Sentiment Analysis

(Application of NLP, Text analysis and ML)

Analyzing Movie Reviews Sentiment

Duration : 2 weeks (70 hrs)

The problem at hand

The problem at hand is sentiment analysis or opinion mining, where we want to analyze some textual documents and predict their sentiment or opinion based on the content of these documents.

Sentiment analysis is perhaps one of the most popular applications of natural language processing and text analytics with a vast number of websites, books and tutorials on this subject. Typically sentiment analysis seems to work best on subjective text, where people express opinions, feelings, and their mood. From a real-world industry standpoint, sentiment analysis is widely used to analyze corporate surveys, feedback surveys, social media data, and reviews for movies, places, commodities, and many more. The idea is to analyze and understand the reactions of people toward a specific entity and take insightful actions based on their sentiment.

Basic terminologies

- A **text corpus** consists of multiple text documents and each document can be as simple as a single sentence to a complete document with multiple paragraphs. Textual data, in spite of being highly unstructured, can be classified into two major types of documents. Factual documents that typically depict some form of statements or facts with no specific feelings or emotion attached to them. These are also known as objective documents. Subjective documents on the other hand have text that expresses feelings, moods, emotions, and opinions.

- **Sentiment analysis** is also popularly known as opinion analysis or opinion mining. The key idea is to use techniques from text analytics, NLP, Machine Learning, and linguistics to extract important information or data points from unstructured text. This in turn can help us derive qualitative outputs like the overall sentiment being on a positive, neutral, or negative scale and quantitative outputs like the sentiment polarity, subjectivity, and objectivity proportions.

- **Sentiment polarity** is typically a numeric score that's assigned to both the positive and negative aspects of a text document based on subjective parameters like specific words and phrases expressing feelings and emotion. *Neutral* sentiment typically has 0 polarity since it does not express and specific sentiment, *positive* sentiment will have polarity > 0, and *negative* < 0. Of course, you can always change these thresholds based on the type of text you are dealing with; there are no hard constraints on this.

What we plan to do

In this Coding Internship project, we focus on trying to analyze a large corpus of movie reviews and derive the sentiment.

We would cover a **two varieties** of techniques for analyzing sentiment, which include the following.

- **Unsupervised lexicon-based models**
- **Traditional supervised Machine Learning models**

How do you submit the project

1. To show case your ***Sentiment analysis work*** you need to write a blog article showcasing your data visualizations and thereby your conclusions.
2. Provide the **URL link of your blog article** when submitting the project for assessment. You can write your Blog using any one of the following tools wordpress.com, wordpress.org, medium.com, blogspot.com or blogger.com
3. ***Immediately on submission,*** the **AI engine would assess your knowledge through a 10 minutes MCQ test**. If you clear it with minimum 6 correct answers out of 10, you would be awarded the Certificate of Internship.

Problem Statement

The main objective in this Internship Project is to predict the sentiment for a number of movie reviews obtained from the ***Internet Movie Database* (IMDb)**. This dataset contains 50,000 movie reviews that have been pre-labeled with "positive" and "negative" sentiment class labels based on the review content. Besides this, there are additional movie reviews that are unlabeled.

The dataset can be obtained from http://ai.stanford.edu/~amaas/data/sentiment/ , courtesy of Stanford University and Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. They have datasets in the form of raw text as well as already processed bag of words formats. **We will only be using the raw labeled movie reviews for our analyses.**

Hence our task will be to predict the sentiment of 15,000 labeled movie reviews and use the remaining 35,000 reviews for training our supervised models.

Setting Up Dependencies

We will be using several Python libraries and frameworks specific to text analytics, NLP, and Machine Learning. Before starting the Internship Project you need to make sure you have pandas, numpy, scipy, and scikit-learn installed.

NLP libraries which will be used; include spacy, nltk, and gensim. Do remember to check that your installed nltk version is at least >= 3.2.4, otherwise, the ToktokTokenizer class may not be present.

For nltk you need to type the following code from a Python or ipython shell after installing nltk using either pip or conda.

```
import nltk
nltk.download('all', halt_on_error=False)
```

For spacy, you need to type the following code in a Unix shell/windows command prompt, to install the library (use pip install spacy if you don't want to use conda) and also get the English model dependency.

```
$ conda config --add channels conda-forge
$ conda install spacy
$ python -m spacy download en
```

Guidelines to complete the Internship with in 2 weeks :

Step 1 : Text Pre-Processing and Normalization

(*use code file text_normalizer.py*)

One of the key steps before diving into the process of feature engineering and modeling involves cleaning, pre-processing, and normalizing text to bring text components like phrases and words to some standard format. This enables standardization across a document corpus, which helps build meaningful features and helps reduce noise that can be introduced due to many factors like irrelevant symbols, special characters, XML and HTML tags, and so on. ***Use the utilities*** *text_normalizer.py* ***file which does below steps (also called text normalization pipeline)***

- **Cleaning text**: Our text often contains unnecessary content like HTML tags, which do not add much value when analyzing sentiment. Hence we need to make sure we remove them before extracting features. The BeautifulSoup library does an excellent job in providing necessary functions for this. ***Our*** *strip_html_tags(...)* ***function does this work.***

- **Removing accented characters**: In our dataset, we are dealing with reviews in the English language so we need to make sure that characters with any other format, especially accented characters are converted and standardized into ASCII characters. A simple example would be converting é to e. ***Our*** *remove_accented_chars(...)* ***function does this work***.

