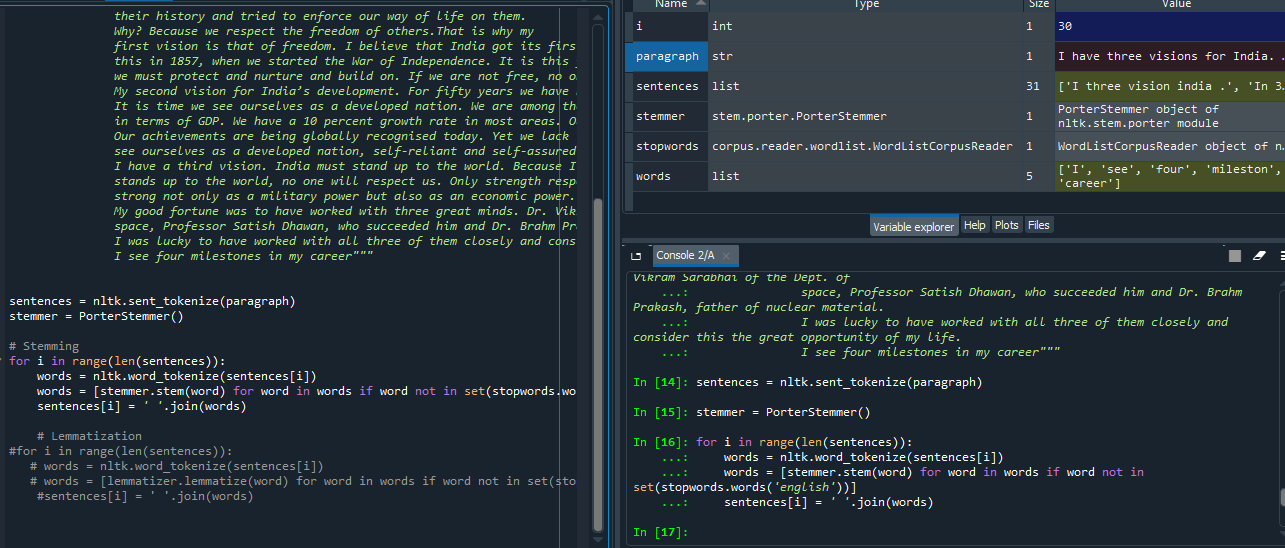
**Techniques:**

**NLTK(Natural Language Toolkit)** is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to  [lexical resources](http://nltk.org/nltk_data/) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

**Text Pre- Processing with NLTK**

The main issue with text data is that it is all in text format (strings). However, Machine learning algorithms need some sort of numerical feature vector in order to perform the task. So before we start with any NLP project we need to pre-process it to make it ideal for work. Basic **text pre-processing** includes:

* Converting the entire text into **uppercase or lowercase**, so that the algorithm does not treat the same words in different cases as different
* **Tokenization**: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.



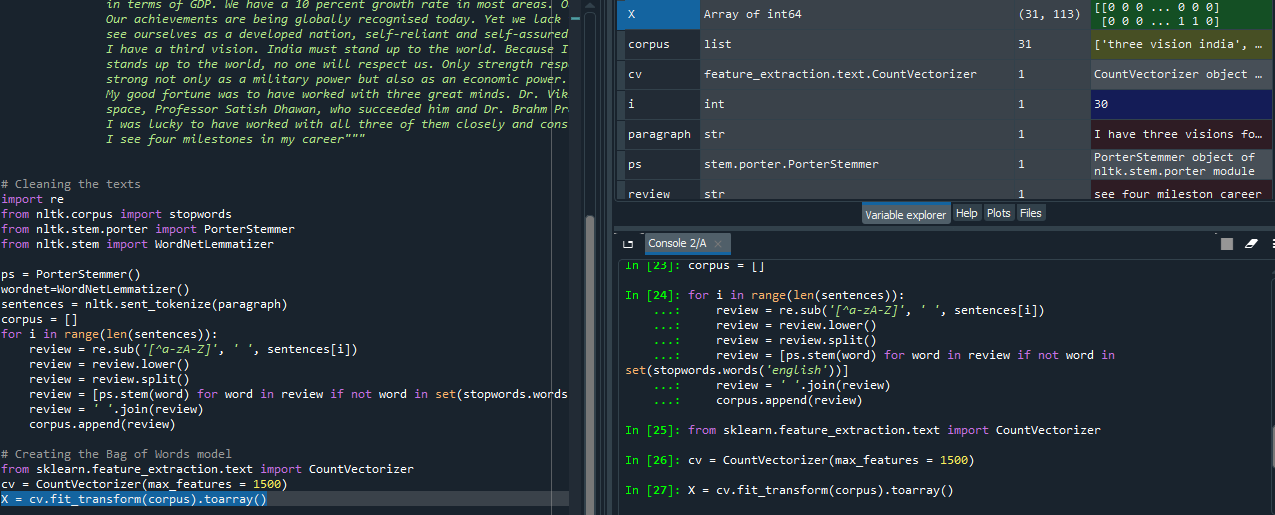
* Removing **Stop words.**Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called *stop words*
* **Stemming**: Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form.
* **Lemmatization**: A slight variant of stemming is lemmatization. The major difference between these is, that, stemming can often create non-existent words, whereas lemmas are actual words. So, your root stem, meaning the word you end up with, is not something you can just look up in a dictionary, but you can look up a lemma.

