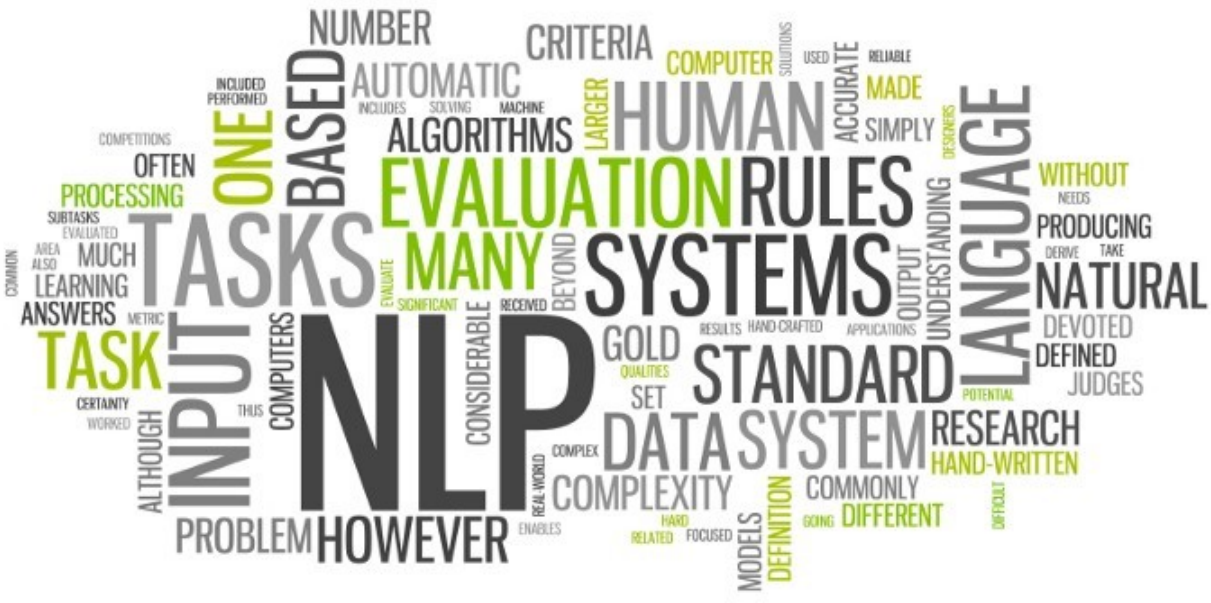
Exploratory Data Analysis for Text Data

yonatan hadar Follow
Mar 17, 2020 · 7 min read ★

Last month I started my new data science job at BigPanda and after a few days of installations, lectures and meeting new people I finally got access to the data. As an educated data scientist that always works according to CRISP-DM, I wanted to start my project with an exploratory data analysis (EDA). I usually do EDA on tabular data by using histograms, bar plots, a correlation matrix, scatter plots and other tools but none of these tools were suitable for textual data — which is the main data type at BigPanda.

In this blog post, I'm going to review several methods I used to get to know the textual data in the data exploration phase with code examples.

I will use a dataset of Donald Trump tweets from Kaggle for demonstration.



The first thing I do in every new data exploration is looking at the data. I want to feel out some basic questions about the text: Do I need a special pre-processing method? Are there any special characteristics? If there are labels, can I understand the label from the text or do I need more knowledge on the domain? Can I already spot special patterns in the data? Is there a connection to other variables in the data?

In the Trump dataset, we can see some Twitter specifics (ie: hashtags and usernames) and many URLs that we probably want to remove.

The Analysis

Now, on to the good stuff!

We'll start the analysis with very simple text data statistics like text length and word frequencies. After that, we'll investigate more sophisticated text

characteristics using unknown words and feature engineering. Finally, we'll use a machine learning-based topic modeling method to create semantic features from the text and assess the text in the context of time series analysis.

Basic statistics on the text

Text length

The length of the samples in the dataset is very important, as it can affect how you represent your text as features for the ML models. For example, TF-IDF is usually too sparse for short texts and average Word2Vec is usually too noisy for long texts.

