Problem statement

The objective of this project is to apply sentiment analysis, the most common text classification tool, to analyse an incoming message and tell whether the underlying sentiment is positive, negative or neutral. The dataset contains tweets on US Airline of February 2015 classified in positive, negative and neutral tweets. The negative tweets are also classified by the negative reason.

The first challenge is that the dataset is very unstructured. Before we do data visualization and model building, we need to do some data pre-processing or wrangling to remove unnecessary characters, symbols and tokens. The definition of unnecessary information actually depends on which machine learning techniques to use. In this project, we will apply both traditional and advanced techniques to build sentiment analysis models. For traditional ones, we will try to clean the unnecessary information as much as possible since we convert the documents into individual tokens without considering the context. By contrast, we need to keep import contextual connection between words if word2vec or deep learning model is built.

Once the model is put into implementation, the business owner is able to predict whether the customers' feedback is positive, negative or neutral. This can help identify potential issues in the process of corporate operations and improve the customers' satisfaction.

Data Pre-processing

There can be multiple ways of cleaning and pre-processing textual data. In this project, we take some of the most popular approaches which are used heavily in Natural Language Processing (NLP) pipelines.

1. Tweet preprocessor:

Tweet preprocessor is a preprocessing library for tweet data written in Python. When building machine learning systems based on tweet data, a preprocessing is required. This library makes it easy to clean, parse or tokenize the tweets. It supports cleaning, tokenizing and parsing: URLs, Hashtags, Mentions, Reserved words (RT,FAV),Emojis and Smileys.Reserved words (RT, FAV). For example, the original content of the first tweet is **@VirginAmerica What @dhepburn said.** After this step, it was converted into "What said".

2. Expanding contractions:

In the English language, contractions are basically shortened versions of words or s yllables. These shortened versions of existing words or phrases are created by rem oving specific letters and sounds. Converting each contraction to its expanded, orig inal form often helps with text standardization. For example, the content of third row is "`@VirginAmerica I didn't today... Must mean I need to take another t rip!`. The step converted "didn't " into "did not".

3. Removing special characters:

   Special characters and symbols which are usually non alphanumeric characters often add to the extra noise in unstructured text. More than often, simple regular expressions (regexes) can be used to achieve this. In this project, we use the regular expression '[^a-zA-z0-9\s]'.

4. Stemming and lemmatization:

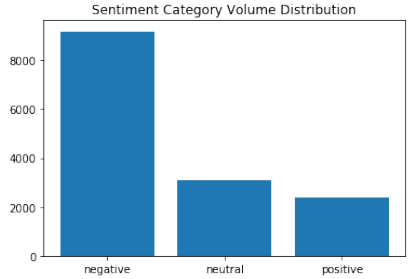
   Word stems are usually the base form of possible words that can be created by attaching *affixes* like *prefixes* and *suffixes* to the stem to create new words. This is known as inflection. The reverse process of obtaining the base form of a word is known as *stemming*. For example, the words "said" have been converted into "say".

5. Removing stopwords:

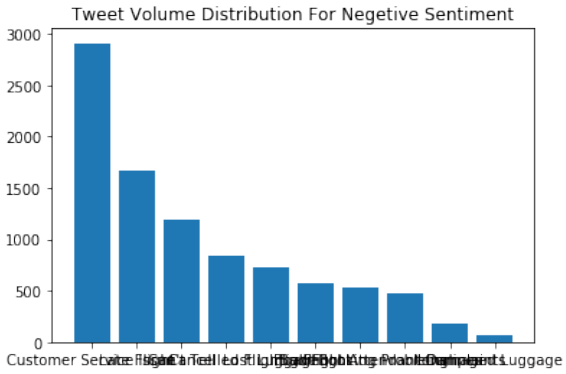
   Words which have little or no significance especially when constructing meaningful features from text are known as stopwords or stop words. These are usually words that end up having the maximum frequency if we do a simple term or word frequency in a corpus. We use package NLTK to create a stopwords list, which contains 17 words.

Exploratory Data Analysis

The first talk for exploratory data analysis is to look at how the volume of different target variables is distributed. We use function .value_counts() to achieve this.

