Predicting Articles Topics from Articles' Content

Melissa G Ngamini

Thinkful Data Science

December 17, 2018

Why do people want to find Article Topics?

Some of the reasons why we would want to know an article topic before reading it.

- **What do you want to know**
  People may want to read for pleasure or to educate oneself on a particular topic. Knowing the topic of an article before reading may help in figuring out if is an article we may be interested in it or know..
- **Research about the 2016 US Presidential elections**
- **Can help in differentiate the type of articles.**
  There are 2 main types of articles:
  - **News articles**
    these are designed to explain the key points first, and then flesh them out with detail. So, the most important information is presented first, with information being less and less useful as the article progresses.
  - **Opinion articles**
    these present a point of view. Here the most important information is contained in the introduction and the summary, with the middle of the article containing supporting arguments. In our dataset they are mostly Breitbart articles.

What else could the Model be used for?

Some of the informations we would get from this research.

- **Find the Most Popular Topics**
  This can help figure out what topic was the most written about. Politics: the 2016 elections, Trump the GOP and the DNC.

- **Predicting the Topic of Articles based on the content**
  This can help figure out what topic any random article can be placed in based on its content.

- **Figuring what kind of reporter they need to hire**
  Depending on the in-demand articles, maybe they need to hire more journalist that can cover particular topics or give more opinion based articles.

- **Find the right articles Titles**
  Since the article Titles are used as a way to figuring out the article contents, we can look at the content of an articles and try to figure out the best titles for say article to attract a certain type of readers

Definitions of the Topic of an Article

1. **Definition:**
   The topic of an article is the subject matter or issue a particular news article title is about.

2. **Method of Differentiation of Topic:**
   Latent semantic analysis (LSA) is a method of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms

3. **Interpretation of Topics from LSA:**
   The topic of an **LSA Component** *is inferred by looking at the set of 10 words that are used the most together in news article titles in that component and cross checked against the occurrence of those words across all components.*

Plan of Presentation

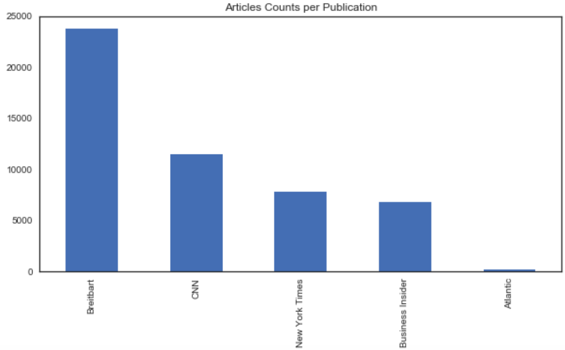
1. **Data Set Exploration**
    - Sampling
    - Analysis
    - Cleaning
2. **Features Creation**
    - Create Vectorizer for Articles Contents
    - Create Vectorizer for Articles Titles
3. **Cluster of Topics using LSA**
    - 10 Main Topics on Article Titles
    - 3 Main Topics on Article Titles
4. **Models used for Prediction**
    - Keras
    - Random Forest Classifier
    - Stochastic Gradient Descent
5. **Prediction of Articles Topics**
    - Prediction using Stochastic Gradient Descent for 10 Topics model
    - Prediction using Stochastic Gradient Descent for 3 Topics model

Original Data Set

```
df = pd.read_csv('articles1.csv')
print('The shape of the data in articles 1 is:', df.shape)
```

```
The shape of the data in articles 1 is: (50000, 10)
```

(a) Size of Data



(b) Article Count Per Publications

Figure: Data Set Pre-Sampling

Sampling of Data Set

```
#Remove article to use for prediction
df_new = df.sample(n=1, replace=False, axis = 0, random_state=20)
rem = df_new.index
X_new = df['content'][rem[0]]
```

(a) Select Article for Example Prediction

```
df.drop(rem, axis=0, inplace = True)
df1 = df.sample(frac=0.2, replace=False, axis = 0, random_state=20)
```

(b) Select 20% of Original Data Set

Figure: Sampled Data Set

Features in data Set

```
print('The shape of the data in articles 1 is:', df1.shape)
display(df1.head())
```

```
The shape of the data in articles 1 is: (10000, 10)
```

	Unnamed: 0	id	title	publication	author	date	year	month	url	content
2082	2082	19617	Raiders, Mosul, Jared Kushner: Your Monday Eve...	New York Times	Karen Zraick and Sandra Stevenson	2017-03-28	2017.0	3.0	NaN	(Want to get this briefing by email? Here's th...
1206	1206	18647	'Tone Down Your Gayness': St. Louis Police Off...	New York Times	Christine Hauser	2017-02-18	2017.0	2.0	NaN	An police sergeant in Missouri has filed a d...
6635	6635	24964	Abortion Pill Orders Rise in 7 Latin American ...	New York Times	Donald G. McNeil Jr. and Pam Belluck	2016-06-23	2016.0	6.0	NaN	Orders for abortion pills by women in seven La...
23920	23924	42675	EXCLUSIVE: After Brussels, Islamic State Suppo...	Breitbart	Aaron Klein and Ali Waked	2016-03-22	2016.0	3.0	NaN	TEL AVIV — Jubilant Islamic State sympathiz...
29456	29464	48228	Murder of American Student Found in Tiber Rive...	Breitbart	Thomas D. Williams, Ph.D.	2016-07-07	2016.0	7.0	NaN	The lifeless body of a American college stu...