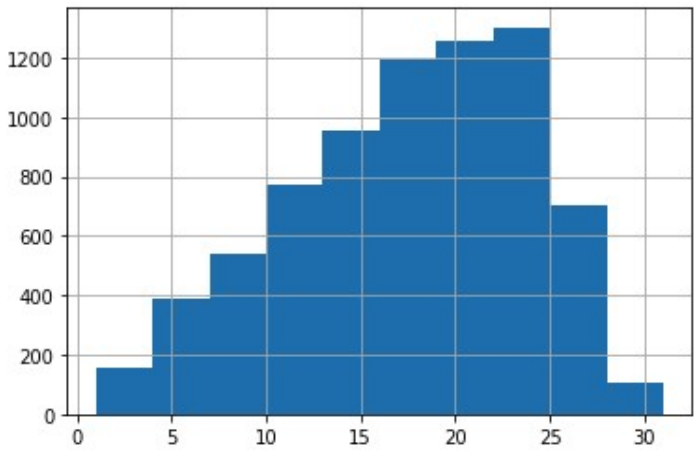
The length can also affect the algorithm you use. For example, LSTM networks are much better than vanilla RNN networks on long texts.

We calculate the number of words in each tweet and look at the length distribution.

```
lens = data.Tweet_Text.str.split().apply(lambda x: len(x))

print(lens.describe())
lens.hist()
```

```
count    7375.000000
mean       16.980746
std         6.330227
min         1.000000
25%        12.000000
50%        18.000000
75%        22.000000
max        31.000000
Name: Tweet_Text, dtype: float64
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x11d6da9d0>
```



Results of this calculation establish that text length is short (average length of 17 words) but not very short (less than 10% has less than 5 words).

Term frequencies

In this step, I find the most frequent words in the data, extracting information about its content and topics.

When doing your analysis, remember to add domain-specific stop words, not just the known English stop words. I remove stop words from this particular analysis because they appear often but are not very informative.

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
stops =  set(stopwords.words('english')+['com'])

co = CountVectorizer(stop_words=stops)
```

```
counts = co.fit_transform(data.Tweet_Text)
pd.DataFrame(counts.sum(axis=0),columns=co.get_feature_names()).T.so
rt_values(0,ascending=False).head(50)
```

realdonaldtrump	1544
trump	1238
great	1049
thank	893
amp	627
trump2016	618
hillary	543
makeamericagreatagain	518
america	489
rt	450
people	445
new	410
poll	367
make	348
donald	331
clinton	322
cnn	312
get	308
foxnews	287
president	270
like	268
big	267
debate	264
vote	258
time	248
tonight	246
one	241

The top words are mostly campaign-related words, ie: Trump, Hillary, America, vote, poll, etc.

Now we can check for frequent bi-grams:

```
co = CountVectorizer(ngram_range=(2,2),stop_words=stops)
counts = co.fit_transform(data.Tweet_Text)
pd.DataFrame(counts.sum(axis=0),columns=co.get_feature_names()).T.so
rt_values(0,ascending=False).head(50)
```

donald trump	244
make america	243
america great	214
crooked hillary	194
hillary clinton	191
makeamericagreatagain trump2016	160
ted cruz	111
new hampshire	106
last night	92
mr trump	90
south carolina	70
new york	62
jeb bush	61
trump2016 makeamericagreatagain	59
marco rubio	58
rt danscavino	57
great job	56
rt teamtrump	45
north carolina	42
illegal immigration	41
bernie sanders	40
thank support	38
failing nytimes	37
realdonaldtrump great	36
realdonaldtrump trump	36
vote trump	36
lyin ted	35
president obama	35
look forward	33
great people	33
united states	32
crippled america	31

The most popular bi-grams are Trump's special phrases, like "crooked Hillary" and "failing NYTimes". There are also many mentions of

competitors in the bi-grams (Hillary Clinton, Ted Cruz, Marco Rubio). This already gives us information about the differences in sentiment towards specific entities in the text.

More Sophisticated Text Characteristics

Unknown words

In this step, I search the text for elements like:

- Unknown words or words that don't match known word lists.
- Words that appear especially often.
- How much of the text is slang?
- Which words are Domain-specific
- How many spelling mistakes appear in the text.

We can also find more steps to our preprocessing that will yield a better percentage of known words. This step is critical if you want to use pre-trained embeddings or pre-trained networks. In these cases, we need to compare the text data to the dictionary of the pre-trained resource.

In this analysis, I will use an English word list from https://github.com/dwyl/english-words and pre-process the text by removing hashtags, usernames and non-alphanumeric symbols.