- **Expanding contractions**: In the English language, contractions are basically shortened versions of words or syllables. These shortened versions of existing words or phrases are created by removing specific letters and sounds. Examples would be, expand don't to do not and I'd to I would. Contractions pose a problem in text normalization because we have to deal with special characters like the apostrophe and we also have to convert each contraction to its expanded, original form. ***Our*** *expand_contractions(...)* ***function does this work.***

Hint : *use code file "contractions.py"*

- **Removing special characters:** Another important task in text cleaning and normalization is to remove special characters and symbols that often add to the extra noise in unstructured text. Simple regexes can be used to achieve this. Its your choice to retain numbers or remove them, if you do not want them in your normalized corpus. ***Our function*** *remove_special_characters(...)* ***helps us remove special characters***.

- **Removing stop words:** Words which have little or no significance especially when constructing meaningful features from text are also known as stop words. Words like a, an, the, and so on are considered to be stopwords. There is no universal stopword list but we use a standard English language stopwords list from nltk. You can also add your own domain specific stopwords if needed. ***The function*** *remove_stopwords(...)* ***helps us remove stopwords and retain words having the most significance and context in a corpus.***

• **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. A simple example are the words WATCHES, WATCHING, and WATCHED. They have the word root stem WATCH as the base form.

The nltk package offers a wide range of stemmers like the PorterStemmer and LancasterStemmer. Lemmatization is very similar to stemming, where we remove word affixes to get to the base form of a word. However the base form in this case is known as the root word but not the root stem. The difference being that the root word is always a lexicographically correct word (present in the dictionary) but the root stem may not be so. **We will be using Lemmatization only** in our normalization pipeline to retain lexicographically correct words. ***The function*** *lemmatize_text(...)* ***helps us with this aspect.***

We use all these components and tie them together in the following function called **normalize_corpus(...)**, which can be used to take a document corpus as input and return the same corpus with cleaned and normalized text documents.

Refer Jupyter Notebook ***Text Normalization Demo.ipynb***

1st Approach : Doing Sentiment Analysis using Unsupervised Lexicon-Based Models

Unsupervised sentiment analysis models use well curated knowledge bases, ontologies, lexicons, and databases that have detailed information pertaining to subjective words, phrases including sentiment, mood, polarity, objectivity, subjectivity, and so on. A lexicon model typically uses a lexicon, also known as a dictionary or vocabulary of words specifically aligned toward sentiment analysis. Usually these lexicons contain a list of words associated with positive and negative sentiment, polarity (magnitude of negative or positive score), parts of speech (POS) tags, subjectivity classifiers (strong, weak, neutral), mood, modality, and so on. You can use these lexicons and compute sentiment of a text document by matching the presence of specific words from the lexicon, look at other additional factors like presence of negation parameters, surrounding words, overall context and phrases and aggregate overall sentiment polarity scores to decide the final sentiment score. There are several popular lexicon models used for sentiment analysis. Some of them are mentioned as follows.

- **Bing Liu's Lexicon**
- **MPQA Subjectivity Lexicon**
- **Pattern Lexicon**
- **AFINN Lexicon**
- **SentiWordNet Lexicon**
- **VADER Lexicon**

Make sure of googling/reading about the last 3 lexicon Models.

We would be using the last **3 lexicon Models for Sentiment Analysis. Refer the Jupyter Notebook** *Sentiment Analysis - Unsupervised Lexical .ipynb*

Official web site links

1> for AFINN Lexicon
https://github.com/fnielsen/afinn/blob/master/afinn/data/

2> for SentiWordNet Lexicon
http://sentiwordnet.isti.cnr.it

3> for VADER Lexicon
https://github.com/cjhutto/vaderSentiment

Conclusion from the first Approach :

1. In your **blog** (*which you would write for submitting the work*) compare the **overall F1-Score and model accuracy** of **the 3 Lexicon Models.**

2. State **the winner** from your calculations. And put across from your point of view , why one of the Models performed best ? Does it has any relation to the data set.

2nd Approach :
Classifying Sentiment with Supervised Learning

Another way to build a model to understand the text content and predict the sentiment of the text based reviews is to use supervised Machine Learning. To be more specific, we will be using classification models for solving this problem.

The major steps to achieve this are mentioned as follows :

1. Prepare train and test datasets *(optionally a validation dataset)*
2. Pre-process and normalize text documents
3. Feature engineering
4. Model training
5. Model prediction and evaluation

Refer Jupyter Notebook Sentiment Analysis - Supervised.ipynb

Conclusion from the Second Approach :

1. In your **blog** (*which you would write for submitting the work*) compare the **overall F1-Score and model accuracy** of **this Supervised ML Model with the best Unsupervised Lexicon Model.**

<u>Objective of this Coding Internship</u> :

You would be wondering that the "Suven Data Science Team" has given around 90% of the code to help me out during the Internship, then what's the use of the Internship ?

Well, the code given is definitely for helping you, but more than the code , its important for each Intern to understand the Approach to solve any Sentiment Analysis problem.

Try to Google and read on different functions which we have used in the code samples (as given in the Jupyter Notebooks). Try to play with their parameters / hyper-parameters values. The more you read about the both the approaches , **better would be your performance in the AI assessment test**.

<u>Note</u> : The AI assessment test is an MCQ test and starts immediately on project link submission.

All the best !!