Figure: Features in Data

Analysis of data Set

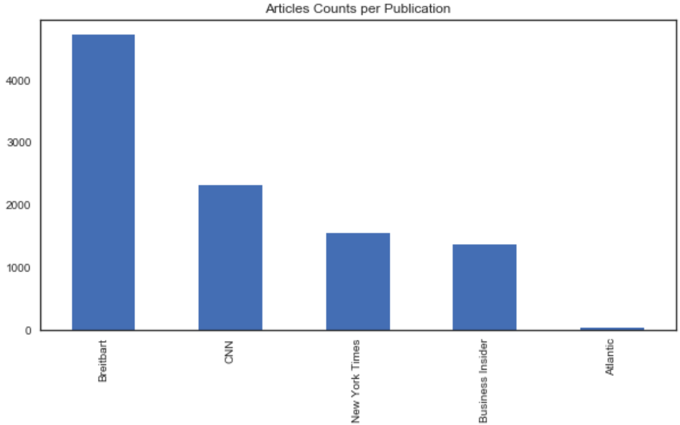


Figure: Articles Distribution in Data per Publications

Analysis of data Set

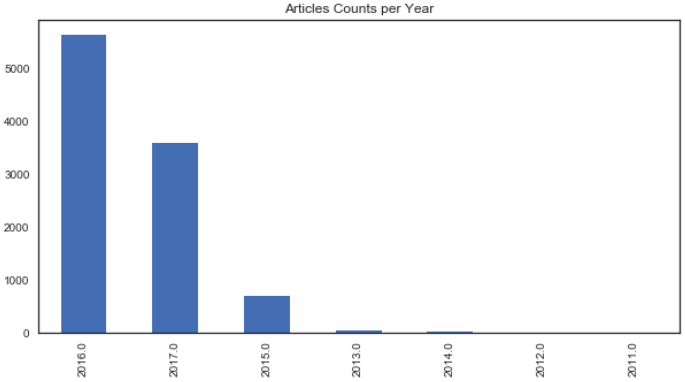


Figure: Articles Distribution in Data Per Year

Distribution of Data per Month

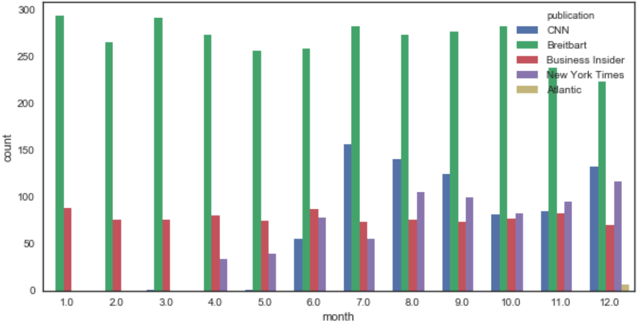


Figure: Articles Publications Counts per Publication in 2016

Distribution of Data per Month

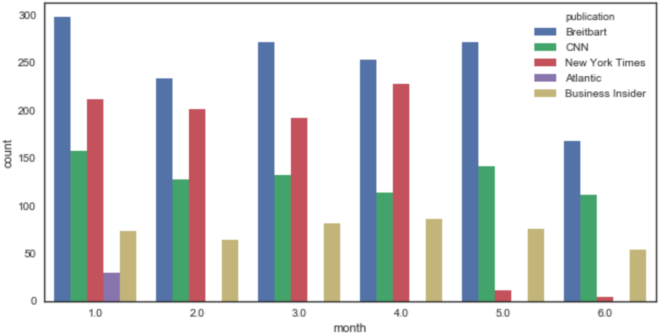


Figure: Articles Publications Counts per Publication in 2017

Test and Training Test Set Splits

```
Y = df['title']
X = df[['publication','content']]
```

```
from sklearn.model_selection import train_test_split

#  Create Training and Test Sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=20)
```

Figure: Features in Test and Training Sets

Vectorizer Using tf-idf on Articles contents

```
print("Extracting features from the titles of articles in dataset using a vectorizer")
t0 = time.clock()
Yvectorizer = TfidfVectorizer(min_df=2, # only use words that appear at least twice
                              stop_words='english',
                              use_idf=False,#we definitely want to use inverse document frequencies in our
                              norm=u'l2', #Applies a correction factor so that longer paragraphs and short
                              smooth_idf=True, #Adds 1 to all document frequencies, as if an extra documen
                              vocabulary=vocab,
                              ngram_range=(1, 3)
                              )

#Applying the vectorizer to Y_train and Y_test

Y_train_tfidf=Yvectorizer.fit_transform(Y_train)
Y_test_tfidf=Yvectorizer.transform(Y_test)
print('\nYvectorizer on articles titles done in '+'%s seconds'% (time.clock() - t0))

print('\nThe shape of Y_train_tfidf for articles titles is:', Y_train_tfidf.shape)
print('\nThe shape of Y_test_tfidf for articles titles is:', Y_test_tfidf.shape)
```

