**IMDB Review Classification using Word Embeddings(GloVe)**

Opinion mining (sometimes known as sentiment analysis or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

We can use classifiers on textual data to classify the text into its signified emotion. This can be used in various ways to predict the sentiments of any text automatically after it is trained in a corpus of sentences.

In this project we will classify the IMDB reviews for positive and negative reviews.

**What are word embeddings?**

In very simplistic terms, Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text..

So far, we have seen deterministic methods to determine word vectors. But these methods proved to be limited in their word representations until Mikolov et. el introduced word2vec to the NLP community. These methods were prediction based in the sense that they provided probabilities to the words and proved to be state of the art for tasks like word analogies and word similarities. They were also able to achieve tasks like King -man +woman = Queen, which was considered a result almost magical. So let us look at the word2vec model used as of today to generate word vectors.

What word2vec does is, it updates the word vectors of a particular word by the probability of finding the word in the vicinity of the context words(i.e. The words around it).So the words which occur around same context words will have word vectors with much similarity.

Ex. ‘need’ and ‘required’ generally have the same context words around it and thus making them similar in meaning. So, the word vectors of ‘need’ and ‘required’ will be very much similar.

In this project we will be using a variation of word2vec i.e. GloVe(Global Vectors for Word Representation by Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014).This introduces a concept of negative sampling into the word2vec algorithm.

We will use pre-trained word vectors in the project that is open sourced in the Stanford website.

<https://nlp.stanford.edu/projects/glove/>

You can download the word embedding with GloVe here:

<http://nlp.stanford.edu/data/glove.6B.zip>

This has many text files with different dimensions of the word vectors.We shall use the glove.6B.50d.txt

Each line has space separated word and embeddings of 50 dimension .

<word> <fd1> <fd2> <fd3> …… <fd50>

“fd1” represents the feature along the first dimension.

So the GloVe vector for the word ‘word’ is [fd1 , fd2 , fd3 , ........ , fd50]

You should use proper data structure assignments for each word vectors to store the word vectors otherwise if your RAM is <4GBs ,your system might crash.

**Note:** We dont require all the words in the text rather we store only the words present in our dataset. So first take a set of all the unique word in your dataset and then extract the word vectors. If there is a word in the dataset that is not present in the text file of word vector, then assign the word vector of that word to vector of zeros of 50 dimensions.

**How to use word embeddings for sentiment classification?**

We will use 2 methods for the classification:

* Unweighted sentence vectors:

Let’s suppose we have a sentence ‘S’ which has 3 words (w1, w2, w3) and we want to predict its sentiment.

Each of the word has 50-dim word vectors [fd1 , fd2 , fd3 , ........ , fd50]. To get the sentence vector we just add the word vectors for all the words in the sentence along each dimension. In this case:

Sentence vector(S)[50-dim]= (word\_vector(w1)+word\_vector(w2)+word\_vector(w3))

This is our training data which has 50 features .

* Normalized sentence vectors:

Sentence vector normalized(S)[50-dim]= (word\_vector(w1)+word\_vector(w2)+word\_vector(w3))/ (number of words)[i.e 3 in this case]

We have sentence vectors generated above we normalize the sentence vectors along each dimensions.(Use L2 Norm for this ).

# 

# **Prerequisites**

We would highly recommend that before the hack night you have some kind of toolchain and development environment already installed and ready. If you have no idea where to start with this, try a combination like:

* Python
* scikit-learn / sklearn
* Pandas
* NumPy
* matplotlib
* An environment to work in - something like Jupyter or Spyder

For Linux people, your package manager should be able to handle all of this. If it somehow can't, see if you can at least install Python and pip and then use pip to install the above packages.

**Objective:**

* Using unweighted and normalized sentence vectors as features for the text using GloVe word embeddings.
* Use different standard classifiers for classification of the texts.
* Compare the accuracy of both normalized and unnormalized sentence vectors.
* Compare the accuracy of the classifiers using different standard classification metrics.

**Dataset:**

The dataset is in form of a txt file and can be downloaded from:

[https://drive.google.com/file/d/1zXOovxa6RSqmIOaFu9RNlluVlmyBfJ0M/view?usp=sharing](https://drive.google.com/file/d/1zXOovxa6RSqmIOaFu9RNlluVlmyBfJ0M/view?usp=sharing%5C)

Each line has the text and tab separated target value.

1-Positive review

0-Negative review

**Workflow:**

* Extract and store the word vectors for the words present in your dataset using proper data structures.
* Generate the sentence vectors both normalized and unnormalized for the textual data.
* Use the sentence vectors as the features of the sentences and the target of the training data to train the classifier.
* Use various standard classifiers and train with both normalized and unnormalized sentence vectors separately and compare the accuracy with various metrics like f1-score and accuracy\_score.
* Finally compare the different classifier models on the data.