Sentiment Analysis witth TextBlob – Part I

Create a TextBlob

First, the import.

```
>>> from textblob import TextBlob
```

Let’s create our first **TextBlob**.

```
>>> wiki = TextBlob("Python is a high-level, general-purpose programming language.")
```

Part-of-speech Tagging

Part-of-speech tags can be accessed through the **tags** property.

```
>>> wiki.tags

[('Python', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('high-level', 'JJ'), ('general-purpose', 'JJ'),
('programming', 'NN'), ('language', 'NN')]
```

Noun Phrase Extraction

Similarly, noun phrases are accessed through the **noun_phrases** property.

```
>>> wiki.noun_phrases

WordList(['python'])
```

Sentiment Analysis

The **sentiment** property returns a namedtuple of the form `Sentiment(polarity, subjectivity)`. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

```
>>> testimonial = TextBlob("Textblob is amazingly simple to use. What great fun!")

>>> testimonial.sentiment

Sentiment(polarity=0.39166666666666666, subjectivity=0.4357142857142857)

>>> testimonial.sentiment.polarity

0.39166666666666666
```

Tokenization

You can break TextBlobs into words or sentences.

```
>>> zen = TextBlob("Beautiful is better than ugly. "
...                "Explicit is better than implicit. "
...                "Simple is better than complex.")
>>> zen.words
WordList(['Beautiful', 'is', 'better', 'than', 'ugly', 'Explicit', 'is', 'better', 'than', 'implicit',
'Simple', 'is', 'better', 'than', 'complex'])
>>> zen.sentences
[Sentence("Beautiful is better than ugly."), Sentence("Explicit is better than implicit."),
Sentence("Simple is better than complex.")]
```

Sentence objects have the same properties and methods as TextBlobs.

```
>>> for sentence in zen.sentences:
...     print(sentence.sentiment)
```

Words Inflection and Lemmatization

Each word in **TextBlob.words** or **Sentence.words** is a **Word** object (a subclass of unicode) with useful methods, e.g. for word inflection.

```
>>> sentence = TextBlob('Use 4 spaces per indentation level.')
>>> sentence.words
WordList(['Use', '4', 'spaces', 'per', 'indentation', 'level'])
>>> sentence.words[2].singularize()
'space'
>>> sentence.words[-1].pluralize()
'levels'
```

WordLists

A **WordList** is just a Python list with additional methods.

```
>>> animals = TextBlob("cat dog octopus")

>>> animals.words

WordList(['cat', 'dog', 'octopus'])

>>> animals.words.pluralize()

WordList(['cats', 'dogs', 'octopodes'])
```

Spelling Correction

Use the **correct()** method to attempt spelling correction.

```
>>> b = TextBlob("I havv goood speling!")

>>> print(b.correct())

I have good spelling!
```

Word objects have a **spellcheck() Word.spellcheck()** method that returns a list of (word, confidence) tuples with spelling suggestions.

```
>>> from textblob import Word

>>> w = Word('falibility')

>>> w.spellcheck()

[('fallibility', 1.0)]
```

Get Word and Noun Phrase Frequencies

There are two ways to get the frequency of a word or noun phrase in a **TextBlob**.

The first is through the word_counts dictionary.

```
>>> monty = TextBlob("We are no longer the Knights who say Ni. "

...                     "We are now the Knights who say Ekki ekki ekki PTANG.")

>>> monty.word_counts['ekki']

3
```

If you access the frequencies this way, the search will *not* be case sensitive, and words that are not found will have a frequency of 0.

The second way is to use the `count()` method.

```
>>> monty.words.count('ekki')

3
```

You can specify whether or not the search should be case-sensitive (default is `False`).

```
>>> monty.words.count('ekki', case_sensitive=True)

2
```

Each of these methods can also be used with noun phrases.

```
>>> wiki.noun_phrases.count('python')

1
```

Translation and Language Detection

TextBlobs can be translated between languages.

```
>>> en_blob = TextBlob(u'Simple is better than complex.')

>>> en_blob.translate(to='es')

TextBlob("Lo simple es mejor que lo complejo.")
```

If no source language is specified, TextBlob will attempt to detect the language. You can specify the source language explicitly, like so. Raises **TranslatorError** if the TextBlob cannot be translated into the requested language or **NotTranslated** if the translated result is the same as the input string.

```
>>> chinese_blob = TextBlob(u"美丽优于丑陋")

>>> chinese_blob.translate(from_lang="zh-CN", to='en')

TextBlob("Beauty is better than ugly")
```

You can also attempt to detect a TextBlob’s language using **TextBlob.detect_language()**.

```
>>> b = TextBlob(u"بسيط هو أفضل من مجمع")

>>> b.detect_language()

'ar'
```

As a reference, language codes can be found here.

Language translation and detection is powered by the Google Translate API.

TextBlobs Are Like Python Strings!