## Bag of Words

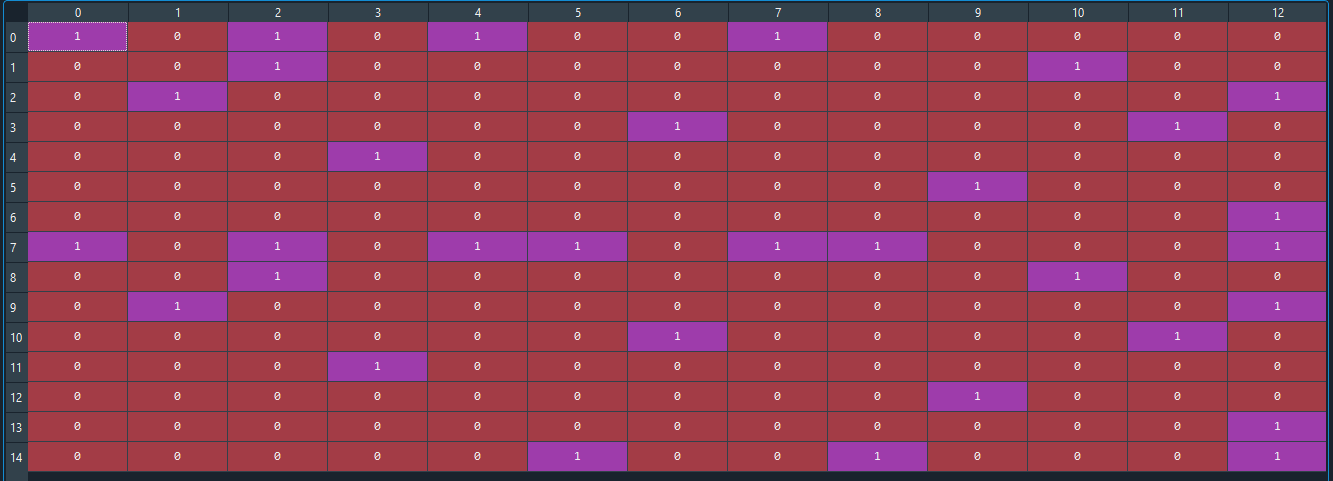
After the initial preprocessing phase, we need to transform the text into a meaningful vector (or array) of numbers. The bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

•A vocabulary of known words.

•A measure of the presence of known words.



**Output:**

****

**TF-IDF Approach:**

A problem with the Bag of Words approach is that highly frequent words start to dominate in the document, but may not contain as much “informational content”. Also, it will give more weight to longer documents than shorter documents.

