DEEP LEARNING, NATURAL LANGUAGE PROCESSING

Unlocking the Power of Text Analytics with Natural Language Processing

An insight into Latent Dirichlet Allocation for topic modeling and Naïve Bayes for text classification

Sharon Lim Follow
Aug 22 · 13 min read ★

Source: SurveySensum

Natural language is a language that is used for everyday communication between humans. It is highly unstructured in nature for both text and speech, thus making it difficult to parse and comprehend

by machines. **Natural language processing ("NLP") is concerned with the interaction between natural human language and computers.** It is an intersection between the fields of linguistics, computer science and artificial intelligence.

According to predictions by International Data Corporation in their report, The Digitization of the World: From Edge to Core, the total volume of data

> *will grow from 33 Zettabytes in 2018 to 175 Zettabytes in 2025.*

As per Forbes in 2019, **90% of data generated daily are unstructured data** and much of this will be text. This represents the largest data source produced and is, therefore, a rich source of data for analytics and deploying AI applications in the enterprise. However, companies are often only used to managing and analyzing 'structured' data that fits neatly within the rows and columns of a database.

In order to unlock insights from unstructured data, we can leverage on the power of text analytics, and use NLP to transform the unstructured text in documents and databases into normalized, structured data suitable for analysis.

Two main components of NLP

- **Natural Language Understanding** helps computers understand and interpret human language. Rather than requiring users to interact with computers through programming codes, NLP allows users to interact with the computer using everyday language, to which the computer can respond appropriately.
- **Natural Language Generation** is the process where computers translate data into readable human languages. The data being

translated includes the bits and bytes that make up the photos and text that appear on your computer screen.

Main Applications of Natural Language Understanding

1. **Topic modeling** extracts meaning from texts by identifying recurrent patterns or topics and unlock semantic structures behind each of the individual texts.
2. **Document classification** helps to classify discrete collections of text into categories. Examples include email spam filter, differentiating positive and negative product reviews and customer reviews.
3. **Document recommendation** selects the most relevant document based on the given information via a content-based recommender system. A good example would be the Google search engine, where the most relevant web pages are shown based on the user's query.

Main Applications of Natural Language Generation

1. **Document summarization** generates text summaries from the growing amount of text data available. With the growth of telecommuting, the ability to capture key ideas and content from conversations is gaining traction. A **speech summarization** system that could turn voice to text and generate summaries from team meetings would be interesting.
2. **Machine translation** to translate text between languages. An example would be Google Translate, which is probably the most used and well-known machine translation engine to-date.

3. **Question answering** by systems which answer questions posed by humans in a natural language. Think of the voice assistants such as Apple's Siri and Amazon's Alexa.

In this post, we shall work through an NLP workflow, explore the basics of using Latent Dirichlet Allocation for topic modeling and Naïve Bayes for text classification.

Problem statement: Perform topic modeling for a set of news corpus and develop a text classification model.

Let's start coding!

```
import os
import string
import pickle
import re, nltk
from tqdm import *
import numpy as np
import pandas as pd
from pandas import DataFrame
from datetime import datetime
from wordcloud import WordCloud
from nltk.tag import pos_tag
from nltk.collocations import *
from nltk.corpus import wordnet
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
import sklearn
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
import pyLDAvis
import pyLDAvis.sklearn
import scikitplot as skplt
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

libraries and packages.py hosted with ♡ by GitHub view raw

Data Preparation

Load the news corpus and preview a snippet of the dataset.

```
text = ''
with open('Data/News/News Set B.txt','r', encoding='utf-8') as f:
    text = " ".join([l.strip() for l in f.readlines()])

text[0:500]
```

load dataset.py hosted with ♡ by GitHub view raw

```
'URL: http://www.nytimes.com/2016/06/30/sports/baseball/washington-nationals-max-scherzer-baffles-mets-completing-a-sweep.html  WASHINGTON — Stellar p
itching kept the Mets afloat in the first half of last season despite their offensive woes. But they cannot produce an encore of their pennant-winning
season if their lineup keeps floundering while their pitching is nicked, bruised and stretched thin. “We were going to ride our pitching,” Manager Ter
ry Collins said before Wednesday’s game. “But we’r'
```

URL links were present in the news corpus, let’s utilize regular expression to remove the URL links. The ***re.compile()*** method combines a regular expression pattern into pattern objects for pattern matching. This enables us to search a pattern again without rewriting it. We would then split the individual news article by line, remove leading and trailing empty spaces.

```
# Remove URLs
p = re.compile('URL:\shttp[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|\
                [!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')

articles = p.split(text)

```

```
for i, line in enumerate(articles): articles[i] = line.strip()

# Display a sample article after initial clean-up
print(articles[1])
```

initial clean up.py hosted with ♡ by GitHub view raw

WASHINGTON — Stellar pitching kept the Mets afloat in the first half of last season despite their offensive woes. But they cannot produce an encore of their pennant-winning season if their lineup keeps floundering while their pitching is nicked, bruised and stretched thin. "We were going to ride our pitching," Manager Terry Collins said before Wednesday's game. "But we're not riding it right now. We've got as many problems with our pitching as we do anything." Wednesday's 4-2 loss to the Washington Nationals was cruel for the already-limping Mets. Pitching in Steven Matz's place, the spot starter Logan Verrett allowed two runs over five innings. But even that was too large a deficit for the Mets' lineup to overcome against Max Scherzer, the Nationals' starter. "We're not even giving ourselves chances," Collins said, adding later, "We just can't give our pitchers any room to work." The Mets did not score until the ninth inning, when a last-gasp two-run homer by James Loney off Nationals reliever Shawn Kelley snapped a streak of 23 scoreless innings for the team. The Mets were swept in the three-game series and fell six games behind the Nationals in the National League East. Of late, the Mets have looked worse than their 40-37 record. "I don't think we've played half our games yet this year," right fielder Curtis Granderson said. "There's still a lot of things left that can and hopefully will happen." Scherzer toyed with the Mets, who were initially without Granderson after he was scratched from the lineup with lingering calf tightness. Even though Granderson has been inconsistent this season, he had hit well against Scherzer in the past. Alejandro De Aza, who entered the game with a .165 average, started in right field instead because Collins said the team had few options. After Scherzer gave up a single to Asdrubal Cabrera and walked Loney in the second inning, he retired the next 18 batters, until an eighth-inning single by Brandon Nimmo. The Mets struggled again with runners on base. After Nimmo and the pinch-hitting Granderson singled in the eighth, pinch-hitter Travis d'Arnaud grounded out, and De Aza struck out. "If they keep adding pressure on themselves, they're going to continue to struggle," Collins said. "That's one of the things we try to make sure they have to understand: They have to be themselves." General Manager Sandy Alderson, Collins and the coaching staff have met about the offense and discussed the odd dynamics: Some players are performing at or better than their career averages, but the lineup as a whole has struggled immensely, especially with runners in scoring position. "We're just not driving in any runs," Collins said. "That's been the frustrating part. It's not that we're striking out. We're popping up, or a double-play ball." The Mets have a power-hitting team, so asking players to bunt or hit and run would go against their strengths. "When you start to change a team that's built one way and start to make them do something different, you're going to get your butt beat," Collins said. Earlier in the season, the Mets appeared like an all-or-nothing, home-run-driven team. Although they hit only .211 as a team in May, they smashed 40 home runs. They have a higher average in June, but they have hit only 24 homers, and the inconsistent offense has put a strain on the pitching staff. In the second inning, Verrett gave up a solo home run to the ex-Met Daniel Murphy. Collins wanted to limit the workloads of Addison Reed and Jeurys Familia, so he turned to reliever Sean Gilmartin in the eighth. Gilmartin gave up a two-run homer to Murphy, who has hit .429 (15 for 35) with four home runs against the Mets this season, his first since leaving the team. "I felt like I kept us in the game and gave us a chance to come back and win it," Verrett said. "I wish that I wouldn't have given up the two runs." Verrett was put in this position because of the effects of bone spurs on the Mets' rotation. The team asked Verrett to start Wednesday and gave Matz an extra day of rest after he received anti-inflammatory medication for the large bone spur in his left elbow. He will try to pitch through it. Noah Syndergaard has a smaller and less intrusive bone spur in the back of his right elbow. "As long as I'm staying on my anti-inflammatories and my mechanics are on point, I'm able to go out there every five days and compete," Syndergaard said. For the Mets, the immediate road ahead will be even tougher. Matz was expected to pitch Thursday against the Chicago Cubs, one of the best teams in baseball this season.

The sample article displayed above looks good to go. Let's load the processed news corpus into a dataframe and create a column named "content".

```
# Load data into a dataframe for further analysis
df = DataFrame(articles,columns=['content'])
df.head()
```

load as dataframe.py hosted with ♡ by GitHub view raw

	content
0	
1	WASHINGTON — Stellar pitching kept the Mets af...
2	Mayor Bill de Blasio's counsel and chief legal...
3	In the early morning hours of Labor Day last y...
4	It was the Apple Store in New York City before...

It appears that the first row under index 0 is empty. Let's find out the total number of articles present in the dataframe before cleaning.

```
# Display total number of news articles before cleaning
len(df)
```

total no before cleaning.py hosted with ♡ by GitHub view raw

```
8889
```

Data Cleaning

```
# Assign nan value to cells with empty fields
nan_value = float("NaN")

# Convert NaN values to empty string
df.replace("", nan_value, inplace=True)
df.dropna(subset = ["content"], inplace=True)
df.head()
```

drop empty cells.py hosted with ♡ by GitHub view raw

	content
1	WASHINGTON — Stellar pitching kept the Mets af...
2	Mayor Bill de Blasio's counsel and chief legal...

3	In the early morning hours of Labor Day last y...
4	It was the Apple Store in New York City before...
5	OMAHA — The United States Olympic swimming tri...