You can use Python’s substring syntax.

```
>>> zen[0:19]

TextBlob("Beautiful is better")
```

You can use common string methods.

```
>>> zen.upper()

TextBlob("BEAUTIFUL IS BETTER THAN UGLY. EXPLICIT IS BETTER THAN
IMPLICIT. SIMPLE IS BETTER THAN COMPLEX.")

>>> zen.find("Simple")

65
```

You can make comparisons between TextBlobs and strings.

```
>>> apple_blob = TextBlob('apples')

>>> banana_blob = TextBlob('bananas')

>>> apple_blob < banana_blob

True

>>> apple_blob == 'apples'

True
```

You can concatenate and interpolate TextBlobs and strings.

```
>>> apple_blob + ' and ' + banana_blob

TextBlob("apples and bananas")

>>> "{0} and {1}".format(apple_blob, banana_blob)

'apples and bananas'
```

n-grams

The **TextBlob.ngrams()** method returns a list of tuples of **n** successive words.

```
>>> blob = TextBlob("Now is better than never.")

>>> blob.ngrams(n=3)

[WordList(['Now', 'is', 'better']), WordList(['is', 'better', 'than']), WordList(['better', 'than',
'never'])]
```

```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
%matplotlib inline
```

```
data_source_url = "https://raw.githubusercontent.com/kolaveridi/kaggle-Twitter-US-Airline-Sentiment-/master/Tweets.csv"
airline_tweets = pd.read_csv(data_source_url)
```

```
airline_tweets.head()
```

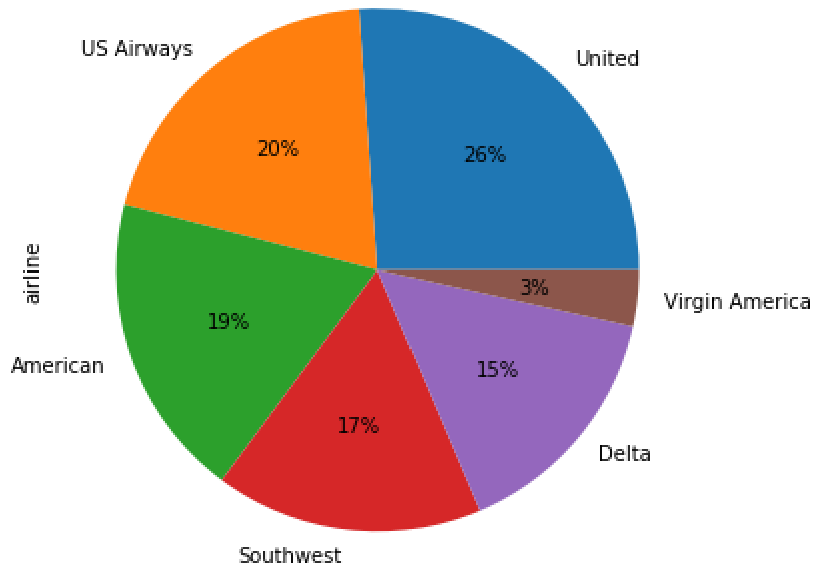
	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	air
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	

```
plot_size = plt.rcParams["figure.figsize"]
print(plot_size[0])
print(plot_size[1])
plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size
```

```
6.0
4.0
```

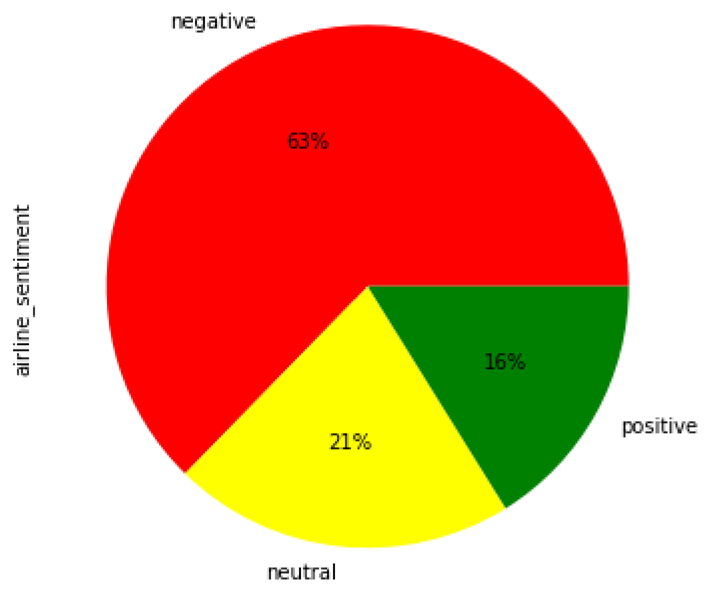
```
airline_tweets.airline.value_counts().plot(kind='pie', autopct='%1.0f%%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe41fdf19e8>
```