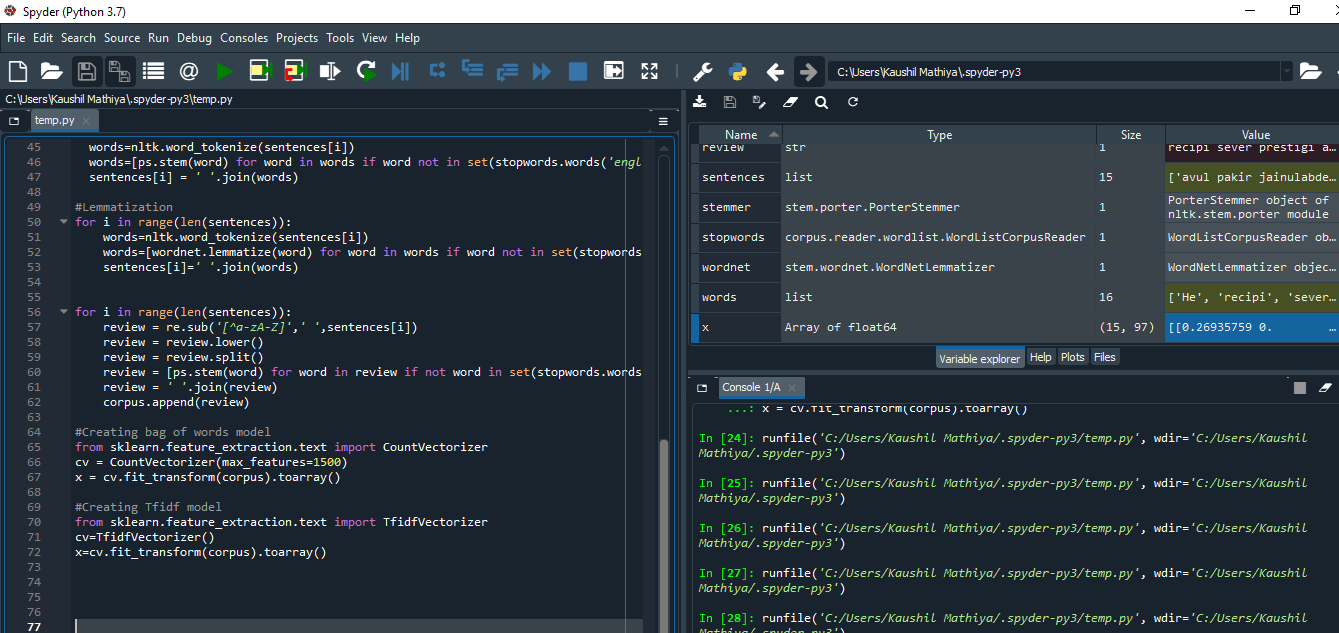
One approach is to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words like “the” that are also frequent across all documents are penalized. This approach is knowns as Term Frequency-Inverse Document Frequency, or TF-IDF.

Term Frequency is a scoring of the frequency of the word in the current document.

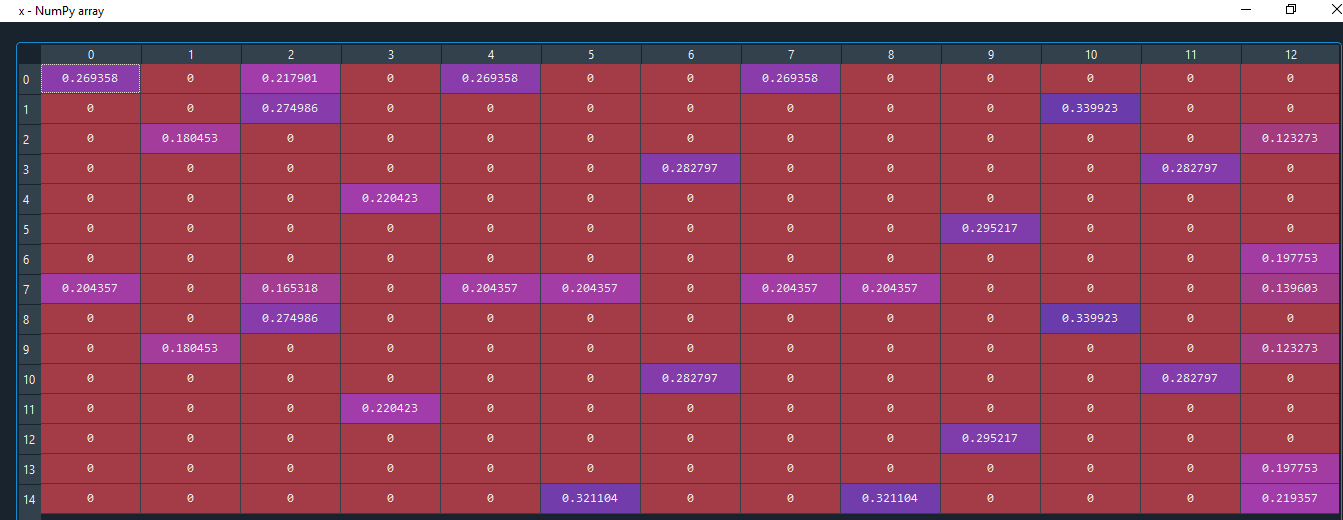
**TF= (Number of times term t appears in a document)/(Number of terms in the document)**

Inverse Document Frequency is a scoring of how rare the word is across document.

**IDF= 1+log(N/n) ,** where N is the number of documents and n is the number of documents a term t has appeared in

****

**Output:**

****

**Source:**

* <https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e>