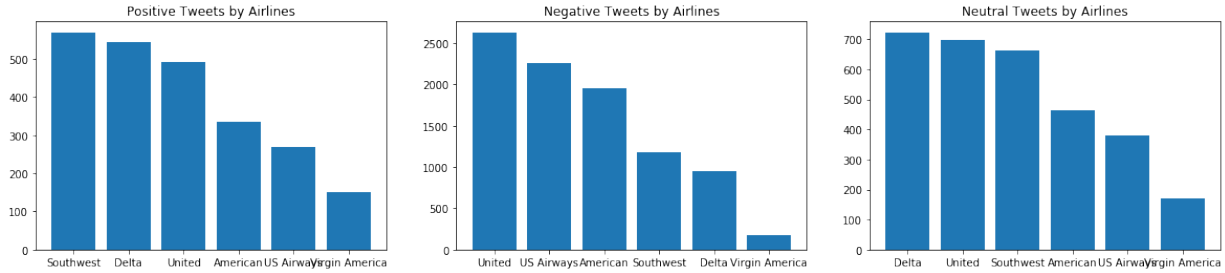


From this chart, we can see this dataset doesn't have an issue of 'imbalance". Another topic we are interested to see is how the negative reasons are distributed for "negative" sentiment group.



We noticed that the top 3 reasons are Customer Service, Late Flight and Can't tell.

This dataset contains the tweets provided by six airlines: Delta, United, Southwest, American, US Airways and Virgin America. To make a comparison between these operators, we will look at how the number of tweets distributed under different sentiment categories for all these six airlines. The chart below indicates that the Southwest had the best performance in terms of positive tweets; United had the worst performance measured by negative tweets.



Unlike other machine learning tasks, Natural Language Processing mainly focuses on textual data. This means we are not going to take the traditional approaches to visualize the data. The package NLTK can help us calculate and view the frequency of each word in the corpus. The size of vocabulary is 20,127. The 10 most frequentl y occurred words are `'@', '.', 'to', 'I', 'the', '!', '?', 'a', ',', 'for'`. They are either stop words or special words, which will be removed in the stage of text preprocessing. The package WordCloud enables us to visualize the frequency occurrence for each word.

Text data encoding

Text data requires special preparation before using it for predictive modeling. The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm. A simple and effective model for thinking about text documents in machine learning is called the Bag-of-Words Model, or BoW. The model is simple in that it throws away all of the order information in the words and focuses on the occurrence of words in a document.

The scikit-learn library provides 2 different schemes that we can use:

Word Counts with CountVectorizer

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. For this project, we created a data frame with 6875 columns.

	00	000	000419	00a	00am	00p	00pm	0200	03	04	...	zagg	zambia	zcc82u	zero	zig	zip	zipper	zone	zoom	zurich
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 6875 columns

Word Frequencies with TfidfVectorizer

One issue with simple counts is that some words will appear many times and their large counts will not be very meaningful in the encoded vectors. An alternative is to calculate word frequencies, and by far the most popular method is called TF-IDF. This stands for "*Term Frequency – Inverse Document*" Frequency which are the components of the resulting scores assigned to each word.

	00	000	000419	00a	00am	00p	00pm	0200	03	04	...	zagg	zambia	zcc82u	zero	zig	zip	zipper	zone	zoom	zurich
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 6875 columns

Traditional Machine Learning Model Building

So far, we have completed the data visualization, data preprocessing and text date encoding. As a result, we created two data frames for CountVectorization and TF-IDF respectively. Both of these data frames contain numerical values only. We can use traditional classification machine learning techniques to build predictive models: Naïve Bayes multinomial classification and Support Vector classification techniques.

- CountVectorization

Naïve Bayes multinomial classification

Accuracy: **0.7491**

Support Vector classification

Accuracy: **0.7692**

- TF-IDF

Naïve Bayes multinomial classification

Accuracy: **0.6841**

Support Vector classification

Accuracy: **0.7651**

TD-IDF has two important hyper-parameters: max_df and ngram_range. When building the vocabulary, the hyper-parameter max_df ignores terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float in range [0.0, 1.0], the parameter represents a proportion of documents, integer absolute counts. The parameter ngram_range defines the lower and upper boundary of the range of n-values for different n-grams to be extracted. SVC has also an important regularization parameter C. The strength of the regularization is inversely proportional to C. We created a new pipeline by combining TF-IDF and SVC.

Grid search is the process of performing hyper-parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper-parameter values specified. We applied the method GridSearchCV on the built pipeline and found the optimal values of hyper-parameter as below:

```
{'model__C': 10, 'tfidf__max_df': 0.5, 'tfidf__ngram_range': (1, 1)}
```

The accuracy score we achieved is 0.7719.