Figure: Vectorizer on Articles contents

Extracting features from the contents of articles in dataset using a vectorizer

- Xvectorizer on articles contents in dataset done in 51.618661 seconds
- The shape of $X_{train-tfidf}$ for articles contents is: (7500, 341635)
- The shape of $X_{test-tfidf}$ for articles content is: (2500, 341635)

Vectorizer Using tf-idf on Articles titles

```
print("Extracting features from the contents of articles in dataset using a vectorizer")
t0 = time.clock()
Xvectorizer = TfidfVectorizer(max_df=.5, # drop words that occur in more than half the paragraphs
                              min_df=2, # only use words that appear at least twice
                              stop_words='english',
                              use_idf=True,#we definitely want to use inverse document frequencies in our w
                              norm=u'l2', #Applies a correction factor so that longer paragraphs and shorte
                              smooth_idf=True, #Adds 1 to all document frequencies, as if an extra document
                              ngram_range=(1, 3)
                              )

#Find Vocab words on the whole articles
#Applying the vectorizer to X_train and X_test
X_train_tfidf=Xvectorizer.fit_transform(X_train)
X_test_tfidf=Xvectorizer.transform(X_test)
vocab = Xvectorizer.vocabulary_

print('\nXvectorizer on articles contents in dataset done in '+'%s seconds'% (time.clock() - t0))


print('\nThe shape of X_train_tfidf for articles contents is:', X_train_tfidf.shape)
print('\nThe shape of X_test_tfidf for articles content is:', X_test_tfidf.shape)
```

Figure: Vectorizer on Articles titles

Extracting features from the titles of articles in dataset using a vectorizer

- Yvectorizer on articles titles done in 0.782976000000005 seconds
- The shape of $Y_{train-tfidf}$ for articles titles is: (7500, 341635)
- The shape of $Y_{test-tfidf}$ for articles titles is: (2500, 341635)

LSA to find 10 Main Topics

```
#Our SVD data reducer.  We are going to reduce the feature space from 84939 to 1000.
t0 = time.clock()
svd= TruncatedSVD(10, random_state = 20)
lsa = make_pipeline(svd, Normalizer(copy=False))


# Run SVD on the training data, then project the training data.
Y_train_lsa10 = lsa.fit_transform(Y_train_tfidf)
Y_test_lsa10 = lsa.transform(Y_test_tfidf)

print('LSA for 10 articles done in '+'%s seconds'% (time.clock() - t0))

variance_explained=svd.explained_variance_ratio_
total_variance = variance_explained.sum()
print('\nPercent variance captured by all components: ', (total_variance*100))
```

Figure: Select 10 Topics

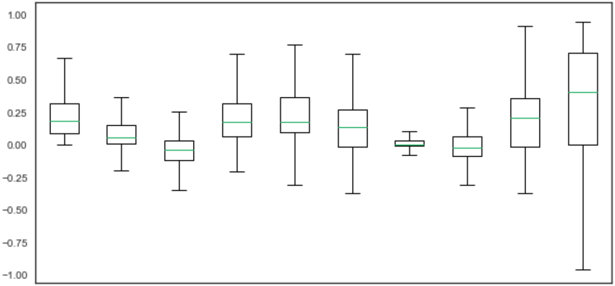
1. LSA for 10 articles done in 2.8697930000000014 seconds
2. Percent variance captured by all components: 5.32%
3. The shape of $Y_{train-lsa}$ for titles is: (7500, 10)
4. The shape of $Y_{test-lsa}$ for titles is: (2500, 10)

Clustering Articles using the 10 Main Topics

The Components Value per Articles are:

	0	1	2	3	4	5	6	7	8	9
27396	0.851600	0.342187	-0.362270	-0.049839	0.073373	-0.080093	-0.040044	-0.004193	-0.086891	-0.054701
49747	0.104038	0.165357	0.014689	0.418140	0.151705	0.347540	0.298861	-0.194855	0.421914	0.581102
26513	0.224825	0.342521	0.110496	0.108915	0.444641	0.171999	0.630741	-0.067130	0.311052	0.285793
19619	0.173143	0.265229	-0.100522	0.310068	0.388326	0.198040	-0.006450	0.142994	0.213670	0.732986
1073	0.301828	0.084896	0.032707	0.506335	0.265323	0.309568	0.055576	-0.170560	0.148380	0.651017

(a) Component Value per Articles



(b) Component Value Distribution

Interpretation of 10 Main Topics

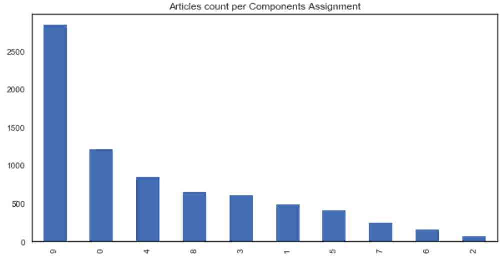


Figure: Article Count Per Topics

- Topic 0: **Donald Trump Campaign**
- Topic 1: **DNC Campaign**
- Topic 2: **Donald Trump's Win**
- Topic 3: **New York Related News**
- Topic 4: **Trump's America First Policies**
- Topic 5: **The Obama Administration**
- Topic 6: **US Supreme Court**
- Topic 7: **White House and Health Care**
- Topic 8: **Ted Cruz Primary Campaign**
- Topic 9: **Fear Mongering against Immigrant**

