[45]:
```

```
corpus_transformed = ldana[corpusna]
```

```
list(zip([a for [(a,b)] in corpus_transformed], data_dtmna.index))
```

```
# ##### Topic 1 - india today
```

```
# ##### Topic 2 - ndtv and republic
```

Results

Data Cleaning

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(stop_words='english')
data_cv = cv.fit_transform(data_clean.transcript)
data_dtm = pd.DataFrame(data_cv.toarray(), columns=cv.get_feature_names())
data_dtm.index = data_clean.index
data_dtm
```

	aapke	aaron	aarathi	ab	aberration	abided	ability	able	absent	absolutely	...	youre	youve	yupi	zafar	zakir	zameen	zameer	zfg	zoom
indiatoday	0	0	0	0	0	0	0	7	0	0	...	4	1	1	0	0	1	0	0	0
ndtv	0	0	2	0	0	0	0	4	0	3	...	5	3	0	1	1	0	2	0	0
republic	2	1	0	1	1	1	3	3	1	5	...	7	4	0	0	0	0	0	1	2

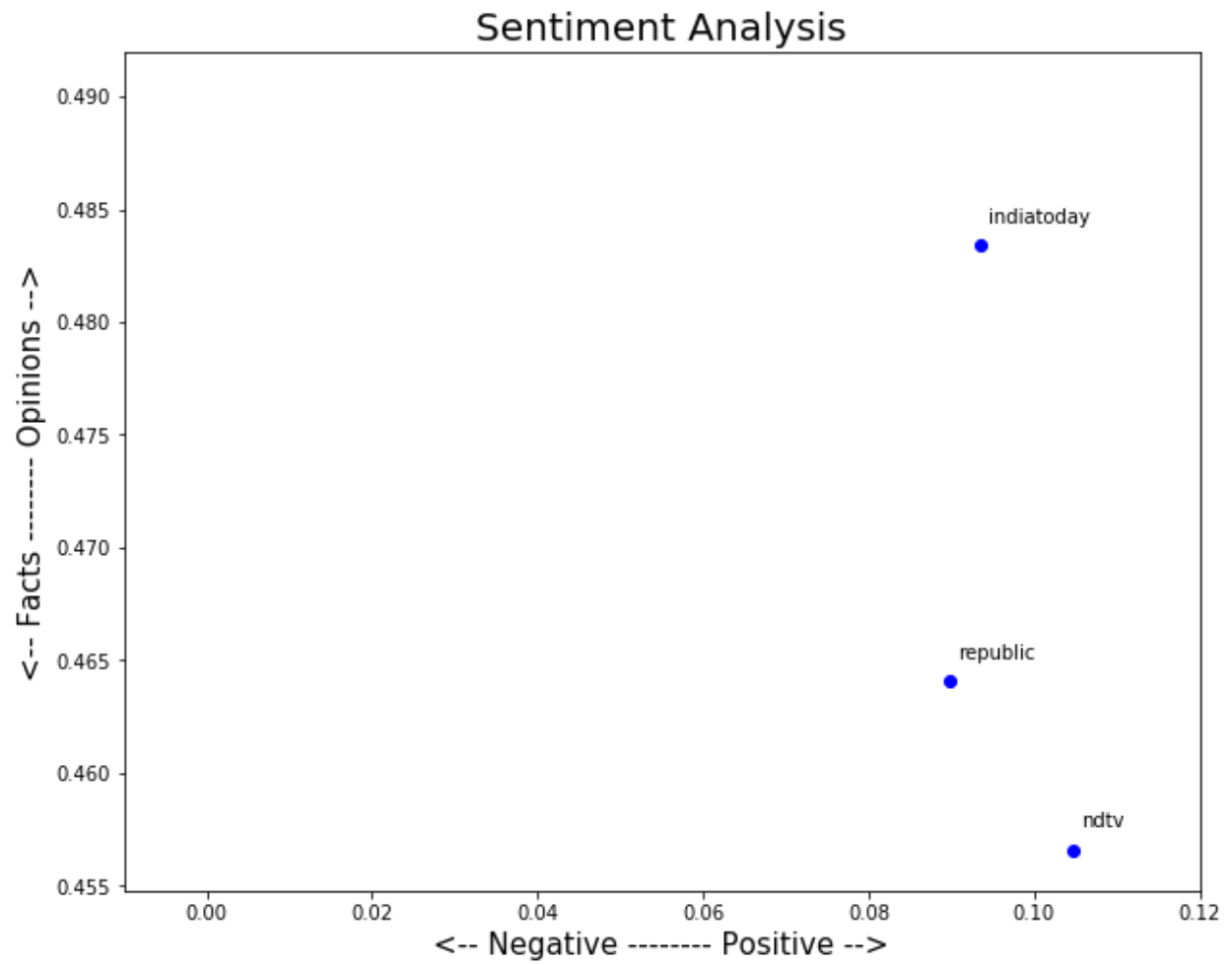
3 rows × 2072 columns

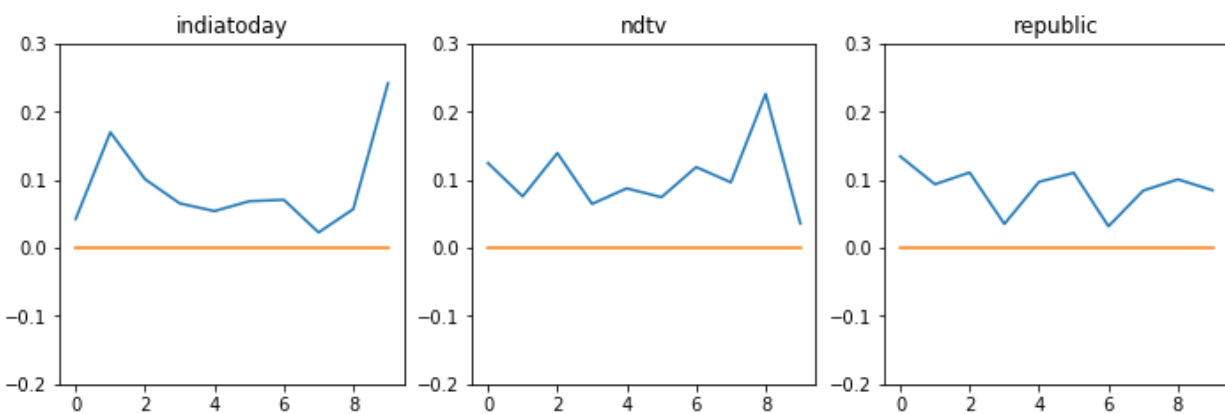
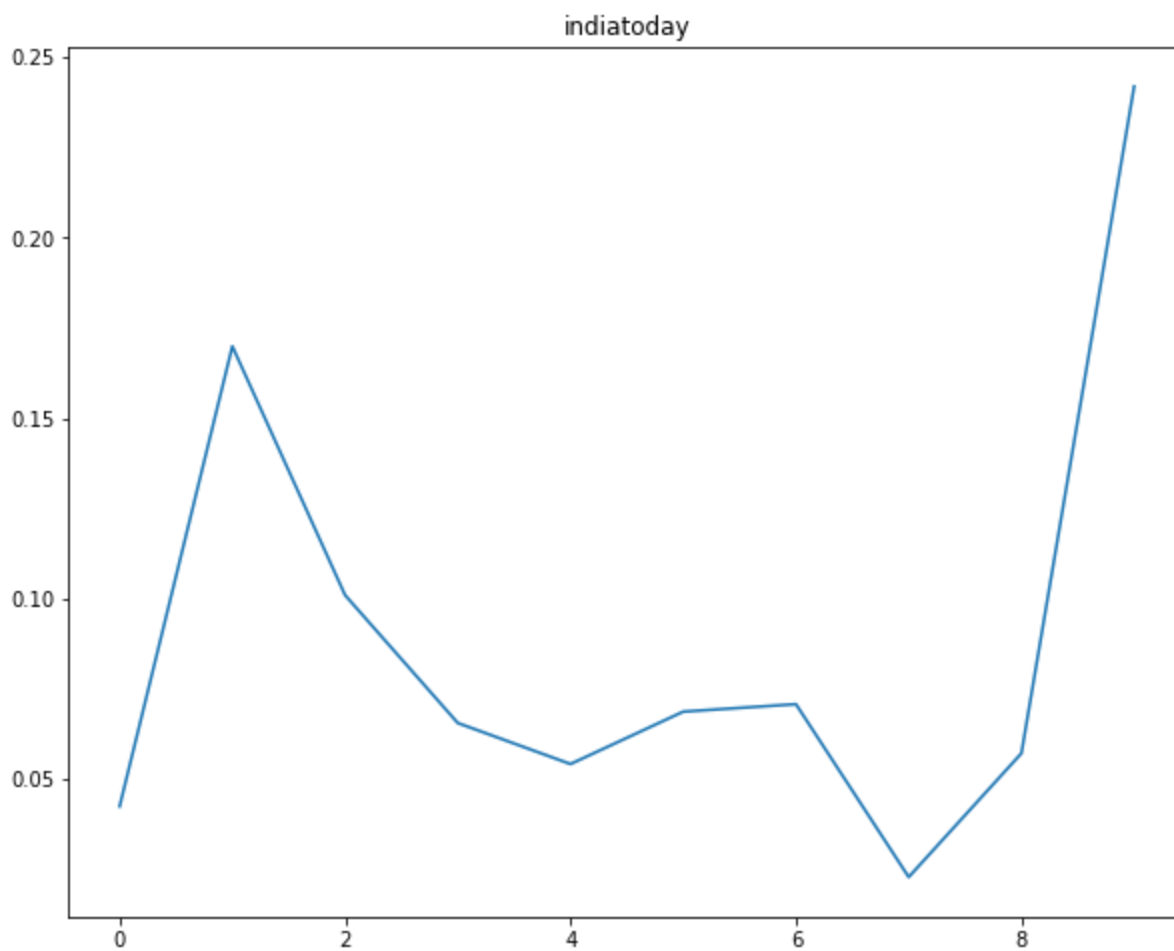
Exploratory Data Analysis