```
# Display total number of news articles after removing empty rows
len(df)
```

total no without empty rows.py hosted with ♡ by GitHub view raw

8884

We have successfully removed 5 empty rows from the dataframe. For the next step, we would use regular expression for the removal of words containing numbers, and replace them with a single space. Standardization is also performed by converting all the words to lowercase.

```
# Replace words containing numbers with a single space and convert words to lowercase
lower_alpha = lambda x: re.sub(r"""\w*\d\w*""", ' ', x.lower())

df['content'] = df.content.map(lower_alpha)
df.head()
```

lowercase.py hosted with ♡ by GitHub view raw

	content
1	washington — stellar pitching kept the mets af...
2	mayor bill de blasio's counsel and chief legal...
3	in the early morning hours of labor day last y...
4	it was the apple store in new york city before...
5	omaha — the united states olympic swimming tri...

The data still looks somehow odd to me, why is the hyphen still there? It would be a good idea to remove all the punctuation marks.

```
# Remove all punctuations
punc_re = lambda x: re.sub(r"""[\.,–“”’:;#$?%!&()_`*"^{|}~]""", ' ', x)

df['content'] = df.content.map(punc_re)
df.head()
```

remove punctuations.py hosted with ♡ by GitHub view raw

	content
1	washington stellar pitching kept the mets af...
2	mayor bill de blasio s counsel and chief legal...
3	in the early morning hours of labor day last y...
4	it was the apple store in new york city before...
5	omaha the united states olympic swimming tri...

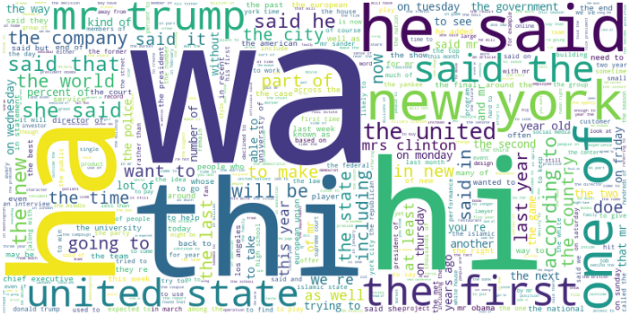
Exploratory Data Analysis

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud, thus making it useful to obtain a visual representation of most common words.

```
# Join the different processed titles together
long_string = ','.join(list(df['content'].values))

# Create a WordCloud object
wordcloud = WordCloud(background_color="white", max_words=500, width=800, height=400,
                      contour_width=20, contour_color='steelblue')

# Generate a word cloud
wordcloud.generate(long_string)

# Visualize the word cloud
```

```
wordcloud.to_image()
```

word cloud.py hosted with ♡ by GitHub view raw



Feature Engineering

The Natural Language Toolkit ("NLTK") is a Python library written for the working and modeling of text. It provides great tools for loading and cleaning text to get the data ready for working with machine learning and deep learning algorithms. Here are some of the more commonly used tools used for feature engineering:

- **Tokenization** involves the splitting of raw text into small, indivisible units for processing. NLTK provides a function - ***word_tokenize(),*** for splitting strings into tokens (nominally words). The tokens are split based on empty space and punctuation.

```
# Tokenize the reviews into words and insert the list of tokens into a new column 'tokens'
df['tokens'] = df.content.map(word_tokenize)
df.head()
```

tokenization.py hosted with ♡ by GitHub view raw

	content	tokens
1	washington stellar pitching kept the mets af...	[washington, stellar, pitching, kept, the, met...
2	mayor bill de blasio s counsel and chief legal...	[mayor, bill, de, blasio, s, counsel, and, chi...
3	in the early morning hours of labor day last y...	[in, the, early, morning, hours, of, labor, da...
4	it was the apple store in new york city before...	[it, was, the, apple, store, in, new, york, ci...
5	omaha the united states olympic swimming tri...	[omaha, the, united, states, olympic, swimming...

Sentences transformed into **individual tokens of words**

- **Stop words** are words that do not contribute to the deeper meaning to the phrase. Examples of the most common words are "*the*", "*a*", "*of*" and "*is*". NLTK provides a list of commonly agreed upon stop words for a variety of languages, such as English.

```
# Remove stop words
stop_words = stopwords.words('english')

# To include these words as stop words after insights from the initial run of code
new_stopwords = ['said','say','mr']
stop_words.extend(new_stopwords)

stop_lambda = lambda x: [y for y in x if y not in stop_words]

df['tokens_stop'] = df.tokens.apply(stop_lambda)
df.head()
```

removal of stop words.py hosted with ♡ by GitHub view raw

	content	tokens	tokens_stop
1	washington stellar pitching kept the mets af...	[washington, stellar, pitching, kept, the, met...	[washington, stellar, pitching, kept, mets, af...
2	mayor bill de blasio s counsel and chief legal...	[mayor, bill, de, blasio, s, counsel, and, chi...	[mayor, bill, de, blasio, counsel, chief, lega...
3	in the early morning hours of labor day last y...	[in, the, early, morning, hours, of, labor, da...	[early, morning, hours, labor, day, last, year...
4	it was the apple store in new york city before...	[it, was, the, apple, store, in, new, york, ci...	[apple, store, new, york, city, thing, apple, ...
5	omaha the united states olympic swimming tri...	[omaha, the, united, states, olympic, swimming...	[omaha, united, states, olympic, swimming, tri...

Removal of stop words such as "the", "in", "it" and "was".

- **Lemmatization** uses the vocabulary and morphological analysis of words and aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. NLTK provides ***WordNetLemmatizer()*** that uses the WordNet Database to lookup lemmas of words.

```
# Perform  basic lemmatization
lemmatizer = WordNetLemmatizer()
lemmatizer_lambda = lambda x: [lemmatizer.lemmatize(y) for y in x]

df['tokens_lemma_simple'] = df.tokens_stop.apply(lemmatizer_lambda)
df.head()
```

basic lemmatization.py hosted with ♡ by GitHub view raw

	content	tokens	tokens_stop	tokens_lemma_simple
1	washington stellar pitching kept the mets af...	[washington, stellar, pitching, kept, the, met...	[washington, stellar, pitching, kept, mets, af...	[washington, stellar, pitching, kept, mets, af...
2	mayor bill de blasio s counsel and chief legal...	[mayor, bill, de, blasio, s, counsel, and, chi...	[mayor, bill, de, blasio, counsel, chief, lega...	[mayor, bill, de, blasio, counsel, chief, lega...
3	in the early morning hours of labor day last y...	[in, the, early, morning, hours, of, labor, da...	[early, morning, hours, labor, day, last, year...	[early, morning, hour, labor, day, last, year,...
4	it was the apple store in new york city before...	[it, was, the, apple, store, in, new, york, ci...	[apple, store, new, york, city, thing, apple, ...	[apple, store, new, york, city, thing, apple, ...
5	omaha the united states olympic swimming tri...	[omaha, the, united, states, olympic, swimming...	[omaha, united, states, olympic, swimming, tri...	[omaha, united, state, olympic, swimming, tria...

Lemmas derived from "hours" to "hour", "states" to "state".

- **Part-of-speech tagging** or **POS-tagging** is the process of classifying words into their parts of speech and labeling them accordingly. A POS tagger is used to assign grammatical information of each word of the sentence.

```
# Perform lemmitization considering parts of speech tagging
pos_lambda = lambda x: nltk.pos_tag(x)

df['tokens_pos'] = (df.tokens_stop.apply(pos_lambda))
df.head()
```

pos tagging.py hosted with ♡ by GitHub view raw

	content	tokens	tokens_stop	tokens_lemma_simple	tokens_pos
1	washington stellar pitching kept the mets af...	[washington, stellar, pitching, kept, the, met...	[washington, stellar, pitching, kept, mets, af...	[washington, stellar, pitching, kept, mets, af...	[(washington, NN), (stellar, NN), (pitching, N...
2	mayor bill de blasio s counsel and chief legal...	[mayor, bill, de, blasio, s, counsel, and, chi...	[mayor, bill, de, blasio, counsel, chief, lega...	[mayor, bill, de, blasio, counsel, chief, lega...	[(mayor, NN), (bill, NN), (de, IN), (blasio, F...
3	in the early morning hours of labor day last y...	[in, the, early, morning, hours, of, labor, da...	[early, morning, hours, labor, day, last, year...	[early, morning, hour, labor, day, last, year,...	[(early, RB), (morning, NN), (hours, NNS), (la...
4	it was the apple store in new york city before...	[it, was, the, apple, store, in, new, york, ci...	[apple, store, new, york, city, thing, apple, ...	[apple, store, new, york, city, thing, apple, ...	[(apple, NN), (store, NN), (new, JJ), (york, N...
5	omaha the united states olympic swimming tri...	[omaha, the, united, states, olympic, swimming...	[omaha, united, states, olympic, swimming, tri...	[omaha, united, state, olympic, swimming, tria...	[(omaha, RB), (united, JJ), (states, NNS), (ol...

A few examples of the above abbreviations, along with its meaning:

NN: Noun, singular or mass

NNS: Noun, plural

IN: Preposition or subordinating conjunction

JJ: Adjective

RB: Adverb

NLTK uses the set of tags from the Penn Treebank Project, which annotates naturally-occurring text for linguistic structure.