Topic 9: Fear Mongering against Immigrant

10 Most Recurring words

["attack', 'border', 'man', 'milo', 'news', 'police', 'report', 'says', 'state', 'texas']

- Police mentioned 135 times in component 9
- Attack mentioned 121 times in component 9
- Border mentioned 76 times in component 9

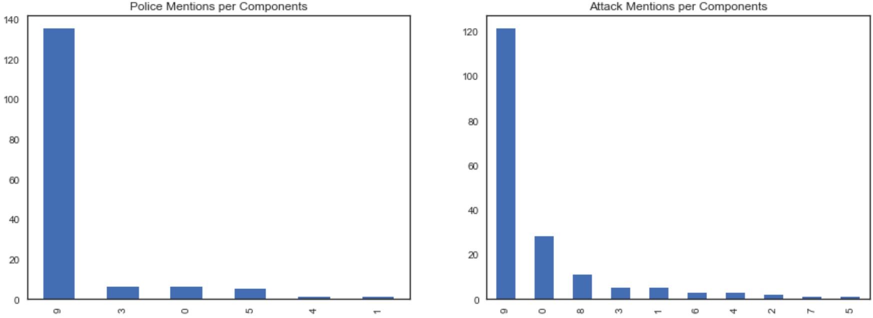


Figure: Top Words in Component 9 for 10 Topics Model

Topic 0: Donald Trump Campaign

10 Most Recurring words

['campaign', 'clinton', 'cruz', 'donald', 'hillary', 'obama', 'poll', 'president', 'says', 'trump']

- Trump mentioned 1149 times in Component 0
- Donald mentioned 307 times in component 0

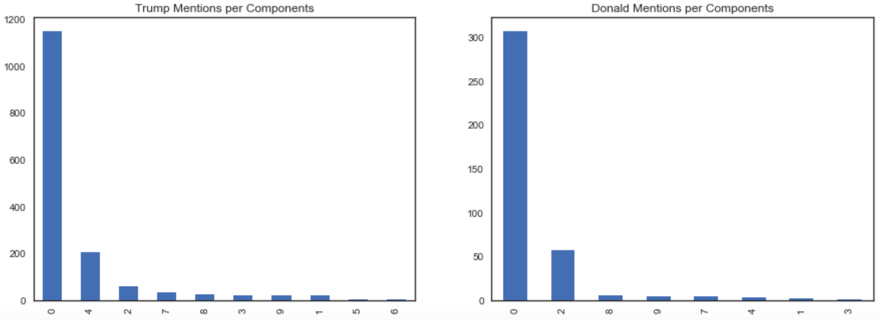


Figure: Top Words in Component 0 for 10 Topics Model

Topic 4: Trump’s America First Policies

10 Most Recurring words

['america', 'ban', 'care', 'immigration', 'plan', 'president', 'russia', 'speech', 'trumps', 'wall']

- Trumps mentioned 195 times in Component 4
- America mentioned 52 times in component 4

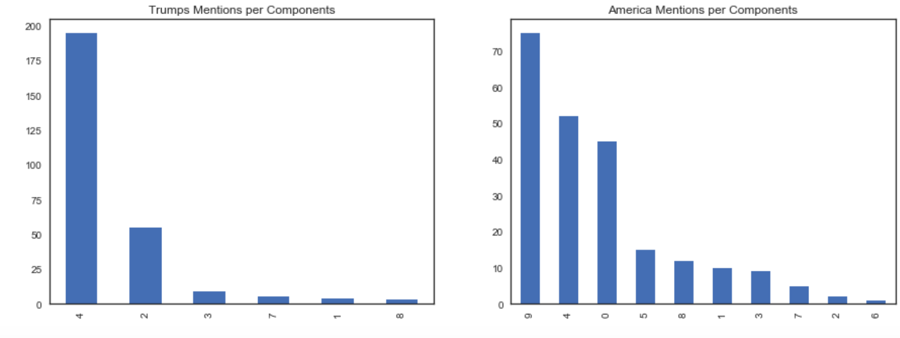


Figure: Top Words in Component 4 for 10 Topics Model

LSA to find 3 Main Topics

```python
#Our SVD data reducer.  We are going to reduce the feature space from 84939 to 1000.
t0 = time.clock()
svd= TruncatedSVD(3)
lsa = make_pipeline(svd, Normalizer(copy=False))


# Run SVD on the training data, then project the training data.
Y_train_lsa3 = lsa.fit_transform(Y_train_tfidf)
Y_test_lsa3 = lsa.transform(Y_test_tfidf)

print('LSA for 3 articles done in '+'%s seconds'% (time.clock() - t0))

variance_explained=svd.explained_variance_ratio_
total_variance = variance_explained.sum()
print('\nPercent variance captured by all components: ', (total_variance*100))
```