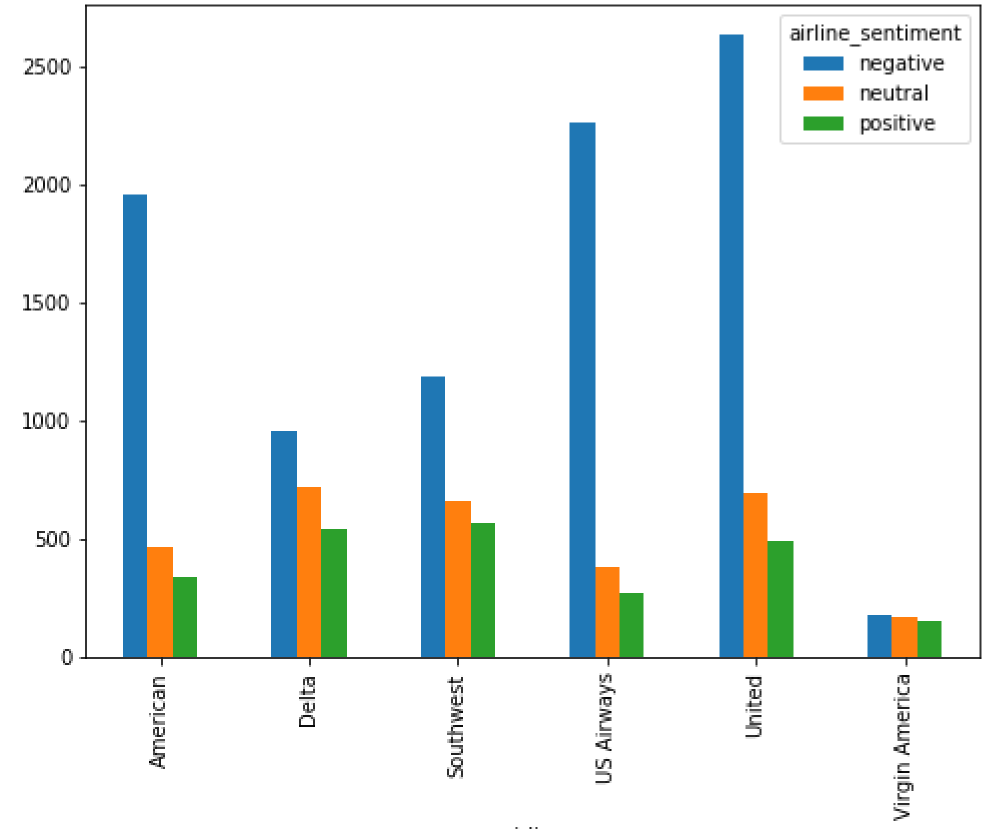
```
airline_tweets.airline_sentiment.value_counts().plot(kind='pie', autopct='%1.0f%%', colors=["red", "yellow", "green"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe41fd2ec18>
```



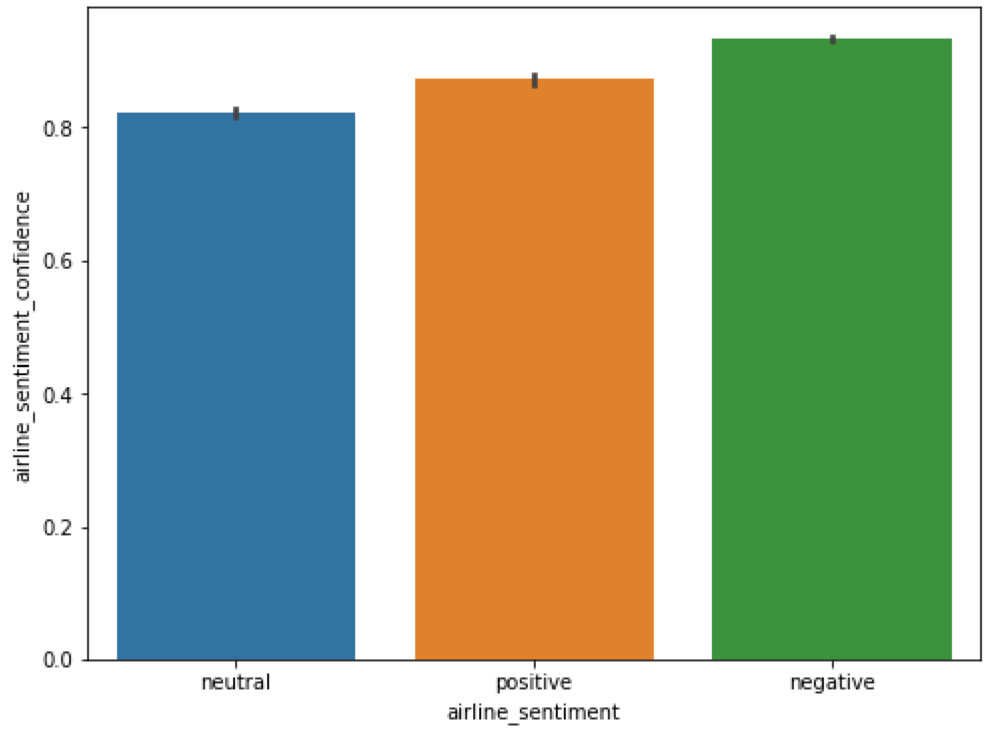
```
airline_sentiment = airline_tweets.groupby(['airline', 'airline_sentiment']).airline_sentiment.count().unstack()
airline_sentiment.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe41f83c160>
```



```
import seaborn as sns
sns.barplot(x='airline_sentiment', y='airline_sentiment_confidence' , data=airline_tweets)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe41f016e48>
```



```
features = airline_tweets.iloc[:, 10].values
labels = airline_tweets.iloc[:, 1].values
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
processed_features = []
for sentence in range(0, len(features)):
    # Remove all the special characters
    processed_feature = re.sub(r'\W', ' ', str(features[sentence]))
    # remove all single characters
    processed_feature= re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)
    # Remove single characters from the start
    processed_feature = re.sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)
    # Substituting multiple spaces with single space
    processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)
    # Removing prefixed 'b'
    processed_feature = re.sub(r'^b\s+', '', processed_feature)
    # Converting to Lowercase
    processed_feature = processed_feature.lower()
    processed_features.append(processed_feature)
vectorizer = TfidfVectorizer (max_features=2500, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state=0)
```

```
from sklearn.ensemble import RandomForestClassifier
text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