```
# Convert the naming scheme to one that is recognized by WordNet
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return "a"
    elif treebank_tag.startswith('V'):
        return "v"
```

```
    elif treebank_tag.startswith('N'):
        return "n"
    elif treebank_tag.startswith('R'):
        return "r"
    else:
        return "n"

# Perform lemmitization using wordnet
lemmatizer = WordNetLemmatizer()
lemmatizer_fun = lambda x: lemmatizer.lemmatize(*x)

df['tokens_lemma_with_pos'] = df.tokens_pos\
                        .apply(lambda x: [(y[0], get_wordnet_pos(y[1])) for y in x])\
                        .apply(lambda x: [lemmatizer_fun(y) for y in x])
df.head()
```

wordnet.py hosted with ♡ by GitHub view raw

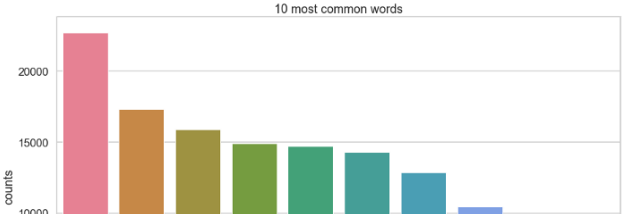
	content	tokens	tokens_stop	tokens_lemma_simple	tokens_pos	tokens_lemma_with_pos
1	washington stellar pitching kept the mets af...	[washington, stellar, pitching, kept, the, met...	[washington, stellar, pitching, kept, mets, af...	[washington, stellar, pitching, kept, mets, af...	[(washington, NN), (stellar, NN), (pitching, N...	[washington, stellar, pitching, keep, mets, af...
2	mayor bill de blasio s counsel and chief legal...	[mayor, bill, de, blasio, s, counsel, and, chi...	[mayor, bill, de, blasio, counsel, chief, lega...	[mayor, bill, de, blasio, counsel, chief, lega...	[(mayor, NN), (bill, NN), (de, IN), (blasio, F...	[mayor, bill, de, blasio, counsel, chief, lega...
3	in the early morning hours of labor day last y...	[in, the, early, morning, hours, of, labor, da...	[early, morning, hours, labor, day, last, year...	[early, morning, hour, labor, day, last, year,...	[(early, RB), (morning, NN), (hours, NNS), (la...	[early, morning, hour, labor, day, last, year,...
4	it was the apple store in new york city before...	[it, was, the, apple, store, in, new, york, ci...	[apple, store, new, york, city, thing, apple, ...	[apple, store, new, york, city, thing, apple, ...	[(apple, NN), (store, NN), (new, JJ), (york, N...	[apple, store, new, york, city, thing, apple, ...
5	omaha the united states olympic swimming tri...	[omaha, the, united, states, olympic, swimming...	[omaha, united, states, olympic, swimming, tri...	[omaha, united, state, olympic, swimming, tria...	[(omaha, RB), (united, JJ), (states, NNS), (ol...	[omaha, united, state, olympic, swim, trial, s...

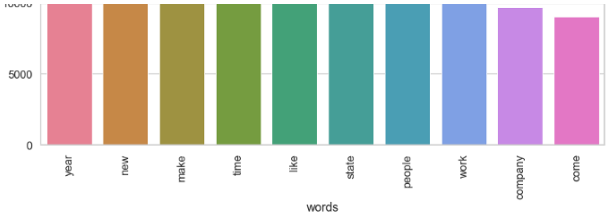
Next, we shall use the Bag of Words method to extract features from the new corpus, and build a vocabulary of all the unique words occurring therein. The new corpus is converted into a simple vector representation. Now, we're able to plot the ten most common words based on the list of vectors.

```
# Convert data type of dataframe to string
df = df.astype(str)

```

```
sns.set_style('whitegrid')

# Function to plot the top 10 common words
def plot_10_most_common_words(count_data, count_vectorizer):
    import matplotlib.pyplot as plt
    words = count_vectorizer.get_feature_names()
    total_counts = np.zeros(len(words))
    for t in count_data:
        total_counts+=t.toarray()[0]

    count_dict = (zip(words, total_counts))
    count_dict = sorted(count_dict, key=lambda x:x[1], reverse=True)[0:10]
    words = [w[0] for w in count_dict]
    counts = [w[1] for w in count_dict]
    x_pos = np.arange(len(words))

    plt.figure(2, figsize=(15, 15/1.6180))
    plt.subplot(title='10 most common words')
    sns.set_context("notebook", font_scale=1.25, rc={"lines.linewidth": 2.5})
    sns.barplot(x_pos, counts, palette='husl')
    plt.xticks(x_pos, words, rotation=90)
    plt.xlabel('words')
    plt.ylabel('counts')
    plt.show()

# Initialise the count vectorizer with the English stop words
count_vectorizer = CountVectorizer(stop_words='english')

# Fit and transform the processed titles
count_data = count_vectorizer.fit_transform(df['tokens_lemma_with_pos'])

# Visualize the 10 most common words
plot_10_most_common_words(count_data, count_vectorizer)
```

top 10 common words.py hosted with ♡ by GitHub view raw





Topic Modelling using Latent Dirichlet Allocation ("LDA")

The key idea behind LDA is that documents in the corpus are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA is a generative probabilistic model of a corpus. We shall implement LDA by using scikit-learn, an open-source Python library for machine learning.

```
# Function to display topics and sort the terms
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print ("Topic %d:" % (topic_idx))
        print (" ".join([feature_names[i]
                         for i in topic.argsort()[:-no_top_words - 1:-1]]))
```

display topics.py hosted with ♡ by GitHub view raw

To begin, define a function for the **topic model object** as above. Next, we shall prepare the **document term matrix**. The LDA topic model algorithm requires a document word matrix as the main input. Vectorize the document using count vectorizer as we can only use raw term counts for LDA, which is a probabilistic graphical model.

```
# Decide on the number of features
```

```
no_features = 1000

# Initialise the count vectorizer with the English stop words
tf_vectorizer = CountVectorizer(max_features=no_features, stop_words='english')

# Fit and transform the processed content
tf_vectorized_documents = tf_vectorizer.fit_transform(df['tokens_lemma_with_pos'])
tf_feature_names = tf_vectorizer.get_feature_names()
```

dtm.py hosted with ♡ by GitHub view raw

Next, perform **parameter selection to select the optimal number of topics.** We can use metrics such as perplexity and log-likelihood to determine good model performance. A model with higher log-likelihood and lower perplexity is considered to be good. Start by specifying an initial range of values.

Apply LDA for all possible topic size, *k*, and determine its log-likelihood and perplexity scores. **Loop through the possible range of topic numbers** by implementing a for-loop to create new LDA objects, and fit it with the document term matrix. The results are then inserted into the topic_models list created. Obtain the optimal topic size, *k,* for the LDA with the highest log-likelihood and lowest perplexity scores.

```
# Set boundaries for minimum and maximum number of topics
kmin=15
kmax=25

topic_models = []

for k in tqdm(range(kmin,kmax+1)):
    print("Applying and scoring LDA for k=%d ..." % k )

    lda_model = LatentDirichletAllocation(n_components= k, max_iter=5,
                                          learning_method='online', learning_offset=50.,
                                          random_state=8)

    lda_output = lda_model.fit_transform(tf_vectorized_documents)
    log_likelihood = lda_model.score(tf_vectorized_documents)
    perplexity = lda_model.perplexity(tf_vectorized_documents)
    topic_models.append((k,lda_model,lda_output, log_likelihood, perplexity))
```

topic loop.py hosted with ♡ by GitHub view raw

```
  0%|          | 0/11 [00:00<?, ?it/s]
Applying and scoring LDA for k=15 ...
  9%|▉         | 1/11 [01:59<19:59, 119.97s/it]
Applying and scoring LDA for k=16 ...
 18%|█▊        | 2/11 [04:01<18:03, 120.36s/it]
Applying and scoring LDA for k=17 ...
 27%|██▋       | 3/11 [06:06<16:15, 121.96s/it]
Applying and scoring LDA for k=18 ...
 36%|███▋      | 4/11 [08:14<14:25, 123.64s/it]
Applying and scoring LDA for k=19 ...
 45%|████▌     | 5/11 [10:26<12:37, 126.24s/it]
Applying and scoring LDA for k=20 ...
 55%|█████▍    | 6/11 [12:36<10:36, 127.37s/it]
Applying and scoring LDA for k=21 ...
 64%|██████▎   | 7/11 [15:05<08:54, 133.69s/it]
Applying and scoring LDA for k=22 ...
 73%|███████▎  | 8/11 [17:45<07:04, 141.64s/it]
Applying and scoring LDA for k=23 ...
 82%|████████▏ | 9/11 [20:11<04:45, 142.88s/it]
Applying and scoring LDA for k=24 ...
 91%|█████████ | 10/11 [22:32<02:22, 142.45s/it]
Applying and scoring LDA for k=25 ...
100%|██████████| 11/11 [24:56<00:00, 136.06s/it]
```

```
# Print the number of topics with its corresponding perplexity and log likelihood
for model in topic_models:
    print("k topics : % 2d, Log Likelihood : % 5.2f   Perplexity : %5.2f" %(model[0], model[3], m
```

perplexity and log likelihood.py hosted with ♡ by GitHub view raw

```
k topics :  15, Log Likelihood : -11837989.92   Perplexity : 534.98
k topics :  16, Log Likelihood : -11832079.03   Perplexity : 533.30
```

```
k topics :  17, Log Likelihood : -11824727.05   Perplexity : 531.22
k topics :  18, Log Likelihood : -11821928.54   Perplexity : 530.44
k topics :  19, Log Likelihood : -11810281.03   Perplexity : 527.17
k topics :  20, Log Likelihood : -11810193.46   Perplexity : 527.14
k topics :  21, Log Likelihood : -11802686.63   Perplexity : 525.05
k topics :  22, Log Likelihood : -11807245.27   Perplexity : 526.32
k topics :  23, Log Likelihood : -11817859.43   Perplexity : 529.29
k topics :  24, Log Likelihood : -11789824.68   Perplexity : 521.48
k topics :  25, Log Likelihood : -11790325.69   Perplexity : 521.61
```

Results showed that ***k topics: 24*** is the **optimal number of topics** that gave the highest log-likelihood and lowest perplexity scores. **Evaluate the topic and terms** using the optimal number of topics obtained, and display the top 15 words to have an idea of the topic category for each news article.