Figure: Select 3 Topics

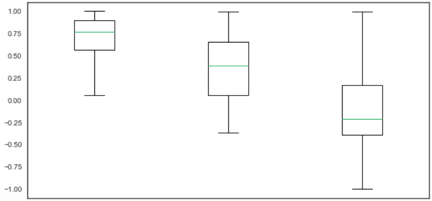
1. LSA for 3 articles done in 1.7483990000000063 seconds
2. Percent variance captured by all components: 2.9094998026860037
3. The shape of $Y_{train-lsa}$ for titles is: (7500, 3)
4. The shape of $Y_{test-lsa}$ for titles is: (2500, 3)

Clustering Articles using the 3 Main Topics

```
The Components Value per Articles are:
```

	0	1	2
27396	0.862915	0.346258	-0.368081
49747	0.531014	0.847255	0.013515
26513	0.532046	0.809791	0.247317
19619	0.517324	0.794489	-0.318063
1073	0.960454	0.267125	0.078557

(a) Component Value per Articles



(b) Component Value Distribution

Interpretation of 3 Main Topics

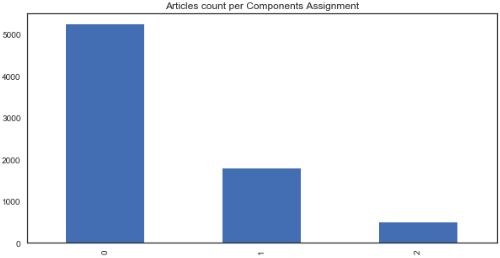


Figure: Article Count Per Topics

- Topic 0: **Trump During and after Elections**
- Topic 1: **Hillary Clinton During and after Elections**
- Topic 2: **Trumps and Miscellaneous topics**

Topic 0: Trump During and after Elections

10 Most Recurring words

['cruz', 'donald', 'gop', 'house', 'obama', 'police', 'report', 'says', 'trump', 'white']

- Trump mentioned 1232 times in Component 0
- Donald mentioned 318 times in component 0

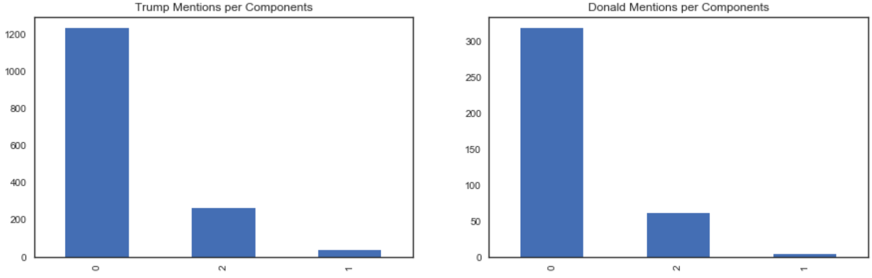


Figure: Top Words in Component 0 for 3 Topics Model

Topic 1: Hillary Clinton During and after Elections

10 Most Recurring words

['campaign', 'clinton', 'facts', 'fast', 'hillary', 'new', 'sanders', 'state', 'years', 'york']

- Clinton mentioned 314 times in Component 1
- Hillary mentioned 239 times in component 1

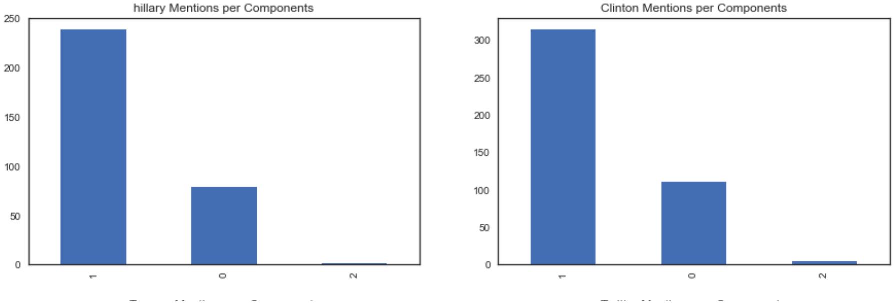


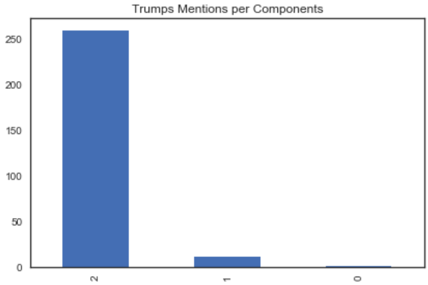
Figure: Top Words in Component 1 for 3 Topics Model

Topic 2: Trumps and Miscellaneous topics

10 Most Recurring words

['ban', 'donald', 'election', 'facebook', 'immigration', 'new', 'speech', 'trumps', 'twitter', 'wall']