```
predictions = text_classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
!pip install --upgrade gensim
```

```
Collecting gensim
  Downloading https://files.pythonhosted.org/packages/d1/dd/112bd4258cee11e0baaba064060eb156475a42362e59e3ff28e7ca2d29d/gensim
     |████████████████████████████████| 24.2MB 147kB/s
Requirement already satisfied, skipping upgrade: scipy>=0.18.1 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.4.1)
Requirement already satisfied, skipping upgrade: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.17.5)
Requirement already satisfied, skipping upgrade: smart-open>=1.8.1 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.9
Requirement already satisfied, skipping upgrade: six>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.12.0)
Requirement already satisfied, skipping upgrade: boto3 in /usr/local/lib/python3.6/dist-packages (from smart-open>=1.8.1->gensi
Requirement already satisfied, skipping upgrade: boto>=2.32 in /usr/local/lib/python3.6/dist-packages (from smart-open>=1.8.1->
Requirement already satisfied, skipping upgrade: requests in /usr/local/lib/python3.6/dist-packages (from smart-open>=1.8.1->ge
Requirement already satisfied, skipping upgrade: s3transfer<0.4.0,>=0.3.0 in /usr/local/lib/python3.6/dist-packages (from boto3
Requirement already satisfied, skipping upgrade: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.6/dist-packages (from boto3->
Requirement already satisfied, skipping upgrade: botocore<1.15.0,>=1.14.14 in /usr/local/lib/python3.6/dist-packages (from boto
Requirement already satisfied, skipping upgrade: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->smart
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->s
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.6/dist-packages (from bo
Requirement already satisfied, skipping upgrade: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/dist-packages (from botocore<
Installing collected packages: gensim
  Found existing installation: gensim 3.6.0
    Uninstalling gensim-3.6.0:
      Successfully uninstalled gensim-3.6.0
Successfully installed gensim-3.8.1
```

```python
import pandas as pd
df = pd.read_csv('data.csv')
df.head()
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible

```
df['Maker_Model']= df['Make']+ " " + df['Model']
df.head()
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible

```
# Select features from original dataset to form a new dataframe
df1 = df[['Engine Fuel Type','Transmission Type','Driven_Wheels','Market Category','Vehicle Size', 'Vehicle Style', 'Maker_Model'
# For each row, combine all the columns into one column
df2 = df1.apply(lambda x: ','.join(x.astype(str)), axis=1)
# Store them in a pandas dataframe
df_clean = pd.DataFrame({'clean': df2})
# Create the list of list format of the custom corpus for gensim modeling
sent = [row.split(',') for row in df_clean['clean']]
# show the example of list of list format of the custom corpus for gensim modeling
sent[:2]
```

```
[['premium unleaded (required)',
  'MANUAL',
  'rear wheel drive',
  'Factory Tuner',
  'Luxury',
  'High-Performance',
  'Compact',
  'Coupe',
  'BMW 1 Series M'],
 ['premium unleaded (required)',
  'MANUAL',
  'rear wheel drive',
  'Luxury',
  'Performance',
  'Compact',
  'Convertible',
  'BMW 1 Series']]
```

```
from gensim.models import Word2Vec
model = Word2Vec(sent, min_count=1,size= 50,workers=3, window =3, sg = 1)
```

```
<gensim.models.word2vec.Word2Vec at 0x7f769bd08e48>
```

```
model['Toyota Camry']
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `__getitem__` (Method wi
```

```
vector = model.wv['Toyota Camry']
print(vector)
vector.size
```

```
[-0.08366955 -0.05964565  0.07203643 -0.04613186 -0.09290247 -0.1433193
  0.02276208 -0.12569097 -0.00580759 -0.16596414 -0.06013276  0.10451026
  0.24728008  0.00752487  0.17998965 -0.06541821 -0.1521897  -0.1394007
  0.11704888 -0.04366257 -0.09647141 -0.08375981 -0.08183265 -0.04778878
  0.08877282  0.08159016 -0.27344024  0.165536    0.08464111 -0.10411307
  0.35802853 -0.12511638  0.01041985  0.06723206 -0.08702769  0.01015439
 -0.00365196 -0.06646202  0.00494511 -0.00929783  0.08671172 -0.15059675
 -0.01291043 -0.22367048  0.16581357  0.08654065 -0.01826465  0.05042068
 -0.09550156  0.3554561 ]

50
```

```
!pip install wikipedia
```

```
Collecting wikipedia
  Downloading https://files.pythonhosted.org/packages/67/35/25e68fbc99e672127cc6fbb14b8ec1ba3dfef035bf1e4c90f78f24a80b7d/wikiped
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.6/dist-packages (from wikipedia) (4.6.3)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages (from wikipedia) (2.21.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests<3.0.0,>=2.0.0->wikipe
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests<3.0.0,>=2.0.0->wik
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests<3.0.0,>=2.0.0->wik
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests<3.0.0,>=2.0.0->wikipedia)
Building wheels for collected packages: wikipedia
  Building wheel for wikipedia (setup.py) ... done
  Created wheel for wikipedia: filename=wikipedia-1.4.0-cp36-none-any.whl size=11686 sha256=a34bcb0113f1c8c797bcb65cf61aa9d83a61
  Stored in directory: /root/.cache/pip/wheels/87/2a/18/4e471fd96d12114d16fe4a446d00c3b38fb9efcb744bd31f4a
Successfully built wikipedia
Installing collected packages: wikipedia
Successfully installed wikipedia-1.4.0
```

```
from keras.preprocessing.text import Tokenizer
from gensim.models.fasttext import FastText
import numpy as np
import matplotlib.pyplot as plt
import nltk
from string import punctuation
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize
from nltk import WordPunctTokenizer
import wikipedia
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
en_stop = set(nltk.corpus.stopwords.words('english'))
%matplotlib inline
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info.