```
best_k = 24
best_lda_model = topic_models[best_k - kmin][1]
no_top_words = 15
display_topics(best_lda_model, tf_feature_names, no_top_words)
```

evaluate topics and terms.py hosted with ♡ by GitHub view raw

```
Topic 0:
million pay year company money sale sell employee accord cost include week time ticket buy
Topic 1:
trump china chinese donald campaign republican speech make country presidential ad american attack party say
Topic 2:
campaign president clinton obama house republican committee senate senator state political mrs make washington time
Topic 3:
dr health drug study patient medical year test care doctor hospital use cancer woman research
Topic 4:
clinton trump republican voter sander party state election vote campaign candidate primary democratic delegate cruz
Topic 5:
police officer city people official arrest kill report department year man shoot authority death mayor
Topic 6:
european union party britain europe country vote british political leave government minister leader london member
Topic 7:
like people time know year think want make life woman tell come work family thing
Topic 8:
percent bank market company year price billion financial stock investor oil rise rate fund investment
Topic 9:
north water island south ali white mile sea river visit japan black park mexico state
Topic 10:
```

```
company use like facebook make new people technology medium online apple google product twitter time
Topic 11:
state government official united military attack country group american war security force islamic russia year
Topic 12:
food like make day restaurant hotel cook wine good room hour driver bar come car
Topic 13:
law court justice right state gun gay supreme texas people decision issue vote case allow
Topic 14:
case court judge lawyer state law federal charge prosecutor investigation year trial legal lawsuit file
Topic 15:
city home building street house build year new space foot park room apartment square area
Topic 16:
player year team sport play game win time race world coach match tournament open club
Topic 17:
state american new united year change policy make obama tax worker president administration work increase
Topic 18:
new art work like film music make year play artist museum include theater movie time
Topic 19:
game season team run play league second hit score point series player lead make time
Topic 20:
new york article father graduate church university work receive manhattan city couple son mother com
Topic 21:
company board executive redstone chief director los angeles control ms statement trust chairman national letter
Topic 22:
deal energy williams network card brown new project year time question agreement television ask cleveland
Topic 23:
school student university college year high education child class program public percent city state new
```

24 topics obtained optimally using LDA.

Interpret the topics in a topic model that has been fit to a corpus of text data by **using pyLDAvis to visualize the topic models.** pyLDAvis is a python library for interactive topic model visualization.

As presented in the research paper LDAvis: A method for visualizing and interpreting topics by Carson Sievert and Kenny Shirley,

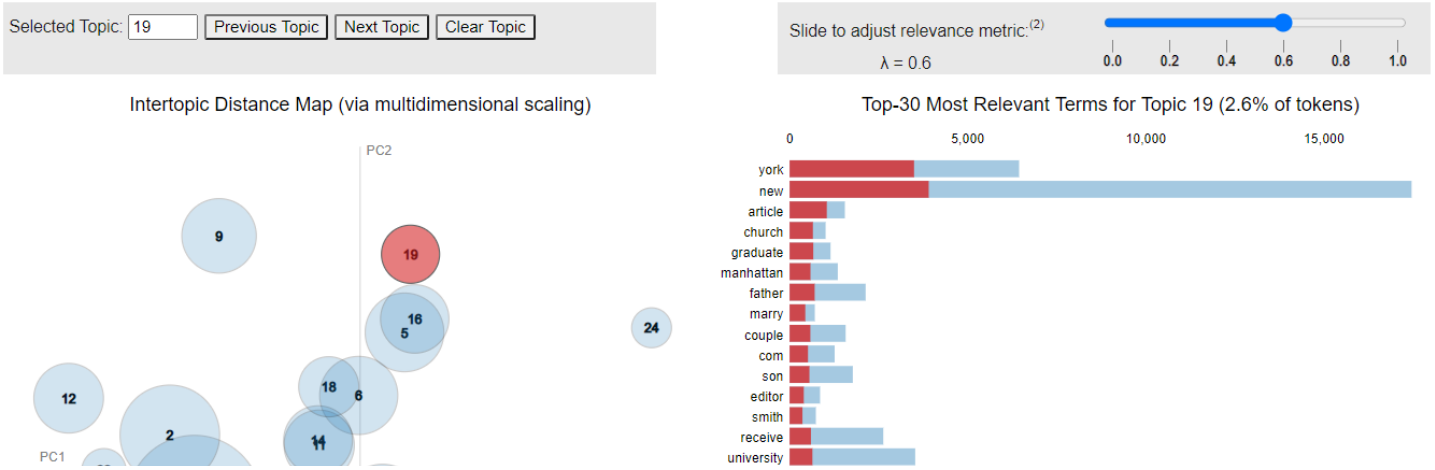
> *LDAvis, a web-based interactive visualization of topics estimated using Latent Dirichlet Allocation that is built using a combination of R and D3. Our visualization provides a global view of the topics (and how they differ from*

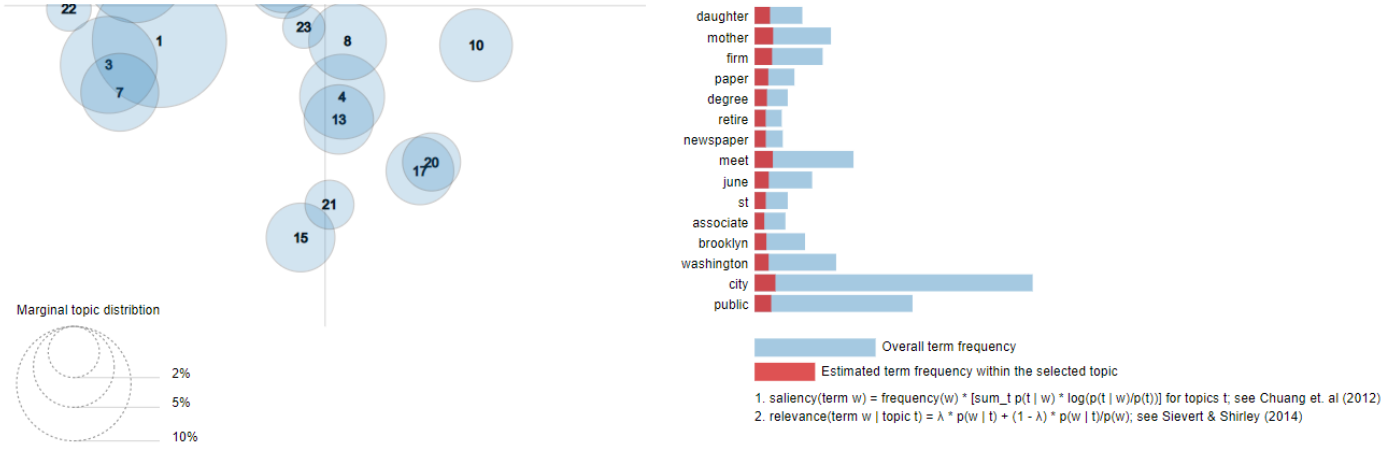
> *each other), while at the same time allowing for a deep inspection of the terms most highly associated with each individual topic.*

A good topic model will have non-overlapping with a fairly big sized area for each topic. We shall pass three parameters into the pyLDAvis, (1) LDA object, (2) document term matrix, and (3) Count Vectorizer. The relevance metric is ranked by probabilities within the topic against probability across the entire corpus. As described in the above research paper, the optimal relevance metric recommended is **λ=0.6.**

```
# Plot the pyLDAvis visualizer with the above input
pyLDAvis.enable_notebook()
panel = pyLDAvis.sklearn.prepare(best_lda_model, tf_vectorized_documents, tf_vectorizer)
panel
```

pyLDAvis.py hosted with ♡ by GitHub view raw





pyLDAvis visualizer showing the most relevant terms for topic 19.

The inter topic distance map allows us to learn about how topics relate to each other, including the potential higher-level structure between groups of topics. The **distance between circles** shows the distribution similarities closeness between the topics. Based on the above inter topic distance map, topics #1, 2, 3, and 7 relate closely with each other.

The **area of the circles** is proportional to a topic's overall prevalence. In this case, circle #1 has terms which have a higher frequency of occurrence. Also, the circles are numbered and sorted based on prevalence.

To obtain the dominant topic for a document, a logical approach is to see which topic has the highest contribution to that document and assign that topic as a label to the document. The attribute ***n_components*** contains the total number of topics, by looping for ***n_components***, an index which is provided for the topic.