- trumps mentioned 259 times in Component 2



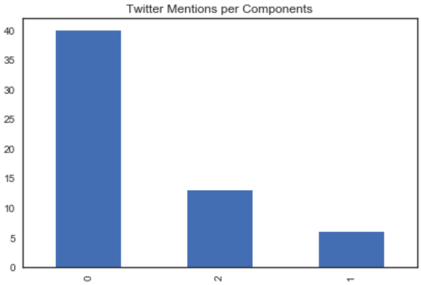


Figure: Top Words in Component 4 for 3 Topics Model

Modeling for Tensor Flow and Keras

```
# Building the Model
model = Sequential()
# First convolutional layer, note the specification of shape
model.add(Convolution1D(filters=nb_filter,kernel_size=kernel_size,
                 activation='relu',
                 input_shape=newshape))
model.add(Dropout(0.15))

model.add(MaxPooling1D())
model.add(Dropout(0.10))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(nb_outputs, activation='softmax'))

#model.compile(loss='mse', optimizer='adam', metrics=['mae'])
#model.compile(loss=keras.losses.categorical_crossentropy, optimizer='sgd', metrics=['accuracy'])
model.compile(loss=keras.losses.categorical_hinge, optimizer='sgd', metrics=['accuracy'])
```

(a) Keras Model

- kernel-size = 3
- nb-filter = 64
- batch-size =128

- epochs = 5
- verbose = 1

- For n = 3 Best Results
  - Test loss: 0.9968
  - Test accuracy: 0.4748

Results using features from LSA

```
confusion_matrix(Y_test_tf1.argmax(axis=1), ypred_10.argmax(axis=1))
```

```
array([[  0,   0,   0,   1,   0, 472,   2,   0,   0,   0],
       [  0,   0,   0,   0,   0, 173,   0,   0,   0,   0],
       [  0,   0,   0,   1,   0, 162,   0,   0,   0,   0],
       [  0,   0,   0,   2,   1, 581,   0,   1,   0,   0],
       [  0,   0,   0,   0,   0,  15,   0,   0,   0,   0],
       [  0,   0,   0,   2,   0, 871,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  47,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  42,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  63,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  64,   0,   0,   0,   0]])
```

(b) n = 10

```
confusion_matrix(Y_test_tf3.argmax(axis=1), ypred_3.argmax(axis=1))
```

```
array([[1274,    0,    0],
       [ 424,   11,    2],
       [ 785,    0,    4]])
```

(c) n = 3

Figure: Results for modeling with Keras

RFC score using with LSA = 10 Features

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0	247	83	0	5	3	10	3	26	21	31
Actual 1	7	126	0	2	0	2	1	1	1	17
Actual 2	16	2	0	0	0	0	0	2	1	3
Actual 3	14	7	0	58	1	6	4	7	2	129
Actual 4	96	26	0	18	4	11	10	28	6	98
Actual 5	12	12	0	14	0	42	1	6	0	47
Actual 6	4	1	0	3	0	0	17	0	1	12
Actual 7	11	2	0	2	1	2	1	39	0	28
Actual 8	20	19	0	12	2	5	6	4	31	98
Actual 9	65	23	1	88	5	28	26	18	12	644

(a) max-depth=10, n-estimators=1000, class-weight ="balanced

Figure: Results for models using LSA = 10 Features on Test Set

hyper parameters: max-depth=10, max-features='auto', n-estimators=1000, class-weight ="balanced"

- Runtime for Random Forest: 21.09 seconds
- Score on Training Set: 0.64
- Score on Test Set: 0.48
- Cross validation results: 49.907% $\pm$ 0.762%

Modeling for Stochastic Gradient Descent

```
start_time = time.clock()
sgdc.fit(X_train_tfidf, Y_train_component10)
print('Runtime for Stochastic Gradient: '+'%s seconds'% (time.clock() - start_time)) # End time

print('Training set accuracy:', sgdc.score(X_train_tfidf, Y_train_component10))
print('\nTest set accuracy:', sgdc.score(X_test_tfidf, Y_test_component10))


cv_train = cross_val_score(sgdc, X_train_tfidf, Y_train_component10, cv=5,scoring ='f1_weighted'

### Put this with the Cros validation score.
plusminus = u"\u00B1"

print('\nCross validation results: {:.3%} {} {:.3%} \n \n {}'.format(cv_train.mean(), plusminus,
```

(a) Stochastic Gradient Descent

Figure: Model Using SGD

Stochastic Gradient Descent with LSA = 10 Features

Comparing SGDC data against the test set data:

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0	331	16	0	2	9	2	0	0	9	60
Actual 1	21	96	0	0	2	1	0	0	1	36
Actual 2	18	0	0	0	1	0	0	0	0	5
Actual 3	9	1	0	5	5	1	0	0	4	203
Actual 4	79	7	0	1	27	1	0	7	4	171
Actual 5	11	3	0	1	4	28	0	0	0	87
Actual 6	1	0	0	0	0	0	0	0	1	36
Actual 7	20	0	0	0	2	3	0	15	1	45
Actual 8	20	5	0	1	9	3	0	0	23	136
Actual 9	36	4	0	2	3	5	0	2	9	849

(a) loss = 'log', class-weight=None

```
Y_test_component10.value_counts()
9    910
0    429
4    297
3    228
8    197
1    157
5    134
7     86
6     38
2     24
Name: component, dtype: int64
```

(b) Topic Distribution

Figure: Results for models using Stochastic Gradient Descent with LSA = 10

loss = 'log', penalty = 'l2', alpha=0.0001, class-weight=None, fit-intercept=True