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
artificial_intelligence = wikipedia.page("Artificial Intelligence").content
machine_learning = wikipedia.page("Machine Learning").content
deep_learning = wikipedia.page("Deep Learning").content
neural_network = wikipedia.page("Neural Network").content
artificial_intelligence = sent_tokenize(artificial_intelligence)
machine_learning = sent_tokenize(machine_learning)
deep_learning = sent_tokenize(deep_learning)
neural_network = sent_tokenize(neural_network)
artificial_intelligence.extend(machine_learning)
artificial_intelligence.extend(deep_learning)
artificial_intelligence.extend(neural_network)
```

```
import re
from nltk.stem import WordNetLemmatizer
stemmer = WordNetLemmatizer()
def preprocess_text(document):
        # Remove all the special characters
        document = re.sub(r'\W', ' ', str(document))
        # remove all single characters
        document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)
        # Remove single characters from the start
        document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)
        # Substituting multiple spaces with single space
        document = re.sub(r'\s+', ' ', document, flags=re.I)
        # Removing prefixed 'b'
        document = re.sub(r'^b\s+', '', document)
        # Converting to Lowercase
        document = document.lower()
        # Lemmatization
        tokens = document.split()
        tokens = [stemmer.lemmatize(word) for word in tokens]
        tokens = [word for word in tokens if word not in en_stop]
        tokens = [word for word in tokens if len(word) > 3]
        preprocessed_text = ' '.join(tokens)
        return preprocessed_text
```

```
sent = preprocess_text("Artificial intelligence, is the most advanced technology of the present era")
print(sent)
final_corpus = [preprocess_text(sentence) for sentence in artificial_intelligence if sentence.strip() !='']
word_punctuation_tokenizer = nltk.WordPunctTokenizer()
word_tokenized_corpus = [word_punctuation_tokenizer.tokenize(sent) for sent in final_corpus]
```

```
artificial intelligence advanced technology present
```

```
embedding_size = 60
window_size = 40
min_word = 5
down_sampling = 1e-2
```

```
%%time
ft_model = FastText(word_tokenized_corpus,
                      size=embedding_size,
                      window=window_size,
                      min_count=min_word,
                      sample=down_sampling,
                      sg=1,
                      iter=100)
```

```
CPU times: user 1min 39s, sys: 318 ms, total: 1min 40s
Wall time: 51.3 s
```

```
print(ft_model.wv['artificial'])
```

```
[-0.30679983  0.14952238 -0.21810903 -0.7395383  -0.23074621  0.10807946
  0.04915254  0.08504375  0.0450432  -0.04488863 -0.15089108 -0.31847388
  0.30588818 -0.19040418  0.05474224  0.05302594 -0.10397033  0.3067033
  0.09131836  0.03521951  0.46956265  0.11327234  0.26022822 -0.03658391
 -0.2489354  -0.22652984  0.2772911  -0.23342757  0.6781643   0.08554471
 -0.05705503  0.09560576 -0.2070933   0.01806752 -0.42353478 -0.44353613
  0.06448416  0.5410614   0.08621874  0.05771734  0.08044741  0.14682104
  0.0913202  -0.12521906  0.2854835  -0.30264926  0.04221692 -0.14606497
 -0.13468763  0.21106029 -0.2290192  -0.45784584 -0.289197   -0.16205801
 -0.645687    0.02343115  0.2587803  -0.24801745 -0.23193733  0.03140956]
```

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
%matplotlib inline
```

```
train = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/train.csv')
train_original=train.copy()
train_original.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

```
test = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/test.csv')
test_original=test.copy()
test_original.head()
```

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

```
combine = train.append(test,ignore_index=True,sort=True)
combine.head()
```

	id	label	tweet
0	1	0.0	@user when a father is dysfunctional and is s...
1	2	0.0	@user @user thanks for #lyft credit i can't us...
2	3	0.0	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation

```
combine.tail()
```

```
def remove_pattern(text,pattern):
    # re.findall() finds the pattern i.e @user and puts it in a list for further task
    r = re.findall(pattern,text)
    # re.sub() removes @user from the sentences in the dataset
    for i in r:
        text = re.sub(i,"",text)
    return text

combine['Tidy_Tweets'] = np.vectorize(remove_pattern)(combine['tweet'], "@[\w]*")
combine.head()
```

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation	factsguide: society now #motivation