```
words = pd.read_table('https://raw.githubusercontent.com
/dwyl/english-words/master/words.txt')
words.columns=['word']
words = words['word'].str.lower().values.tolist()


data['clean_text'] = data.Tweet_Text.apply(lambda x: ' '.join([i for
i in x.split(' ') if not (i.startswith('@') or i.startswith('#'))]))
```

```
data['clean_text'] = data.clean_text.str.lower().str.replace('[^a-
zA]', ' ')


non_list = {}
for sent in tqdm(data.clean_text.str.split().values):
    for token in sent:
        if token not in words:
            non_list[token] = 1 if token not in non_list else
non_list[token]+1

pd.Series(non_list).sort_values(ascending=False).head(30)
```

https	2441
http	446
makeamericagreatagain	116
donaldtrump	76
realdonaldtrump	75
kasich	63
false	39
lyin	39
mccain	35
clintons	33
pme	33
hillaryclinton	32
obamacare	32
dems	27
hillarys	26
emails	26
megyn	22
foxnews	19
turnberry	17
rnc	17
snl	16
putin	16
nafta	16
obamas	16
tpp	16
jebbush	16
americafirst	15
fiorina	15
facebook	12
wikileaks	11

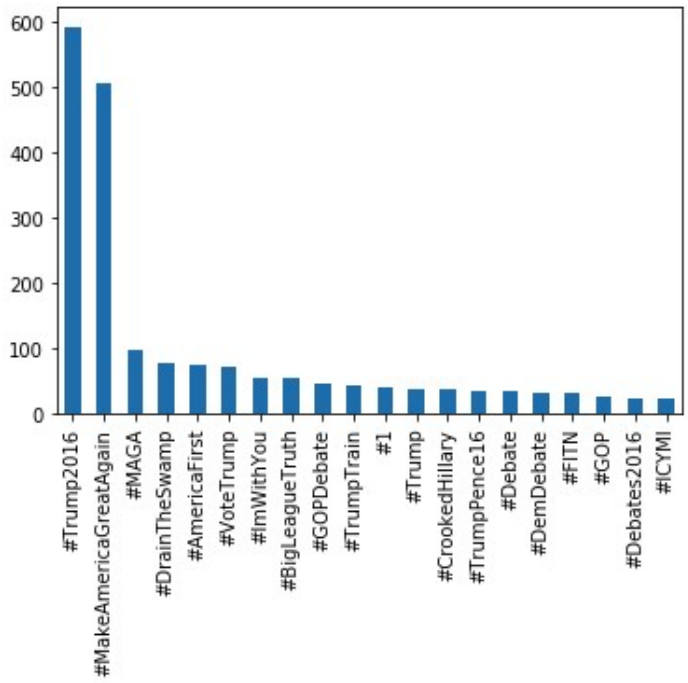
We probably need to add ″http” and ″https″ to the stop words (domain-specific stop words) list because they are frequent but not significant. Moreover, mostly names and bigrams were united (maybe we need a smarter tokenizer). Stemming or lemmatization can also help with removing more unknown words.

Feature engineering

It is possible to build some structure from the text by extracting features and then using the tools for structured data (histograms, bars, etc) on the features. Even though these features are very domain-specific, we can extract different types of entities or the sentiment of a sentence as features for almost every text dataset.

Here is a bar graph using hashtag frequency as features.