- Runtime for Stochastic Gradient: 2.047241000000213 seconds
- Training set score: 0.7038666666666666
- Test set score: 0.5496
- Cross validation results: 55.561% $\pm$ 1.222%
- Cross validation results with f1: 47.065% $\pm$ 1.525%

Stochastic Gradient Descent with LSA = 10 Features

```
Comparing SGDC data against the test set data:
```

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0	313	25	3	9	15	8	1	6	20	29
Actual 1	13	109	0	8	5	4	1	0	2	15
Actual 2	16	0	2	1	1	1	0	0	1	2
Actual 3	5	2	0	70	16	6	5	5	5	114
Actual 4	64	11	1	18	48	12	6	19	15	103
Actual 5	8	3	0	10	6	47	1	4	7	48
Actual 6	2	0	0	5	1	0	16	0	1	13
Actual 7	13	0	0	2	7	5	0	31	4	24
Actual 8	15	5	0	8	11	7	3	2	55	91
Actual 9	38	8	1	68	28	18	5	12	30	702

(a) loss = 'log', class-weight='balanced'

```
Y_test_component10.value_counts()
```

```
9    910
0    429
4    297
3    228
8    197
1    157
5    134
7     86
6     38
2     24
Name: component, dtype: int64
```

(b) Topic Distribution

Figure: Results for models using Stochastic Gradient Descent with LSA = 10

loss = 'log', penalty='l2', alpha=0.0001, class-weight='balanced',fit-intercept= True

- Runtime for Stochastic Gradient: 2.15 seconds
- Training set accuracy score: 0.8681333333333333
- Test set accuracy score: 0.5572
- Cross validation accuracy: 56.254% $\pm$ 2.843%
- Cross validation f1-weighted: 53.231% $\pm$ 2.631%

Results using Stochastic Gradient Descent with LSA = 10 Features

Comparing SGDC data against the test set data:

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	Predicted 5	Predicted 6	Predicted 7	Predicted 8	Predicted 9
Actual 0	294	22	0	10	25	12	0	5	25	36
Actual 1	11	108	0	3	4	7	1	1	4	18
Actual 2	17	0	0	2	2	1	0	1	0	1
Actual 3	8	2	0	54	17	8	4	3	12	120
Actual 4	56	10	0	21	62	11	0	16	23	98
Actual 5	8	4	0	9	11	46	0	2	13	41
Actual 6	3	0	0	4	1	0	13	0	5	12
Actual 7	15	2	0	2	5	6	0	24	6	26
Actual 8	19	6	0	9	11	6	1	0	59	86
Actual 9	37	8	1	51	57	18	4	9	38	687

(a) loss = 'modified-huber', class-weight='balanced'

```
Y_test_component10.value_counts()
9    910
0    429
4    297
3    228
8    197
1    157
5    134
7     86
6     38
2     24
Name: component, dtype: int64
```

(b) Topic Distribution

Figure: Results for models using Stochastic Gradient Descent with LSA = 10

loss='modified-huber', penalty='l2', alpha=0.0001, class-weight='balanced',fit-intercept= True

- Runtime for Stochastic Gradient: 1.52 seconds
- Training set accuracy score: 0.9974666666666666
- Test set accuracy score: 0.5388
- Cross validation accuracy: 55.014% $\pm$ 2.270%
- Cross validation f1-weighted: 52.173% $\pm$ 3.441%

Results using Stochastic Gradient Descent with LSA = 3 Features

Comparing SGDC data against the test set data:

	Predicted 0	Predicted 1	Predicted 2
Actual 0	1606	115	22
Actual 1	424	196	1
Actual 2	117	3	16

(a) loss = 'log', class-weight='balanced'

```
Y_test_component3.value_counts()

0    1743
1     615
2     142
Name: component, dtype: int64
```

(b) Topic Distribution

Figure: Results for models using Stochastic Gradient Descent with LSA = 3

loss='log', penalty='l2', alpha=0.0001, class-weight='balanced',fit-intercept= True

- Runtime for Stochastic Gradient: 0.85 seconds
- Training set accuracy score: 0.8
- Test set accuracy score: 0.7316
- Cross validation accuracy: 73.120% ± 0.941%
- Cross validation f1-weighted: 65.894% ± 1.300%

Article which Topic we want to predict

```
print(X_new[rem[0]])
```

```
berlin reuters     tens of thousands of people protested in european cities on saturday against planned f
ree trade deals with the united states and canada they say would undermine democracy and lower food safet
y environmental and labour standards organisers     an alliance of environmental groups labour unions and
opposition parties     said 320 000 people took part in rallies in seven german cities including berlin h
amburg munich and frankfurt police put the figure at around 180 000  smaller protests were also planned i
n other european cities including vienna and salzburg in austria and gothenburg and stockholm in sweden i
n berlin demonstrators waved banners reading stopp ceta     stopp ttip another placard said people over p
rofits the demonstrations are against the transatlantic trade and investment partnership ttip with the un
ited states and the comprehensive economic trade agreement ceta with canada currently being negotiated by
the european unions executive with the respective governments across the atlantic opposition in europe to
the trade deals has risen over the past year with critics saying the pacts would hand too much power to b
ig multinationals at the expense of consumers and workers by establishing arbitration courts to settle di
sputes between companies and governments horror stories eu trade commissioner cecilia malmstrom defended
the planned trade deals and accused the opponents of deliberately heating up the debate with horror stori
es and lies the idea that ttip will lower environmental standards is simply not true malmstrom told germa
n daily bild also the assertion that well be flooded with genetically modified food is simply wrong our d
emocracy of course wont be undermined as some seem to believe  malmstrom said german exporters would bene
fit highly from the deals because they would reduce   barriers to trade this helps germany and creates jo
bs she added german economy minister sigmar gabriel who faces crunch ceta vote on monday by his social de
mocrats spd said that the trade agreements were europes best chance to shape globalisation so that it ser
```