```
combine['Tidy_Tweets'] = combine['Tidy_Tweets'].str.replace("[^a-zA-Z#]", " ")
combine.head(10)
```

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can t use cause th...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation	factsguide society now #motivation
5	6	0.0	[2/2] huge fan fare and big talking before the...	huge fan fare and big talking before the...
6	7	0.0	@user camping tomorrow @user @user @user @use...	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð□□...	the next school year is the year for exams ...
8	9	0.0	we won!!! love the land!!! #allin #cavs #champ...	we won love the land #allin #cavs #champ...
9	10	0.0	@user @user welcome here ! i'm it's so #gr...	welcome here i m it s so #gr

```
combine['Tidy_Tweets'] = combine['Tidy_Tweets'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
combine.head(10)
```

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation
5	6	0.0	[2/2] huge fan fare and big talking before the...	huge fare talking before they leave chaos disp...
6	7	0.0	@user camping tomorrow @user @user @user @use...	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð□□...	next school year year exams think about that #...
8	9	0.0	we won!!! love the land!!! #allin #cavs #champ...	love land #allin #cavs #champions #cleveland #...

```
tokenized_tweet = combine['Tidy_Tweets'].apply(lambda x: x.split())
tokenized_tweet.head()
```

```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                              [bihday, your, majesty]
3                        [#model, love, take, with, time]
4                      [factsguide, society, #motivation]
Name: Tidy_Tweets, dtype: object
```

```
from nltk import PorterStemmer
ps = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda x: [ps.stem(i) for i in x])
tokenized_tweet.head()
```

```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                              [bihday, your, majesti]
3                     [#model, love, take, with, time]
4                         [factsguid, societi, #motiv]
Name: Tidy_Tweets, dtype: object
```

```
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
combine['Tidy_Tweets'] = tokenized_tweet
combine.head()
```

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunct selfish drag kid into dys...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit caus they offer wheelchair ...
2	3	0.0	bihday your majesty	bihday your majesti
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguid societi #motiv

```
from wordcloud import WordCloud,ImageColorGenerator
from PIL import Image
import urllib
import requests
```

```
all_words_positive = ' '.join(text for text in combine['Tidy_Tweets'][combine['label']==0])
```

```
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-PNG-Image.png', stream=True).raw))
# We use the ImageColorGenerator library from Wordcloud
# Here we take the color of the image and impose it over our wordcloud
image_colors = ImageColorGenerator(Mask)
# Now we use the WordCloud function from the wordcloud library
wc = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(all_words_positive)
```

```
# Size of the image generated
plt.figure(figsize=(10,20))
# Here we recolor the words from the dataset to the image's color
# recolor just recolors the default colors to the image's blue color
# interpolation is used to smooth the image generated
plt.imshow(wc.recolor(color_func=image_colors),interpolation="hamming")
plt.axis('off')
plt.show()
```

love
thi
happi
today
thank
life
wait
smile
peopl
time
feel
bihday
friend
make
beauti
good
girl
want
comego
look
famili
thank
posit
tomorrow
weekend
posit affirm

1

```
# Run in python console
import nltk; nltk.download('stopwords')
# Run in terminal or command prompt
!python3 -m spacy download en
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Requirement already satisfied: en_core_web_sm==2.1.0 from https://github.com/explosion/spacy-models/releases/download/en_core_w
✔ Download and installation successful
You can now load the model via spacy.load('en_core_web_sm')
✔ Linking successful
/usr/local/lib/python3.6/dist-packages/en_core_web_sm -->
/usr/local/lib/python3.6/dist-packages/spacy/data/en
You can now load the model via spacy.load('en')
```

```
import re
import numpy as np
import pandas as pd
from pprint import pprint

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy for lemmatization
import spacy

# Plotting tools
!pip install pyLDAvis
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline

```

```
# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.ERROR)

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

```
Collecting pyLDAvis
  Downloading https://files.pythonhosted.org/packages/a5/3a/af82e070a8a96e13217c8f362f9a73e82d61ac8fff3a2561946a97f96266/pyLDAv
     |████████████████████████████████| 1.6MB 3.3MB/s
Requirement already satisfied: wheel>=0.23.0 in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (0.34.2)
Requirement already satisfied: numpy>=1.9.2 in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (1.17.5)
Requirement already satisfied: scipy>=0.18.0 in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (1.4.1)
Requirement already satisfied: pandas>=0.17.0 in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (0.25.3)
Requirement already satisfied: joblib>=0.8.4 in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (0.14.1)
Requirement already satisfied: jinja2>=2.7.2 in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (2.11.1)
Requirement already satisfied: numexpr in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (2.7.1)
Requirement already satisfied: pytest in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (3.6.4)
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from pyLDAvis) (0.16.0)
Collecting funcy
  Downloading https://files.pythonhosted.org/packages/ce/4b/6ffa76544e46614123de31574ad95758c421aae391a1764921b8a81e1eae/funcy-
     |████████████████████████████████| 552kB 24.5MB/s
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.17.0->pyLDAvis) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.17.0->pyLDAvis)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2>=2.7.2->pyLDAvis) (1.1.1
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (45.1.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (1.12.0)
Requirement already satisfied: pluggy<0.8,>=0.5 in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (0.7.1)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (19.3.0)
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (1.8.1)
Requirement already satisfied: atomicwrites>=1.0 in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (1.3.0)
Requirement already satisfied: more-itertools>=4.0.0 in /usr/local/lib/python3.6/dist-packages (from pytest->pyLDAvis) (8.2.0)
Building wheels for collected packages: pyLDAvis, funcy
  Building wheel for pyLDAvis (setup.py) ... done
  Created wheel for pyLDAvis: filename=pyLDAvis-2.1.2-py2.py3-none-any.whl size=97711 sha256=297333871d3eaf0d9dc6f5dd30bfb55fe2
  Stored in directory: /root/.cache/pip/wheels/98/71/24/513a99e58bb6b8465bae4d2d5e9dba8f0bef8179e3051ac414
  Building wheel for funcy (setup.py) ... done
  Created wheel for funcy: filename=funcy-1.14-py2.py3-none-any.whl size=32042 sha256=8d838cfe3a79afd34eb7de3e3318f2482120940be
  Stored in directory: /root/.cache/pip/wheels/20/5a/d8/1d875df03deae6f178dfdf70238cca33f948ef8a6f5209f2eb
Successfully built pyLDAvis funcy
Installing collected packages: funcy, pyLDAvis
Successfully installed funcy-1.14 pyLDAvis-2.1.2
```

```python
# NLTK Stop words
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