(Word cloud obtained from number of times words used in the particular speech)

Sentiment Analysis





Polarity Graph through the debate with respect to time

Topic Modelling

```
[44]: ldana = models.LdaModel(corpus=corpusna, num_topics=2, id2word=id2wordna, passes=80)
      ldana.print_topics()
```

```
[44]: [(0,
      '0.003*"closure" + 0.003*"idea" + 0.002*"issues" + 0.002*"mathura" + 0.002*"lives" + 0.002*"beginnings" + 0.001
      *"community" + 0.001*"relief" + 0.001*"feeling" + 0.001*"mp"'),
      (1,
      '0.004*"party" + 0.003*"gandhi" + 0.002*"national" + 0.002*"media" + 0.002*"debate" + 0.002*"yeah" + 0.002*"impo
      rtant" + 0.002*"board" + 0.002*"idea" + 0.002*"minute"')]
```

Two Topics estimated

1. Closure of issues, relief among community
2. Party Politics and Media revolving around the case

Conclusion

Two Topics estimated

1. Closure of issues, relief among community

2. Party Politics and Media revolving around the case

```
corpus_transformed = ldata[corpusna]
list(zip([a for [(a,b)] in corpus_transformed], data_dtmna.index))
[(0, 'indiatoday'), (1, 'ndtv'), (1, 'republic')]
```

Topic 1 - india today ¶

Topic 2 - ndtv and republic

References

- [1] Benamara, F., C. Cesarano, A. Picariello, D. Reforgiato, and V. S. Subrahmanian, "Sentiment analysis: Adjectives and adverbs are better than adjectives alone", International AAAI Conference on Weblogs and Social Media (ICWSM) (2007), 203–206.
- [2] Blei, D. M., A. Y. Ng, M. I. Jordan, and J. Lafferty, "Latent dirichlet allocation", Journal of Machine Learning Research 3 (2003), 993–1022.
- [3] Cesarano, C., A. Picariello, D. Reforgiato, and V. S. Subrahmanian, "The oasys 2.0 opinion analysis system.", International AAAI Conference on Weblogs and Social Media (ICWSM) (2007), 313–314.
- [4] Future, Recorded, "Recorded future - temporal & predictive analytics engine, media analytics & news analysis" (2010), [Online; accessed 22-November-2010].
- [5] Gerner, D. J., R. Abu-Jabr, P. A. Schrod, and . Yilmaz, "Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions", International Studies Association of Foreign Policy Interactions (2002).
- [6] Hofmann, T., "Probabilistic latent

semantic analysis”, *Uncertainty in Artificial Intelligence, UAI99*(1999), 289–296.[7]Kim, D., and A.Oh, “Topic chains for understanding a news corpus”, *12th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2011)*12(2011).[8]Leskovec, J., L.Backstrom, and J.Kleinberg, “Meme-tracking and the dynamics of the news cycle”, *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*(2009), 497–506.[9]Libby, D., “Rss 0.91 spec, revision 3”, *Netscape Communications*(1997).[10]McClelland, C., “World event/interaction survey”, *Defense Technical Information Center*(1971).[11]Minai, A., M.Perdoor, K.Byadarhaly, S.Vasa, and L.Iyer, “Asynergistic view of autonomous cognitive systems”, *Proceedings of the World Congress on Computational Intelligence*(2010).264[12]Newman, D., “Topic modeling scripts and code”, *Department of Computer Science, University of California, Irvine*(2010).[13]Oh, A., H.Lee, and Y.Kim, “User evaluation of a system for classifying and displaying political viewpoints of weblogs”, *AAAI Publications, Third International AAAI Conference on Weblogs and Social Media*(2009).[14]Palantir, “Privacy and civil liberties are in palantir’s dna” (2004), [Online; accessed 10-December-2010].[15]Proxem, “Antelope (Advanced Natural Language Object-oriented Processing Environment)” (2010), [Online; accessed 30-April-2010].[16]Tomlinson, R.G., “World event/interaction survey (weis) coding manual”, *Department of Political Science, United States Naval Academy, Annapolis, MD*.(1993