Figure: New Article Content

10 Topics Model

- Topic 0: **Donald Trump Campaign**
- Topic 1: **DNC Campaign**
- Topic 2: **Donald Trump's Win**
- Topic 3: **New York Related News**
- Topic 4: **Trump's America First Policies**
- Topic 5: **The Obama Administration**
- Topic 6: **US Supreme Court**
- Topic 7: **White House and Health Care**
- Topic 8: **Ted Cruz Primary Campaign**
- Topic 9: **Fear Mongering against Immigrant**

3 Topics Model

- Topic 0: **Trump During and after Elections**
- Topic 1: **Hillary Clinton During and after Elections**
- Topic 2: **Trumps and Miscellaneous topics**

Prediction Probabilities

hyper parameters: loss = 'log', penalty='l2', alpha=0.0001, class-weight='balanced',fit-intercept= True

1. **LSA with 10 features:**

```
#ynew1_pro_10 = sgdc.predict(X_new_tfidf)
ynew1_pro_10 = sgdc.predict_proba(X_new_tfidf)
ynew1_pro_10
```

```
array([[0.07846364, 0.02296198, 0.01151345, 0.05158394, 0.13974375,
        0.10639698, 0.01875583, 0.01749489, 0.14361437, 0.40947116]])
```

(a) n = 10

Figure: Prediction using LSA = 10 features

2. **LSA with 3 features:**

```
#ynew1_pro_3 = sgdc.predict(X_new_tfidf)
ynew1_pro_3 = sgdc.predict_proba(X_new_tfidf)
ynew1_pro_3
```

```
array([[0.85473453, 0.09295288, 0.05231259]])
```

(a) n = 3

Figure: Prediction LSA = 3 features

Conclusion and Future Work

1. Use 2 other data sets that had a different Publications distribution than the one I used for my Training Set and Test Set.
2. Get a better Computing Engine because with the computer I have I could only with a small of the data and running small amount of epochs
3. Use Data from the other 2 Data set to improve Test and Training Sets for my Keras Model.
4. Use Data from other 2 Data sets to predict topic of articles on the SGD Model.

That's all folks!

Questions?

Find Classes for Different Clusters

```
# Convert class vectors to binary class matrices
nb_classes = 10
print(nb_classes, 'classes')
```

```
10 classes
```

```
Y_train_tf1 = keras.utils.to_categorical(Y_train_component10, nb_classes)
Y_test_tf1 = keras.utils.to_categorical(Y_test_component10, nb_classes)


print('Y_train shape:', Y_train_tf1.shape)
print('Y_test shape:', Y_test_tf1.shape)
```

(a) 10 Classes

```
# Convert class vectors to binary class matrices
nb_classes2 = 3
print(nb_classes2, 'classes')
```

```
3 classes
```

```
Y_train_tf3 = keras.utils.to_categorical(Y_train_component3, nb_classes2)
Y_test_tf3 = keras.utils.to_categorical(Y_test_component3, nb_classes2)


print('Y_train shape:', Y_train_tf3.shape)
print('Y_test shape:', Y_test_tf3.shape)
```

(b) 3 Classes

Procedure to Reshape Data for Keras Model

```
nb_filter = 64                           ##Always 2^x features
nb_outputs = Y_train_tf1.shape[1]

kernel_size = 3
nb_samples = X_train_tfidf.shape[0]
nb_features = X_train_tfidf.shape[1]
newshape = (nb_features,1)
```

```
### Transform Sparse matrix into array
X1 = X_train_tfidf.toarray()
X2 = X_test_tfidf.toarray()
```

```
# reshape Train data
X_train_r = np.zeros((X_train_tfidf.shape[0], nb_features, 1))
X_train_r[:, :, 0] = X1[:,:]
```

```
# reshape Test data
X_test_r = np.zeros((X_test_tfidf.shape[0], nb_features, 1))
X_test_r[:, :, 0] = X2[:,:]
```

(c) Reshaping of Data

RFC score using features from LSA = 3

	Predicted 0	Predicted 1	Predicted 2
Actual 0	1196	531	16
Actual 1	234	386	1
Actual 2	107	18	11

(d) max-depth=10, n-estimators=1000, class-weight ="balanced

Figure: Results for models using LSA = 3 Features on Test Set

hyper parameters: max-depth=10, max-features='auto', n-estimators=1000, class-weight ="balanced"

- Runtime for Random Forest: 20.98 seconds
- Score on Training Set: 0.83
- Score on Test Set: 0.64
- Cross validation results: 50.454% $\pm$ 0.336%