```
topicnames = ["Topic " + str(i) for i in range(best_lda_model.n_components)]
print (topicnames)
```

basic topic names.py hosted with ♡ by GitHub view raw

```
['Topic 0', 'Topic 1', 'Topic 2', 'Topic 3', 'Topic 4', 'Topic 5', 'Topic 6', 'Topic 7', 'Topic 8', 'Topic 9', 'Topic 10', 'Topic 11', 'Topic 12', 'To
pic 13', 'Topic 14', 'Topic 15', 'Topic 16', 'Topic 17', 'Topic 18', 'Topic 19', 'Topic 20', 'Topic 21', 'Topic 22', 'Topic 23']
```

```
# Create document names out of the index
docnames = ["Doc" + str(i) for i in range(len(df['tokens_lemma_with_pos']))]

# Create dataframe by fitting the best LDA output to document-topic matrix
best_lda_output = best_lda_model.fit_transform(tf_vectorized_documents)
df_document_topic = pd.DataFrame(np.round(best_lda_output, 2), columns=topicnames, index=docname

# Obtain dominant topic for each document
dominant_topic = np.argmax(df_document_topic.values, axis=1)
df_document_topic['dominant_topic'] = dominant_topic

# Styling
def color_green(val):
    color = 'green' if val > .1 else 'black'
    return 'color: {col}'.format(col=color)

def make_bold(val):
    weight = 900 if val > .2 else 400
    return 'font-weight: {weight}'.format(weight=weight)

# Create copy of dataframe, apply style to all documents and display first 20 documents
df_document_topic2 = df_document_topic.head(20).style.applymap(color_green).applymap(make_bold)
df_document_topic2
```

df document topic.py hosted with ♡ by GitHub view raw

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15	Topic 16	Topic 17	Topic 18	Topic 19	Topic 20	Topic 21	Topic 22	Topic 23	dominant_topic
Doc0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	**1.000000**	0.000000	0.000000	0.000000	0.000000	**19**
Doc1	0.000000	0.000000	**0.260000**	0.000000	0.000000	**0.330000**	0.000000	0.000000	0.000000	0.000000	0.120000	0.000000	0.000000	0.000000	0.140000	0.000000	0.000000	0.000000	0.000000	0.000000	0.020000	0.110000	0.000000	0.010000	**5**
Doc2	0.000000	0.000000	0.010000	0.000000	0.000000	**0.680000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.120000	0.000000	0.000000	0.000000	0.150000	0.000000	0.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.010000	**5**
Doc3	0.200000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.140000	0.000000	0.000000	**0.380000**	0.000000	0.000000	0.000000	0.000000	**0.280000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	**10**
Doc4	0.000000	0.000000	0.000000	0.000000	0.000000	0.010000	0.000000	0.090000	0.000000	0.050000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	**0.810000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.040000	**16**

Doc5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.180000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	**0.810000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	16
Doc6	0.170000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.130000	0.000000	0.130000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.070000	0.070000	0.000000	0.070000	**0.350000**	0.000000	22
Doc7	0.000000	0.000000	0.000000	0.040000	0.000000	0.140000	0.000000	**0.210000**	0.000000	0.000000	0.000000	0.000000	0.130000	0.160000	0.000000	0.160000	0.000000	0.000000	0.110000	0.000000	0.040000	0.000000	0.000000	0.000000	7
Doc8	0.090000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.180000	0.000000	0.000000	0.000000	0.000000	0.000000	0.040000	0.020000	0.000000	0.000000	0.010000	0.000000	0.020000	**0.610000**	0.040000	0.000000	0.000000	19
Doc9	0.000000	0.000000	0.000000	0.000000	**0.430000**	0.170000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.100000	0.000000	0.180000	0.000000	0.000000	0.120000	0.000000	0.000000	4
Doc10	0.000000	0.000000	0.000000	0.010000	0.000000	0.030000	0.010000	0.090000	0.000000	0.040000	0.010000	**0.580000**	0.110000	0.000000	0.000000	0.060000	0.020000	0.020000	0.000000	0.010000	0.000000	0.000000	0.000000	0.000000	11
Doc11	0.000000	0.000000	**0.240000**	0.000000	0.000000	0.130000	0.000000	0.130000	0.000000	0.000000	0.000000	0.000000	0.070000	**0.430000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	13
Doc12	0.100000	0.000000	**0.310000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	**0.570000**	0.000000	0.000000	0.000000	0.000000	0.010000	0.000000	0.000000	0.000000	0.000000	14
Doc13	0.000000	0.000000	0.000000	0.110000	0.000000	0.030000	0.000000	**0.280000**	0.000000	0.020000	0.000000	**0.460000**	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.080000	0.000000	0.000000	0.020000	11
Doc14	0.110000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	**0.760000**	0.000000	0.040000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000	0.000000	0.000000	0.000000	8
Doc15	0.000000	0.000000	**0.860000**	0.000000	0.000000	0.010000	0.000000	0.000000	0.000000	0.000000	0.000000	0.030000	0.000000	0.000000	0.100000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2
Doc16	0.040000	0.000000	0.000000	0.000000	0.000000	0.010000	0.020000	0.100000	0.000000	0.000000	0.000000	**0.740000**	0.040000	0.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	0.020000	0.000000	0.000000	0.000000	11
Doc17	0.000000	0.000000	0.030000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.020000	0.000000	0.000000	0.110000	0.050000	0.000000	0.050000	0.000000	0.000000	0.000000	0.000000	**0.740000**	23
Doc18	0.000000	0.000000	**0.360000**	0.020000	**0.400000**	0.000000	0.000000	0.110000	0.050000	0.000000	0.000000	0.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.010000	0.000000	0.000000	0.000000	0.020000	4
Doc19	0.000000	0.000000	0.000000	0.010000	0.000000	0.000000	0.000000	0.060000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.040000	0.000000	0.000000	0.000000	0.000000	**0.860000**	0.000000	0.000000	0.040000	0.000000	19

Highest contribution for a dominant topic as represented in bold and green font.

Let's perform a visual inspection with the dominant topic identified.

Topic 19 is made up of terms as shown above in the pyLDAvis visualizer. Hence, by visual inspection, we can confirm that the dominant topic for Doc 19 was indeed topic 19.

```
1  print (f"contents of doc19 is : {df['tokens_lemma_with_pos'][19]}")
```

contents of doc19.py hosted with ♡ by GitHub view raw

```
contents of doc19 is : ['washington', 'lusty', 'applause', 'greet', 'return', 'capitol', 'behind', 'peck', 'cheek', 'receive', 'sat', 'desk', 'senate', 'floor', 'look',
'vaguely', 'glum', 'receive', 'good', 'wish', 'like', 'warrior', 'return', 'civilization', 'injure', 'intact', 'senator', 'bernie', 'sander', 'vermont', 'back', 'give',
'floor', 'speech', 'deride', 'rich', 'defend', 'misery', 'write', 'op-ed', 'piece', 'trade', 'give', 'television', 'interview', 'decline', 'fully', 'support', 'hillary',
'clinton', 'president', 'spend', 'much', 'wednesday', 'vigorously', 'denounce', 'rescue', 'bill', 'puerto', 'rico', 'support', 'two-thirds', 'fellow', 'senator', 'let',
'u', 'clear', 'sander', 'senate', 'floor', 'wednesday', 'revive', 'familiar', 'brooklyn-inflected', 'pedagogy', 'issue', 'significant', 'part', 'entire', 'debate', 'rega
rd', 'puerto', 'rico', 'billionaire', 'hedge', 'fund', 'manager', 'purchase', 'puerto', 'rican', 'bond', 'penny', 'dollar', 'want', 'percent', 'return', 'investment', 's
chool', 'shut', 'puerto', 'rico', 'pension', 'threaten', 'cut', 'child', 'island', 'go', 'hungry', 'morally', 'unacceptable', 'sander', 'withdraw', 'presidential', 'rac
e', 'really', 'leave', 'democratic', 'primary', 'battlefield', 'apparently', 'defeat', 'decidedly', 'unbowed', 'bring', 'campaign', 'capitol', 'hill', 'visibly', 'larg
e', 'security', 'detail', 'surround', 'move', 'secure', 'note', 'senator', 'tim', 'scott', 'republican', 'south', 'carolina', 'also', 'comportment', 'colleague', 'eage
r', 'expand', 'left-leaning', 'coalition', 'build', 'campaign', 'sander', 'push', 'colleague', 'take', 'policy', 'fight', 'help', 'propel', 'base', 'passion', 'give', 'n
ew', 'gravitas', 'among', 'democrat', 'addition', 'oppose', 'measure', 'aid', 'puerto', 'rico', 'work', 'hard', 'kill', 'trade', 'agreement', 'threaten', 'bill', 'woul
d', 'govern', 'label', 'genetically', 'modify', 'food', 'everyone', 'know', 'fervency', 'opinion', 'different', 'thing', 'senator', 'harry', 'reid', 'minority', 'leade
r', 'like', 'senator', 'eager', 'get', 'business', 'town', 'fourth', 'july', 'democrat', 'independent', 'sander', 'caucus', 'tolerant', 'not-quite-campaign', 'small', 'p
art', 'wish', 'emulate', 'republican', 'whose', 'wound', 'ooze', 'openly', 'many', 'democratic', 'colleague', 'especially', 'woman', 'grow', 'weary', 'progressive', 'lec
ture', 'seem', 'fit', 'dais', 'lunchroom', 'encounter', 'unwillingness', 'energetically', 'back', 'mrs', 'clinton', 'feel', 'duty', 'follower', 'raise', 'flag', 'issue',
'care', 'senator', 'claire', 'mccaskill', 'democrat', 'missouri', 'patient', 'hopeful', 'campaign', 'mrs', 'clinton', 'sander', 'seem', 'adjust', 'slightly', 'mournfull
y', 'fork', 'road', 'kind-of', 'former', 'candidate', 'definitely', 'current', 'senator', 'walk', 'hall', 'time', 'emulate', 'senator', 'elizabeth', 'warren', 'massachus
etts', 'whose', 'shoulder', 'give', 'squeeze', 'bound', 'basement', 'capitol', 'wednesday', 'brush', 'reporter', 'wave', 'time', 'becomes', 'chatty', 'talk', 'legislatio
```

n', 'despises', 'sure', 'really', 'hard', 'senator', 'barbara', 'boxer', 'democrat', 'california', 'lose', 'awful', 'eye', 'senior', 'democratic', 'slot', 'senate', 'hig
h-profile', 'committee', 'health', 'education', 'labor', 'pension', 'late', 'bipartisan', 'dismay', 'republican', 'prefer', 'senator', 'patty', 'murray', 'washington',
'senior', 'democrat', 'committee', 'even', 'democrats', 'fret', 'sander', 'might', 'try', 'push', 'committee', 'leave', 'several', 'issue', 'opportunity', 'arise', 'mich
ael', 'briggs', 'spokesman', 'sander', 'would', 'proud', 'chair', 'committee', 'deal', 'many', 'issue', 'vital', 'importance', 'american', 'people', 'decision', 'm', 'mu
rray', 'make', 'others', 'already', 'see', 'sander', 'influence', 'democrat', 'make', 'big', 'splash', 'scott', 'influence', 'see', 'hillary', 'clinton', 'campaign', 'la
st', 'month', 'especially', 'bank', 'committee', 'move', 'people', 'leave', 'sander', 'clearly', 'want', 'work', 'democrat', 'put', 'together', 'legislation', 'reflect',
'priority', 'like', 'affordable', 'college', 'still', 'work', 'get', 'much', 'agenda', 'platform', 'party', 'belong', 'convention', 'next', 'month', 'many', 'time', 'how
ever', 'believe', 'real', 'change', 'go', 'take', 'place', 'political', 'revolution', 'occur', 'million', 'people', 'stand', 'fight', 'right', 'briggs', 'sander', 'campa
ign', 'give', 'newfound', 'clout', 'among', 'democrat', 'risk', 'lose', 'endorse', 'mr', 'clinton', 'soon', 'still', 'work', 'figure', 'spend', 'capital', 'caucus', 'lac
k', 'many', 'close', 'friend', 'bernie', 'go', 'bernie', 'go', 'senator', 'dianne', 'feinstein', 'democrat', 'california', 'thing', 'think', 'something', 'constructive',
'helpful', 'ticket', 'senator', 'chuck', 'schumer', 'democrat', 'new', 'york', 'recognize', 'sander', 'appeal', 'valuable', 'party', 'try', 'harness', 'attempt', 'convin
ce', 'unite', 'around', 'mrs', 'clinton', 'best', 'way', 'defeat', 'donald', 'j', 'trump', 'november', 'caucus', 'schumer', 'general', 'view', 'bernie', 'constructive',
'force']

Text Classification using Naïve Bayes

Text classification is the process of assigning tags, labels, or categories to text according to its content. Naïve Bayes is a family of probabilistic algorithms that take advantage of the Bayes Theorem to predict the tag of a text.

A look into the research paper, Text Classification Algorithms: A Survey, by Kamran Kowsari, et al,

> *Naïve Bayes text classification has been widely used for document categorization tasks since the 1950s. The Naïve Bayes classifier method is theoretically based on Bayes theorem, which was formulated by Thomas Bayes between 1701–1761. Recent studies have widely addressed this technique in information retrieval. This technique is a generative model, which is the most traditional method of text categorization.*