```
data.Tweet_Text.str.extractall(r'(\#\w+)')
[0].value_counts().head(20).plot.bar()
```



Using machine learning models

Topic modeling

Topic modeling is a group of unsupervised machine learning methods aimed to extract meaningful topics from a text corpus. Topic modeling is helpful in exploring a new text dataset and understanding the different types of text samples. We can also use regular clustering but topic modeling is more interpretable and thus better for exploration.

I use the Latent Dirichlet Allocation algorithm with 10 topics and print the most important words in every topic.

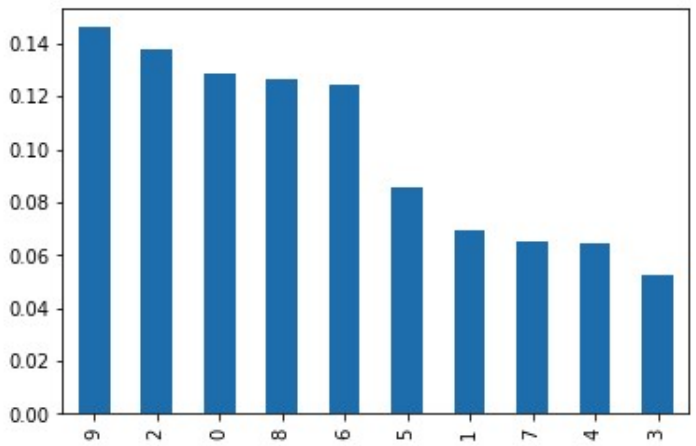
```
from sklearn.decomposition import LatentDirichletAllocation, NMF


vectorizer = CountVectorizer(stop_words=stops)
model = vectorizer.fit(data.clean_text)
docs = vectorizer.transform(data.clean_text)


lda = LatentDirichletAllocation(20)
lda.fit(docs)


def print_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        message = "Topic #%d: " % topic_idx
        message += " ".join([(feature_names[i])
                             for i in topic.argsort()[:-n_top_words
- 1:-1]])
        print(message)
    print()

print_top_words(lda,vectorizer.get_feature_names(),10)
```

```
Topic #0: hillary clinton crooked bad people cruz ted bernie said win
Topic #1: trump book pm right illegal crippled would deal bush amp
Topic #2: trump america great make donald president need vote us want
Topic #3: said rt trump want watch speech repeal club people jeb
Topic #4: trump carson thank candidate presidential rt people nation ben wow
Topic #5: trump amp get rt campaign donald debate special much better
Topic #6: amp great tonight rt enjoy interviewed morning new poll interview
Topic #7: join pm tomorrow rubio thank watching watch wall us today
Topic #8: thank new poll trump great people hampshire makeamericagreatagain amp one
Topic #9: great thank trump big crowd night people make america carolina
```

Here we see some clear topics. For example, Topic #0 about competitors, Topic #9 about rallies and Topic #2 about the campaign phrases:

Now we assess how often Trump tweets about each topic:

```
data['topic']=lda.transform(docs).argmax(axis=1)
data.topic.value_counts(normalize=True).plot.bar()
```



And we can look at examples of specific topics:

```
data[data.topic==0].Tweet_Text.sample(5).values
```

```
array(['The American people are sick and tired of not being able to lead normal lives and to constantly be on the lookout for terror and terrorists!',
       'WHAT THEY ARE SAYING ABOUT MIKE PENCE DOMINATING\u06dd THE DEBATE:\nhttps://t.co/mUw9S5GgNM #VPDebate',
       'LYIN TED https://t.co/Q0qXDGCN1l',
       'Army training slide lists Hillary Clinton as insider threat: https://t.co/CQTSo2ETJF',
       'Sanders said only black lives matter - wow! Hillary did not answer question!'],
      dtype=object)
```

Time series analysis

In many applications, our text corpus is associated with a time series (ie: news, tweets, reviews, etc.). Here we investigate the connection of the text to the time series. Now that we've built a text structure using the topic modeling, we can look at the topic distribution on the time series. We can also use time series anomaly detection and time series clustering methods to gain more insights on the topics.


```
data['Date'] = pd.to_datetime(data.Date,yearfirst=True)
data.index = data.Date
for i in range(10):
    temp = data[data.topic==i]
    temp.resample('7D').size().plot()
    plt.title('topic %s' %i)
    plt.show()
```