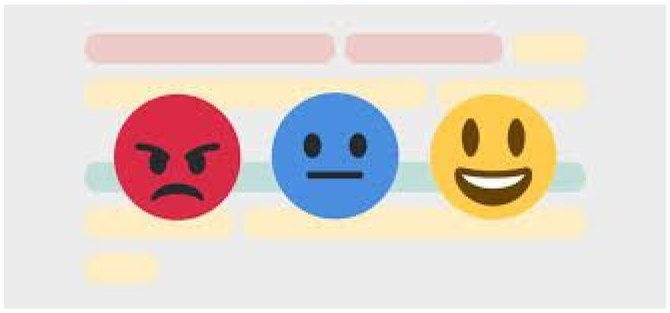
NLP + OCR Task - Building a simple classifier model for text data

Hello everyone, so the aim of this task is to build a simple classifier model on Text to classify sentences based on their sentiment, **either positive or negative**. We hope this task will give you some basic insights into the domain of NLP and help you proceed with this project.



So first things first, what is the data like - basically text, to be specific **IMBD reviews** with their corresponding sentiment. The dataset is well balanced with equal proportions of positive and negative reviews. We will be using the **nltk - natural language toolkit library** in python to help with our coding as there is a lot of functions in the library to help reduce a lot of redundant code.

A basic outline of what we expect you to do:

- **Explore the dataset** - Have a look at the dataset and understand what has to be done. Since these are reviews written by people there is a lot of text slang like LOL etc. Every sentence is separated using a < br/>. There are a few capitalized words, etc. **So first step is to make a not of what's there on the data and plan ideas to make the best use of this.**

- Text can be assumed as a sequence of characters, words, phrases, sentences or paragraphs.It might be commonly understood that **text is a sequence of words.** So the best way to represent a sentence is by using words. In English, words are separated by punctuations or spaces. **The process of splitting a text into chunks of our choice (in this case words) is called tokenization.** The nltk library contains some easy to use functions for the above task.

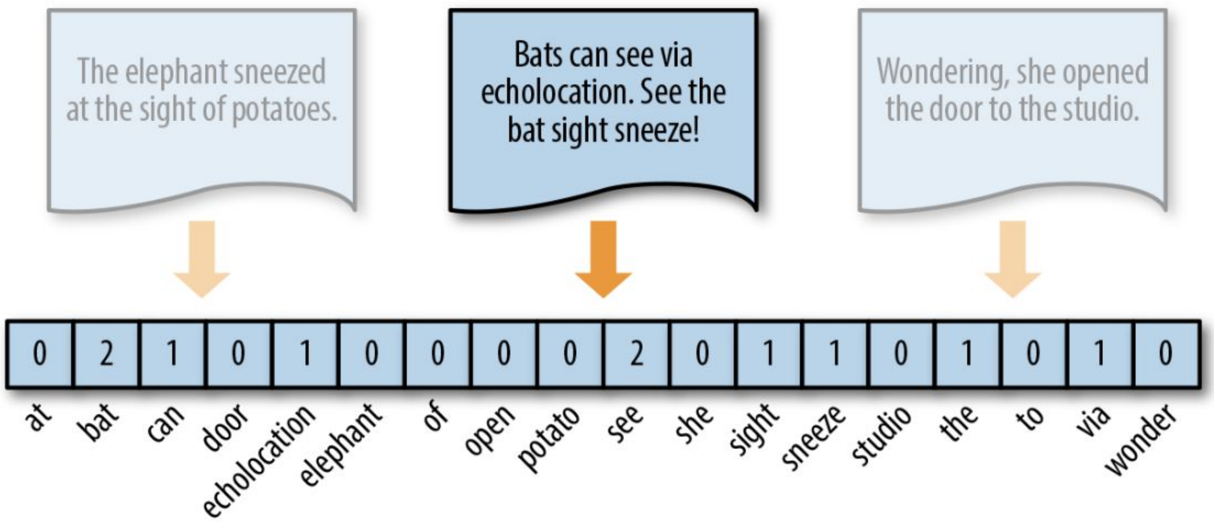
  For example:
  - 1) *nltk.tokenize.WhitespaceTokeniser* splits a sentence on white spaces.
  - 2) *nltk.tokenize.WordPunctTokenizer* splits a sentence on punctuations too.

  Now, this isn't the most ideal way of doing this. Lets say we have the word - "doesn't" splitting this word into "doesn" and "t" makes no sense. Hence there must be some rules as well for deciding on this splitting. **There are functions for the same as well in the nltk library and I am not going to mention it and its for you to look and explore.**

- Passing text to numbers isn't going to make much sense for the machine. Hence the simplest way to go about this to convert text into numbers (which are easily understood by machines). **Since we have chosen a word as the simplest unit in a sentence, we will be representing every word by a number and hence a sentence becomes a collection of numbers (a vector).**

  Now wait, suppose we have "children" and "child" as two separate words won't they be assigned different numbers but they refer to the same thing. **Now the process of removing and replacing suffixes in words so that they can be grouped together is called stemming.** A better way to counter this issue is to use **Lemmatisation** where the meaning of the word is also taken into consideration and once again there are functions for the same in the nltk library.

- As you must have already guessed by now, we will be using a machine learning model to predict the sentiment given a text. Now simple machine learning models require inputs and outputs of fixed lengths. (we will be doing something simple only). As you would have understood we need to convert a sentence to a vector, to be specific **a fixed size vector** to represent a sentence. Now one of the simplest ways to do this is using the bag of words technique. **A bag-of-words** is a representation of text that describes the occurrence of words within a document. In this method, a vocabulary is created and vectors are created based on this vocabulary where a number at a position corresponds to the number of occurrences of that particular word in the sentence. Now simply creating a vocabulary and creating a vector-based on this is not going to contain information regarding the order of words. One simple way to add this information is by using the n-gram technique. (Leaving it here - Make sure to read about it)


The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
0 2 1 0 1 0 0 0 0 2 0 1 1 0 1 0 1 0
at
bat
can
door
echolocation
elephant
of
open
potato
see
she
sight
sneeze
studio
the
to
via
wonder

- Now that we have our sentence vectors, we need to build a function to map these vectors to its corresponding sentiment. Now, it’s not possible for us to manually create this function and hence we use a simple linear model called logistic regression for the same. To give you a gist of what it means: Basically every vector is multiplied with some weights (randomly initialized at first) and its assumed that the output of this multiplication is the sentiment itself **(forward propagation)**. Now it’s not going to be true first and hence based on the error the weights are going to be fine-tuned till the error is minimum **(backpropagation)**. know this looks too simple to work but there is some simple math involved in this and I don’t want to bore you with the same. **Check out these links for the same:** **https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html - start from here and move onto logistic regression.**

In case all of you face issues with understanding this we can take it up as a small quick session as well. Similarly for this, there are functions on the **sklearn library** of python to perform logistic regression which you can use for this task.

Some tips and tricks :

- Try to look at the documentation of nltk and sklearn cause most of the operations can be done automatically using functions and doesn’t require you to manually code a lot for the same.
- Choosing how you wish to tokenize the text is very important, whether or not you want to use exclamations, differentiate between all caps words, etc cause each method will carry some specific information.
- Also, give a thought if all the words are to be considered to build the vocabulary and is it better if we removed some.

Bonus:

Have a look at some more complex models or algorithms for the same task. You don’t need to necessarily implement them but reading through them and mentioning suitable improvements for your solution will be sufficient.