```
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
```

```
# Import Dataset
df = pd.read_json('https://raw.githubusercontent.com/selva86/datasets/master/newsgroups.json')
print(df.target_names.unique())
df.head()
```

```
['rec.autos' 'comp.sys.mac.hardware' 'comp.graphics' 'sci.space'
 'talk.politics.guns' 'sci.med' 'comp.sys.ibm.pc.hardware'
 'comp.os.ms-windows.misc' 'rec.motorcycles' 'talk.religion.misc'
 'misc.forsale' 'alt.atheism' 'sci.electronics' 'comp.windows.x'
 'rec.sport.hockey' 'rec.sport.baseball' 'soc.religion.christian'
 'talk.politics.mideast' 'talk.politics.misc' 'sci.crypt']
```

	content	target	target_names
0	From: lerxst@wam.umd.edu (where's my thing)\nS...	7	rec.autos
1	From: guykuo@carson.u.washington.edu (Guy Kuo)...	4	comp.sys.mac.hardware
2	From: twillis@ec.ecn.purdue.edu (Thomas E Will...	4	comp.sys.mac.hardware
3	From: jgreen@amber (Joe Green)\nSubject: Re: W...	1	comp.graphics
4	From: jcm@head-cfa.harvard.edu (Jonathan McDow...	14	sci.space

```
# Convert to list
data = df.content.values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]

pprint(data[:1])
```

```
['From: (wheres my thing) Subject: WHAT car is this!? Nntp-Posting-Host: '
 'rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: '
 '15 I was wondering if anyone out there could enlighten me on this car I saw '
 'the other day. It was a 2-door sports car, looked to be from the late 60s/ '
 'early 70s. It was called a Bricklin. The doors were really small. In '
 'addition, the front bumper was separate from the rest of the body. This is '
 'all I know. If anyone can tellme a model name, engine specs, years of '
 'production, where this car is made, history, or whatever info you have on '
 'this funky looking car, please e-mail. Thanks, - IL ---- brought to you by '
 'your neighborhood Lerxst ---------']
```

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations

data_words = list(sent_to_words(data))

print(data_words[:1])
```

```
[['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nntp', 'posting', 'host', 'rac', 'wam', 'umd', 'edu
```

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
```

```
/usr/local/lib/python3.6/dist-packages/gensim/models/phrases.py:598: UserWarning: For a faster implementation, use the gensim.m
```

```
# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out
```

```
# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)

# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
nlp = spacy.load('en', disable=['parser', 'ner'])

# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])

print(data_lemmatized[:1])
```

```
[['where', 's', 'thing', 'car', 'nntp_poste', 'host', 'umd', 'organization', 'university', 'maryland_college', 'park', 'line',
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
```

```
[[(0, 1), (1, 2), (2, 1), (3, 1), (4, 1), (5, 1), (6, 5), (7, 1), (8, 1), (9, 2), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1),
```

```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

```
[[('addition', 1),
  ('anyone', 2),
  ('body', 1),
  ('bricklin', 1),
  ('bring', 1),
  ('call', 1),
  ('car', 5),
  ('could', 1),
  ('day', 1),
  ('door', 2),
  ('early', 1),
  ('engine', 1),
  ('enlighten', 1),
  ('front_bumper', 1),
  ('funky', 1),
  ('history', 1),
  ('host', 1),
  ('info', 1),
  ('know', 1),
  ('late', 1),
  ('lerxst', 1),
  ('line', 1),
  ('look', 2),
  ('mail', 1),
  ('make', 1),
  ('maryland_college', 1),
  ('model', 1),
  ('name', 1),
  ('neighborhood', 1),
  ('nntp_poste', 1),
  ('organization', 1),
  ('park', 1),
  ('production', 1),
  ('really', 1),
  ('rest', 1),
  ('s', 1),
  ('see', 1),
  ('separate', 1),
  ('small', 1),
  ('specs', 1),
  ('sport', 1),
  ('tellme', 1),
  ('thank', 1),
  ('thing', 1),
  ('umd', 1),
  ('university', 1),
  ('where', 1),
  ('wonder', 1),
  ('year', 1)]]
```

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=20,
                                           random_state=100,
                                           update_every=1,
                                           chunksize=100,
                                           passes=10,
                                           alpha='auto',
                                           per_word_topics=True)
```

