Sentiment Analysis of Movie Reviews

What is Sentiment Analysis ?

Sentiment Analysis helps a computer understand the underlying tone of a subjective piece of writing.It helps us in better understanding of content.

This Implementation using Natural Language Processing to Preprocess and Clean Text Data

Theory

First Step is loading up the data and then process it through a Natural Learning Processing pipeline.It includes a lot of steps:

- Tokenizing Sentences: to break it down into words or other units.
- Removing Stop Words: like “if”,”but”,”or”and so on.
- Normalizing words to root form.
- Vectorizing: changing the word into numbers for the classifier to be able to classify them.

All of this helps in reducing noise inherent in any human speech or sentence and improves accuracy of the classifier.There are lots of great tools to help with this, such as the Natural Language Toolkit, TextBlob, and spaCy.

- **Tokenization:** As already elucidated tokenization helps to break into smaller parts.There are two ways to do that:
    - **Word Tokenization:** Breaking down words into individual letters.
    - **Sentence Tokenization:** Breaking Sentences into words.
    - #Comment 1 in the Notebook shows Tokenization
- **Removing Stop Words:**there are certain words that are important for human interaction. spaCy comes with a default list of stop words that you can customize.

  What’s the difference between Tokenization and stop words ?
  On observing the outputs after the two processes we can deduce With the stop words removed, the token list is much shorter, and there’s less context to help you understand the tokens.
- **Normalizing Words**:normalization is a little more complex than tokenization and Removing stop words in the fact that in this words are condensed into their form.For example: “watching”,”watched” and “watches” will all be normalized to “watch”.There are two ways on to do the same:
    - **Lemmatization**:it seeks to address this issue. This process uses a data structure that relates all forms of a word back to its simplest form, or lemma. Because

lemmatization is generally more powerful than stemming, it's the only normalization strategy offered by spaCy.

    - **Stemming**:With stemming, a word is cut off at its stem, the smallest unit of that word from which you can create the descendant words. You just saw an example of this above with "watch." Stemming simply truncates the string using common endings, so it will miss the relationship between "feel" and "felt," for example. We don't need any additional code to do this. It happens automatically—along with a number of other activities, such as part of speech tagging and named entity recognition—when you call nlp(). You can inspect the lemma for each token by taking advantage of the .lemma_ attribute.[Underscore with the lemma_ attribute is not a typo, but instead a convention that gets the human readable version of the attribute]
- **Vectorizing Text**:Vectorization is a process that transforms a token into a vector, or a numeric array that, in the context of NLP, is unique to and represents various features of a token. Vectors are used under the hood to find word similarities, classify text, and perform other NLP operations.This particular representation is a dense array, one in which there are defined values for every space in the array. This is in opposition to earlier methods that used sparse arrays, in which most spaces are empty. vectorization is taken care of automatically with the nlp() call.

Building the Sentiment Analyser

For this part we will build a sentiment analyser which when fed with data would able to predict the connotation of the input.For the same we will be training our model on the "large Movie Dataset" presented by Stanford researcher.that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

The IMDB dataset has already been divided into train and test, but it lacks a validation set.But a validation set was missing, the same was created using 80:20 split using the validation_split argument.

The labels are 0 or 1. To see which of these correspond to positive and negative movie reviews, you can check the `class_names` property on the dataset.

Preparing Dataset for training:

We standardize, tokenize, and vectorize the data using the helpful (preprocessing.TextVectorization) layer.

As the reviews contain raw text, they will occasionally also contain HTML tags.With the standardization process we will remove the text, typically to remove punctuation or HTML elements to simplify the dataset. Tokenization refers to splitting strings into tokens (for example, splitting a sentence into individual words, by splitting on whitespace). Vectorization refers to converting tokens into numbers so they can be fed into a neural network. All of these tasks can be accomplished with this layer.These tags will not be removed by the default standardizer in the TextVectorization layer (which converts text to lowecase and strips punctuation by default, but doesn't strip HTML). For the same we will write a custom standardization function to remove the HTML.

A model needs a loss function and an optimizer for training. Since this is a binary classification problem and the model outputs a probability (a single-unit layer with a sigmoid activation), you'll use losses.BinaryCrossentropy loss function.

In the end we get to see that accuracy is 87% and in the end we try it on custom input.

We see that the prediction for "The movie was great!", "The movie was okay." and "The movie was terrible..." are 0.62, 0.451 and 0.366 respectively, which are positive(closer to 1), neutral and Negative(closer to 0).