```
# Display first 5 rows in dataframe
df_document_topic.head()
```

first 5 rows.py hosted with ♡ by GitHub view raw

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	...	Topic 15	Topic 16	Topic 17	Topic 18	Topic 19	Topic 20	Topic 21	Topic 22	Topic 23	dominant_topic
Doc0	0.0	0.0	0.00	0.0	0.0	0.00	0.0	0.00	0.0	0.00	...	0.00	0.00	0.0	0.00	1.0	0.00	0.00	0.0	0.00	19
Doc1	0.0	0.0	0.26	0.0	0.0	0.33	0.0	0.00	0.0	0.00	...	0.00	0.00	0.0	0.00	0.0	0.02	0.11	0.0	0.01	5

Doc2	0.0	0.0	0.01	0.0	0.0	0.68	0.0	0.00	0.0	0.00	...	0.15	0.00	0.0	0.02	0.0	0.00	0.00	0.0	0.01	5
Doc3	0.2	0.0	0.00	0.0	0.0	0.00	0.0	0.14	0.0	0.00	...	0.28	0.00	0.0	0.00	0.0	0.00	0.00	0.0	0.00	10
Doc4	0.0	0.0	0.00	0.0	0.0	0.01	0.0	0.09	0.0	0.05	...	0.00	0.81	0.0	0.00	0.0	0.00	0.00	0.0	0.04	16

5 rows × 25 columns

The dominant topics are represented as integers, thus we shall replace them and label the dataframe with the most dominant topic labels.

```
df_label = {0:'finance', 1:'us_domestic', 2:'us_domestic', 3:'health', 4:'us_domestic', 5:'us_dom
            8:'finance', 9:'world', 10:'technology', 11:'world', 12:'travel', 13:'us_domestic', 1
            16:'sports', 17:'us_domestic', 18:'arts', 19:'sports', 20:'real_estate', 21:'us_domes

df_document_topic.replace({'dominant_topic': df_label}, inplace=True)
df_document_topic
```

topic labels.py hosted with ♡ by GitHub view raw

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	...	Topic 15	Topic 16	Topic 17	Topic 18	Topic 19	Topic 20	Topic 21	Topic 22	Topic 23	dominant_topic
Doc0	0.00	0.0	0.00	0.0	0.00	0.00	0.00	0.00	0.0	0.00	...	0.00	0.00	0.0	0.00	1.00	0.00	0.00	0.0	0.00	sports
Doc1	0.00	0.0	0.26	0.0	0.00	0.33	0.00	0.00	0.0	0.00	...	0.00	0.00	0.0	0.00	0.00	0.02	0.11	0.0	0.01	us_domestic
Doc2	0.00	0.0	0.01	0.0	0.00	0.68	0.00	0.00	0.0	0.00	...	0.15	0.00	0.0	0.02	0.00	0.00	0.00	0.0	0.01	us_domestic
Doc3	0.20	0.0	0.00	0.0	0.00	0.00	0.00	0.14	0.0	0.00	...	0.28	0.00	0.0	0.00	0.00	0.00	0.00	0.0	0.00	technology
Doc4	0.00	0.0	0.00	0.0	0.00	0.01	0.00	0.09	0.0	0.05	...	0.00	0.81	0.0	0.00	0.00	0.00	0.00	0.0	0.04	sports
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Doc8879	0.00	0.0	0.23	0.0	0.69	0.00	0.00	0.00	0.0	0.00	...	0.00	0.00	0.0	0.01	0.00	0.00	0.00	0.0	0.00	us_domestic
Doc8880	0.09	0.0	0.00	0.0	0.00	0.54	0.00	0.21	0.0	0.00	...	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.0	0.00	us_domestic
Doc8881	0.07	0.0	0.00	0.0	0.00	0.00	0.11	0.15	0.0	0.04	...	0.00	0.00	0.0	0.00	0.01	0.00	0.00	0.0	0.06	travel
Doc8882	0.00	0.0	0.00	0.0	0.00	0.27	0.00	0.53	0.0	0.00	...	0.00	0.00	0.0	0.18	0.00	0.00	0.00	0.0	0.00	lifestyle
Doc8883	0.00	0.0	0.00	0.0	0.00	0.15	0.00	0.32	0.0	0.07	...	0.06	0.00	0.0	0.00	0.00	0.00	0.05	0.0	0.00	lifestyle

8884 rows × 25 columns

```
# Extract topic labels to list
topic_labels = df_document_topic['dominant_topic'].tolist()