```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
  '0.015*"pin" + 0.015*"slave" + 0.014*"sphere" + 0.012*"character" + '
  '0.010*"lord" + 0.009*"headache"'),
 (2,
  '0.015*"choose" + 0.012*"input" + 0.011*"sin" + 0.010*"notice" + 0.009*"eat" '
  '+ 0.009*"cd" + 0.009*"food" + 0.009*"material" + 0.008*"signal" + '
  '0.008*"external"'),
 (3,
  '0.035*"not" + 0.022*"write" + 0.022*"do" + 0.020*"would" + 0.020*"line" + '
  '0.019*"organization" + 0.017*"be" + 0.017*"article" + 0.014*"get" + '
  '0.014*"know"'),
 (4,
  '0.533*"ax" + 0.008*"rlk" + 0.006*"cub" + 0.005*"echo" + '
  '0.004*"tufts_university" + 0.004*"stl" + 0.004*"pitcher" + 0.004*"pit" + '
  '0.004*"lk" + 0.003*"differential"'),
 (5,
  '0.027*"israel" + 0.015*"israeli" + 0.011*"jew" + 0.009*"lebanese" + '
  '0.009*"arab" + 0.009*"jewish" + 0.009*"war" + 0.008*"death" + 0.008*"kill" '
  '+ 0.007*"attack"'),
 (6,
  '0.030*"drive" + 0.018*"card" + 0.014*"mac" + 0.013*"driver" + '
  '0.012*"system" + 0.011*"cpu" + 0.009*"memory" + 0.009*"computer" + '
  '0.009*"chip" + 0.009*"use"'),
 (7,
  '0.053*"_" + 0.045*"max" + 0.012*"dn" + 0.010*"eeg" + 0.009*"cx" + 0.007*"c" '
  '+ 0.007*"mv" + 0.005*"mk" + 0.005*"sw" + 0.004*"mj"'),
 (8,
  '0.015*"library" + 0.015*"section" + 0.013*"st" + 0.011*"ed" + 0.009*"title" '
  '+ 0.009*"art" + 0.009*"author" + 0.009*"pa" + 0.009*"translation" + '
  '0.009*"page"'),
 (9,
  '0.017*"car" + 0.011*"new" + 0.009*"buy" + 0.009*"physical" + 0.008*"power" '
  '+ 0.008*"type" + 0.007*"old" + 0.007*"graphic" + 0.007*"screen" + '
  '0.007*"good"'),
 (10,
  '0.035*"god" + 0.021*"evidence" + 0.017*"christian" + 0.015*"reason" + '
  '0.015*"believe" + 0.012*"say" + 0.011*"faith" + 0.010*"claim" + '
  '0.010*"exist" + 0.010*"sense"'),
 (11,
  '0.019*"university" + 0.017*"organization" + 0.015*"line" + 0.014*"instal" + '
  '0.011*"michael" + 0.010*"format" + 0.010*"package" + 0.010*"problem" + '
  '0.009*"distribution" + 0.009*"robert"'),
 (12,
  '0.018*"pay" + 0.015*"item" + 0.014*"service" + 0.012*"cover" + 0.012*"cost" '
  '+ 0.011*"sell" + 0.010*"recommend" + 0.010*"replace" + 0.009*"gateway" + '
  '0.009*"air"'),
 (13,
  '0.019*"line" + 0.017*"window" + 0.016*"mail" + 0.016*"file" + 0.016*"thank" '
  '+ 0.015*"program" + 0.013*"use" + 0.011*"organization" + 0.011*"system" + '
  '0.009*"email"'),
 (14,
  '0.019*"state" + 0.012*"law" + 0.012*"issue" + 0.011*"right" + 0.010*"case" '
  '+ 0.008*"group" + 0.006*"new" + 0.005*"people" + 0.005*"national" + '
  '0.005*"support"'),
 (15,
  '0.025*"internet" + 0.020*"bike" + 0.017*"server" + 0.014*"md" + 0.013*"com" '
  '+ 0.012*"engine" + 0.011*"ride" + 0.011*"steve" + 0.011*"pain" + '
  '0.010*"route"'),
 (16,
  '0.019*"gun" + 0.010*"kill" + 0.010*"people" + 0.009*"armenian" + '
  '0.008*"say" + 0.008*"fire" + 0.008*"child" + 0.007*"greek" + '
  '0.006*"government" + 0.006*"american"'),
 (17,
  '0.020*"win" + 0.016*"year" + 0.014*"player" + 0.013*"university" + '
  '0.012*"patient" + 0.009*"fan" + 0.008*"run" + 0.008*"drug" + 0.007*"score" '
  '+ 0.007*"mouse"'),
 (18,
  '0.027*"space" + 0.012*"research" + 0.009*"faq" + 0.008*"earth" + '
  '0.008*"mount" + 0.007*"science" + 0.007*"launch" + 0.006*"project" + '
  '0.006*"moon" + 0.006*"datum"'),
 (19,
  '0.041*"key" + 0.014*"system" + 0.014*"ripem" + 0.013*"government" + '
  '0.013*"public" + 0.012*"security" + 0.012*"encryption" + 0.010*"tape" + '
  '0.009*"chip" + 0.009*"clipper"')]
```

```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

# Compute Coherence Score
```

```python
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```

Perplexity:  -8.732797516655772

Coherence Score:  0.5017680246997409
```