# Add column with topic labels to dataframe with cleaned data for 'content'
df['topic labels'] = topic_labels
```

```
df
```

add topic labels to df.py hosted with ♡ by GitHub view raw

	content	tokens	tokens_stop	tokens_lemma_simple	tokens_pos	tokens_lemma_with_pos	topic labels
1	washington stellar pitching kept the mets af...	['washington', 'stellar', 'pitching', 'kept', ...	['washington', 'stellar', 'pitching', 'kept', ...	['washington', 'stellar', 'pitching', 'kept', ...	[('washington', 'NN'), ('stellar', 'NN'), ('pi...	['washington', 'stellar', 'pitching', 'keep', ...	sports
2	mayor bill de blasio s counsel and chief legal...	['mayor', 'bill', 'de', 'blasio', 's', 'counse...	['mayor', 'bill', 'de', 'blasio', 'counsel', '...	['mayor', 'bill', 'de', 'blasio', 'counsel', '...	[('mayor', 'NN'), ('bill', 'NN'), ('de', 'IN')...	['mayor', 'bill', 'de', 'blasio', 'counsel', '...	us_domestic
3	in the early morning hours of labor day last y...	['in', 'the', 'early', 'morning', 'hours', 'of...	['early', 'morning', 'hours', 'labor', 'day', ...	['early', 'morning', 'hour', 'labor', 'day', '...	[('early', 'RB'), ('morning', 'NN'), ('hours',...	['early', 'morning', 'hour', 'labor', 'day', '...	us_domestic
4	it was the apple store in new york city before...	['it', 'was', 'the', 'apple', 'store', 'in', '...	['apple', 'store', 'new', 'york', 'city', 'thi...	['apple', 'store', 'new', 'york', 'city', 'thi...	[('apple', 'NN'), ('store', 'NN'), ('new', 'JJ...	['apple', 'store', 'new', 'york', 'city', 'thi...	technology
5	omaha the united states olympic swimming tri...	['omaha', 'the', 'united', 'states', 'olympic'...	['omaha', 'united', 'states', 'olympic', 'swim...	['omaha', 'united', 'state', 'olympic', 'swimm...	[('omaha', 'RB'), ('united', 'JJ'), ('states',...	['omaha', 'united', 'state', 'olympic', 'swim'...	sports
...	...	...	...	...	...	...	...
8884	there is a second critical contest in america ...	['there', 'is', 'a', 'second', 'critical', 'co...	['second', 'critical', 'contest', 'america', '...	['second', 'critical', 'contest', 'america', '...	[('second', 'JJ'), ('critical', 'JJ'), ('conte...	['second', 'critical', 'contest', 'america', '...	us_domestic
8885	on april police officers from the precinc...	['on', 'april', 'police', 'officers', 'from', ...	['april', 'police', 'officers', 'precinct', 'c...	['april', 'police', 'officer', 'precinct', 'ca...	[('april', 'JJ'), ('police', 'NN'), ('officers...	['april', 'police', 'officer', 'precinct', 'ca...	us_domestic
8886	khilad india the cattle camp on a dusty ba...	['khilad', 'india', 'the', 'cattle', 'camp', '...	['khilad', 'india', 'cattle', 'camp', 'dusty',...	['khilad', 'india', 'cattle', 'camp', 'dusty',...	[('khilad', 'NNS'), ('india', 'VBP'), ('cattle...	['khilad', 'india', 'cattle', 'camp', 'dusty',...	travel
8887	the director j j abrams dropped a possible hi...	['the', 'director', 'j', 'j', 'abrams', 'dropp...	['director', 'j', 'j', 'abrams', 'dropped', 'p...	['director', 'j', 'j', 'abrams', 'dropped', 'p...	[('director', 'NN'), ('j', 'NN'), ('j', 'NN'),...	['director', 'j', 'j', 'abrams', 'drop', 'poss...	lifestyle
8888	palos verdes estates calif from high atop ...	['palos', 'verdes', 'estates', 'calif', 'from'...	['palos', 'verdes', 'estates', 'calif', 'high'...	['palos', 'verdes', 'estate', 'calif', 'high',...	[('palos', 'NN'), ('verdes', 'NNS'), ('estates...	['palos', 'verdes', 'estates', 'calif', 'high'...	lifestyle

8884 rows × 7 columns

```
# Create new dataframe with required columns for training
df_clean = df.filter(['content'], axis=1)
df_clean['topic_labels'] = topic_labels
df_clean
```

df clean.py hosted with ♡ by GitHub view raw

	content	topic_labels

1	washington stellar pitching kept the mets af...	sports
2	mayor bill de blasio s counsel and chief legal...	us_domestic
3	in the early morning hours of labor day last y...	us_domestic
4	it was the apple store in new york city before...	technology
5	omaha the united states olympic swimming tri...	sports
...	...	...
8884	there is a second critical contest in america ...	us_domestic
8885	on april police officers from the precinc...	us_domestic
8886	khilad india the cattle camp on a dusty ba...	travel
8887	the director j j abrams dropped a possible hi...	lifestyle
8888	palos verdes estates calif from high atop ...	lifestyle

8884 rows × 2 columns

The dataframe is now ready for training. Split the dataframe into input texts and output labels.

```
# Split the data into input texts and output labels
X = df_clean.content
y = df_clean.topic_labels

X.head()
```

split dataframe.py hosted with ♡ by GitHub view raw

```
1    washington   stellar pitching kept the mets af...
2    mayor bill de blasio s counsel and chief legal...
3    in the early morning hours of labor day last y...
4    it was the apple store in new york city before...
5    omaha   the united states olympic swimming tri...
Name: content, dtype: object
```

```
y.head()
```

labels.py hosted with ♡ by GitHub view raw

```
1         sports
2    us_domestic
3    us_domestic
4     technology
5         sports
Name: topic_labels, dtype: object
```

```
# Split dataset for training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

train test split.py hosted with ♡ by GitHub view raw

Use the count vectorizer to convert a collection of text documents to a matrix of token counts by using scikit-learn's built-in stop word list for English. The ***ngram_range*** represents the lower and upper boundary of the range of n-values for different word n-grams or char n-grams to be extracted. All values of n such such that min_n <= n <= max_n will be used. In this case, the default **ngram_range** of (1,1), ie. unigrams was used.

```
cv = CountVectorizer(stop_words='english', ngram_range=(1,1))

# Fit_transform learns the vocab and perform one-hot encoding
X_train_cv = cv.fit_transform(X_train)

# Transform uses the same vocab and perform one-hot encoding
X_test_cv  = cv.transform(X_test)

# Print the dimensions of the training set in the format (# text messages, # terms)
print(X_train_cv.toarray().shape)
print(X_test_cv.toarray().shape)
```

count vectorizer.py hosted with ♡ by GitHub view raw

```
(7107, 90635)
(1777, 90635)
```

The multinomial Naive Bayes model was selected to implement the Naive Bayes algorithm for multinomially distributed data. It is one of the two classic naive Bayes variants used in text classification, where the data are typically represented as word vector counts. Fit the model on the training data and apply the fitted model to predict the topic labels of the test set.

```
# Create and fit a Naive Bayes model
nb = MultinomialNB()
nb.fit(X_train_cv, y_train)

# Use model that was trained on the X_train_cv data and apply it to X_test_cv data
y_pred_cv_nb = nb.predict(X_test_cv)

# Display output for the predictions
y_pred_cv_nb
```

naive bayes model.py hosted with ♡ by GitHub view raw

```
array(['us_domestic', 'us_domestic', 'lifestyle', ..., 'lifestyle',
       'sports', 'us_domestic'], dtype='<U11')
```

Evaluation of Results

Use of Decision Trees and Random Forest in Machine Learning

An Insight into Supervised Learning for Classification Problems

medium.com

Do check out my above post for an in-depth understanding on **confusion matrix** and **classification report.**

1. Confusion Matrix

```
sns.set_context('talk')
sns.set_palette('dark')
sns.set_style('ticks')

skplt.metrics.plot_confusion_matrix(y_test, y_pred_cv_nb, normalize=True,figsize=(28,16),
                                    title_fontsize='large',text_fontsize='medium')
plt.show()
```

confusion matrix.py hosted with ♡ by GitHub view raw

Based on the above-normalized confusion matrix, the Naïve Bayes classifier has performed a good job with a high percentage of true labels predicted.

2. Classification Report and Error Metrics

```
print(metrics.classification_report(y_test, y_pred_cv_nb))
```

classification report.py hosted with ♡ by GitHub view raw

	precision	recall	f1-score	support
arts	0.87	0.89	0.88	217
education	1.00	0.40	0.57	25
finance	0.80	0.83	0.82	127
health	0.85	0.64	0.73	61
lifestyle	0.72	0.79	0.75	245
real_estate	0.96	0.83	0.89	160
sports	0.96	0.92	0.94	232
technology	0.85	0.65	0.74	69
travel	0.85	0.72	0.78	74
us_domestic	0.79	0.83	0.81	407
world	0.73	0.87	0.79	160
accuracy			0.82	1777
macro avg	0.85	0.76	0.79	1777
weighted avg	0.83	0.82	0.82	1777

The **classification report** shows a representation of the main classification metrics **on a per-class basis** and gives a deeper intuition of the classifier behavior over global accuracy. The metrics are defined in terms of true and false positives, and true and false negatives.

A significant area to take note would be the scores for the “education” category. A precision score of 1.00 was achieved, which translates to a 100% score for all instances classified as positive. However, a poor recall score of 0.4 (< 0.5) was obtained with a slightly above average f1-score of 0.57. This indicates that **for all instances that were actually positive, a low percentage was in fact classified correctly**. With low support of 25 actual occurrences for “educational” articles in the dataset, perhaps training

more of articles within the “education” category will help to improve the scores.

```
print("Accuracy score:", sklearn.metrics.accuracy_score(y_test, y_pred_cv_nb, normalize=True))
print("Precision score:", sklearn.metrics.precision_score(y_test, y_pred_cv_nb, average='weighted
print("Recall score:", sklearn.metrics.recall_score(y_test, y_pred_cv_nb, average='weighted'))
print("f1 score:", sklearn.metrics.f1_score(y_test, y_pred_cv_nb, average='weighted'))
```

error metrics.py hosted with ♡ by GitHub view raw

```
Accuracy score: 0.8244231851435003
Precision score: 0.8321953692495975
Recall score: 0.8244231851435003
f1 score: 0.8238495981219915
```

On a global basis, we have also computed various **error metrics** such as accuracy, precision, recall and f1-score, by comparing the test labels against the predicted labels. Overall, the scores obtained for the Naïve Bayes classifier were well above the baseline of 0.5.

Deployment of Trained Model

```
# Save the model for reloading in the deployed application
# Save the counter vectorizer in order to retain the vocabulary information

time = datetime.now().strftime("%Y-%m-%d")
path1 = 'nb_classifier-{}.pkl'.format(time)

path2 = 'nb_countvectoriser-{}.pkl'.format(time)
with open(path1, 'wb') as f1:
    pickle.dump(nb, f1)

with open(path2, 'wb') as f2:
    pickle.dump(cv, f2)
```

save model and count vectoriser.py hosted with ♡ by GitHub view raw

```
# Reload saved count vectorizer
path2 = "nb_countvectoriser-2020-08-16.pkl"
with open(path2, 'rb') as f:
    trained_cv = pickle.load(f)

# Reload saved model
path1 = "nb_classifier-2020-08-16.pkl"
with open(path1, 'rb') as f:
    model = pickle.load(f)

# Check the class label of reloaded model
model.classes_
```

reload saved models.py hosted with ♡ by GitHub view raw

```
array(['arts', 'education', 'finance', 'health', 'lifestyle',
       'real_estate', 'sports', 'technology', 'travel', 'us_domestic',
       'world'], dtype='<U11')
```

```
# Preprocess new text
def preprocess(text):
    alphanumeric = lambda x: re.sub(r"""\w*\d\w*""", ' ', x)
    punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x.lower())
    text = alphanumeric(text)
    text = punc_lower(text)
    return text

# Numerically encode the input
def encode_text_to_vector(cv, text):
    new_cv = CountVectorizer(stop_words='english', vocabulary=cv.vocabulary_)
    text_vector = new_cv.fit_transform( [text ] )
    return text_vector
```

preprocess and encode new text.py hosted with ♡ by GitHub view raw

Prediction of Topic Names

With the trained model, let's request the user to input the new text.

The example below takes in a new text and call the functions defined above for pre-processing and prediction of the label.

```
new_text = input("Enter the new text: \n\n")
new_text2 = preprocess(new_text)
new_text_vector = encode_text_to_vector(trained_cv, new_text2)
predicted_label = (model.predict(new_text_vector))
predicted_prob = model.predict_proba(new_text_vector)

print ("\n\n The text is predicted as <" , predicted_label , ">.")
```

predict label for new text.py hosted with ♡ by GitHub view raw

```
Enter the new text:

 "The chaotic response to the coronavirus in Brazil, where it has killed more than 105,000 people, made the country's experience a cautionary tale tha
t many around the world have watched with alarm.  But as the country's caseload soared, vaccine researchers saw a unique opportunity.  With sustained
widespread contagion, a deep bench of immunization experts, a robust medical manufacturing infrastructure and thousands of vaccine trial volunteers, B
razil has emerged as a potentially vital player in the global scramble to end the pandemic.  Three of the most promising and advanced vaccine studies
in the world are relying on scientists and volunteers in Brazil, according to the World Health Organization's report on the progress of vaccine resear
ch.  The embattled government hopes its citizens could be among the first in the world to be inoculated. And medical experts are imagining the possibi
lity that Brazil could even manufacture the vaccine and export it to neighboring countries, a prospect that fills them with something that has been in
short supply this year: pride.  "I'm very optimistic," said Dimas Covas, the director of the Butantan Institute, an internationally renowned biopharma
ceutical producer that is partnering with China's Sinovac on one of the studies that has reached the third stage of research, during which potential v
accines are tested on 9,000 people.  "Brazil will be one of the first countries to have the vaccine." "


 The text is predicted as < ['health'] >.
```

```
# Display probability scores associated to the predicted labels
predicted_prob
```

predicted probability.py hosted with ♡ by GitHub view raw

```
array([[6.54864708e-67, 4.96427662e-80, 6.63576197e-41, 2.61062209e-18,
        1.22819273e-30, 5.82606390e-57, 9.18445778e-66, 4.02529655e-34,
        3.04238852e-78, 2.63059154e-14, 1.00000000e+00]])
```

```
# Display the classes (labels) that the model can predict
model.classes_
```

model classes.py hosted with ♡ by GitHub view raw

```
array(['arts', 'education', 'finance', 'health', 'lifestyle',
       'real_estate', 'sports', 'technology', 'travel', 'us_domestic',
       'world'], dtype='<U11')
```

Results for more sample news snippets

```
Enter the new text:

 "Kremlin critic Alexei Navalny was fighting for his life in a Siberian hospital on Thursday (Aug 20) after drinking tea that allies said they believe
was laced with poison.  If confirmed, it would be the latest in a long series of poisonings and suspected poisonings of people who have fallen out wit
h the Kremlin, which denies settling scores with its foes by murdering them.  Navalny, a fierce critic of President Vladimir Putin and his lieutenant
s, began feeling ill on a plane to Moscow on Thursday morning after drinking tea at an airport cafe in the Siberian city of Tomsk.  His condition beca
me so serious that the plane made an emergency landing at the city of Omsk, en route to Moscow, where he was carried off on a stretcher.  Kira Yarmys
h, his spokeswoman, said he was in intensive care in a serious but stable condition, and on an artificial lung ventilator in a hospital in the city, a
bout 2,200km east of the Russian capital.  ""We assume that Alexei was poisoned with something mixed into his tea. It was the only thing that he drank
in the morning. Alexei is now unconscious,"" Yarmysh said. Kremlin spokesman Dmitry Peskov said that any poisoning would need to be confirmed by labor
atory tests and that doctors were doing everything they could to help Navalny. He wished him a speedy recovery.  The incident coincides with a politic
al crisis in Belarus, a close Russian ally, and comes ahead of regional Russian elections next month.  Some anti-Kremlin protesters in Russia's far ea
st have started chanting ""Long live Belarus!"" in support of the protesters in Minsk, 9,000 km (5,590 miles) to the west.  ""Putin is scared,"" said
an EU diplomat, who declined to be named. ""He is sending a message to his own people not to try do at home what they see on TV from Belarus.""  US Pr
esident Donald Trump, asked about the situation, told reporters at the White House that his administration was looking at the matter.  Doctors gave co
ntradictory information about Navalny's condition, saying it had stabilised and that he was in a coma, but also that there was still a threat to his l
ife and they were working to save him."


 The text is predicted as < ['world'] >.
```

. . .

```
Enter the new text:

 "Facebook on Friday (Aug 21) pushed for legislation that makes it easier for users to transfer photos and videos to a rival tech platform, in comment
s it sent to the Federal Trade Commission ahead of a hearing on the topic on Sep 22.  Data portability - considered a potential remedy for large techn
ology companies whose control of social media material makes it harder for smaller rivals to get started - has become a key part of the antitrust deba
te in the United States and Europe.  In April, Facebook allowed users in the United States and Canada to transfer photos and videos to Alphabet-owned
Google Photos for the first time - a move that is likely to help the company respond to U.S. regulators and lawmakers, who are investigating its compe
titive practices and allegations it has stifled competition.  ""The FTC often issues reports following these workshops ... I think their recommendatio
ns should include dedicated portability legislation,"" Bijan Madhani, privacy and public policy manager at Facebook told Reuters.  Facebook supports a
portability bill already doing the rounds in Congress called the Access Act from Democratic Senators Richard Blumenthal and Mark Warner, and Republica
n senator Josh Hawley. It would require large tech platforms to let their users easily move their data to other services.  The bill is a good first st
ep, Madhani said. Facebook has engaged with the lawmakers on it and will continue working with them, he added."


 The text is predicted as < ['technology'] >.
```

. . .

```
Enter the new text:

 "Germany: Antonio Conte may not welcome comparisons with Jose Mourinho given the bad blood between the two, but for the first time since the Portugue
se's departure, Conte has made Inter Milan a force to be reckoned with.  Friday's (Aug 21) Europa League final against Sevilla in Cologne will be the
Nerazzurri's first European final since Mourinho walked out on the high of securing the only ever treble of Champions League, Serie A and Coppa Italia
in 2010. It is nine years since Inter won any trophy, but Conte has developed a habit of making an instant impact in his managerial career.  He took B
ari and Sienna into Serie A, began Juventus' hegemony of the Italian top-flight by winning three straight titles, took an unfancied Italy to the semi-
finals of Euro 2016 and won the Premier League in his first season at Chelsea.  ""You can tell he's a winner. He wants to win, and he demands that fro
m all the players, a winning mentality,"" said Ashley Young, who has slotted seamlessly into Conte's system since a January move from Manchester Unite
d. ""He demands everything, not just in matches, in training as well.""  Conte's demands, however, mean he rarely hangs around for the long haul. His
three-year spell at Juventus is the longest of a career already spanning seven clubs and his country at just 51.  Even after a first season that saw I
nter close the gap on Juventus to just one point at the top of Serie A after the champions eased off in the final weeks, Conte plunged his future at t
he San Siro into doubt when on the final day of the domestic season he claimed he and his players had been offered ""little protection from the clu
b.""  In a familiar complaint from his time at Juventus and Chelsea, Conte blamed a Champions League exit at the group stage to a lack of investment e
ven though the club had smashed its transfer record to sign Romelu Lukaku for 75 million euros (US$89 million) as part of the fourth highest transfer
spend in Europe."


 The text is predicted as < ['sports'] >.
```

Conclusion

The modeling goal of LDA has been achieved successfully. The optimal number of topics was obtained from the new corpus for the evaluation of the topic and terms. With that, we were **able to determine the various topic categories and unlock the hidden topic structures**.

In addition, the use of the Naïve Bayes classifier has yet again proved to be suitable for text classification. With a large amount of data used for training, the outcome of the topic modeling and text classification was satisfactory based on the evaluation of the results.

By testing the trained model with various news snippets extracted from the internet, the **results have proven to be positive and within the expectations of what the topic labels should be**.

. . .

References

1. The Digitization of the World: From Edge to Core
2. What is Text Mining, Text Analytics and Natural Language Processing
3. Unlocking Insights from Unstructured Data with Text Mining
4. How to Clean Text for Machine Learning with Python
5. LDAvis: A method for visualizing and interpreting topics
6. Text Classification Algorithms: A Survey

Text Analytics | Naturallanguageprocessing | NLP | Lda | Naive Bayes