Sentiment Analysis

Anirudh Ramanan
8th July 2018

Sentiment Analysis is a process of determining whether a given statement is positive, negative or neutral. A basic task in **sentiment analysis** is classifying the polarity of a given text of the document, sentence, or feature/aspect level —whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral.

A common use case for this is to determine what people think or feel about a particular topic, or helping a business to understand the social sentiment of their brand, product or the services they offer by monitoring the online conversations.

Domain Background

Sentiment Analysis is being commonly used to understand the customer's opinion, process reviews and to understand the social sentiment of the company. **SemEval,** an annual event is an ongoing series of evaluations of semantic analytics systems intended to explore the nature of the meaning in the language. This began with single attempts to identify word senses computationally. They have now evolved to investigate the interrelationships among the elements in a sentence, relations between sentences.

Sentiment Analysis has been using a more linguistic approach, and have been focusing on extracting the opinion holders, and quotes from the text. As the Natural Language Processing techniques keeps on improving and computational power gets cheaper, and with availability of internet with good training dataset, it has become much easier and more efforts are being put into automated text processing methods.

I have been working on a model to predict stock market prices, and one of the main part of it is the Sentiment Analysis i.e to predict the sentiment of a given company by extracting out tweets, news articles and such other sources. This will be used to determine if the company or a brand has a good image among the consumers as the company's image is one of the major factor which affects the company's share prices.

Problem Statement

To classify the sentiment of sentences from the given dataset. For this project, I will be using the **Large Movie Review Dataset** to classify a given movie review as positive or negative.

Datasets and Inputs

The dataset has been downloaded from http://ai.stanford.edu website. This dataset contains movie reviews along with their associated binary sentiment polarity labels. It is intended to serve as a benchmark for sentiment classification. This document outlines how the dataset was gathered, and how to use the files provided. Checkout the References section for credits.

The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). It also includes an additional 50,000 unlabelled documents for unsupervised learning.

This means the training dataset contains 12.5k positive and 12.5k negative reviews.

In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets. In the unsupervised set, reviews of any rating are included and there are an even number of reviews > 5 and <= 5.

Solution Statement

The solution is to map each item to a sentiment classification (positive or negative). The trained model will be first tested on the validation dataset to

measure the performance of the model, and then will be used for the test dataset. The final aim of the model will be to assign a sentiment score to the individual data in the test dataset. I will be trying out different approaches to test which model with what hyper-parameters gives the best result (mentioned in the Project Design section)

Benchmark Model

I will be using random forest classifier as a benchmark model which will be trained on the vectors created from the words in the sentences, which in turn will be used to predict and classify the sentiment of a review.

Evaluation Metrics

The test set is already divided into positive and negative categories. We will cross-verify the results that the model has predicted against the test set category. We can then calculate the number of False Positive and True Negatives. Along with the FP and TNs, accuracy score will also be used to compare against different models. Since the dataset has a balanced set of positive and negative reviews, I think accuracy score will be a right metric against which we can compare models.

Project Design

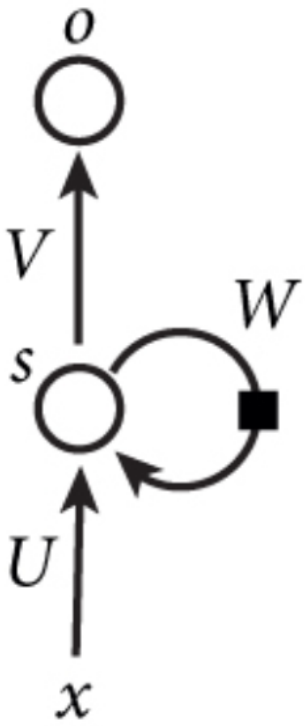
Neural Networks with NLP can be applied in determining the emotion from a given sentence. When you think of NLP task for solving sentiment analysis, this is what you may think it looks like

But this is not as straight forward as depicted in the image. There is no way of doing back-propagation and other neural network tasks on a single input string. Instead each word in the sentence has to be converted into a vector format which can then be fed into the neural network for training. We will be using **Word2Vec** to create word embeddings. **Word2Vec** is a group of related

models that are used to produce word embeddings. Its input is a text corpus and its outputs a set of vectors i.e it turns text into numerical form that the neural network can understand.

RNN (Recurrent Neural Networks)

Given that we have the vector representation of the words of the given sentence, we can start building the neural network. RNN (Recurrent Neural Networks) is a suitable model as we need to make use of the sequential information to predict the sentiment. Unlike feedforward neural networks, RNNs can use their internal memory state to process the sequence of the inputs.



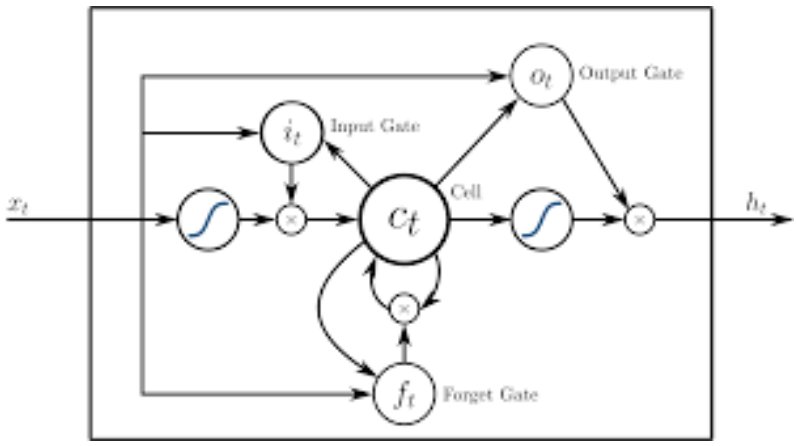
RNNs perform the same task for every element of a sequence, with the output being dependent on the previous computations. Theoretically, RNNs can make use of information in arbitrarily long sentences, but when implemented they are limited to looking only few steps backward. So for arbitrarily long sentences, RNN may not perform well.

LSTM (Long Short Term Memory)

LSTMs are a special kind of RNN which is capable of learning and storing long term dependencies. A regular LSTM consists of a cell, an input gate, an output

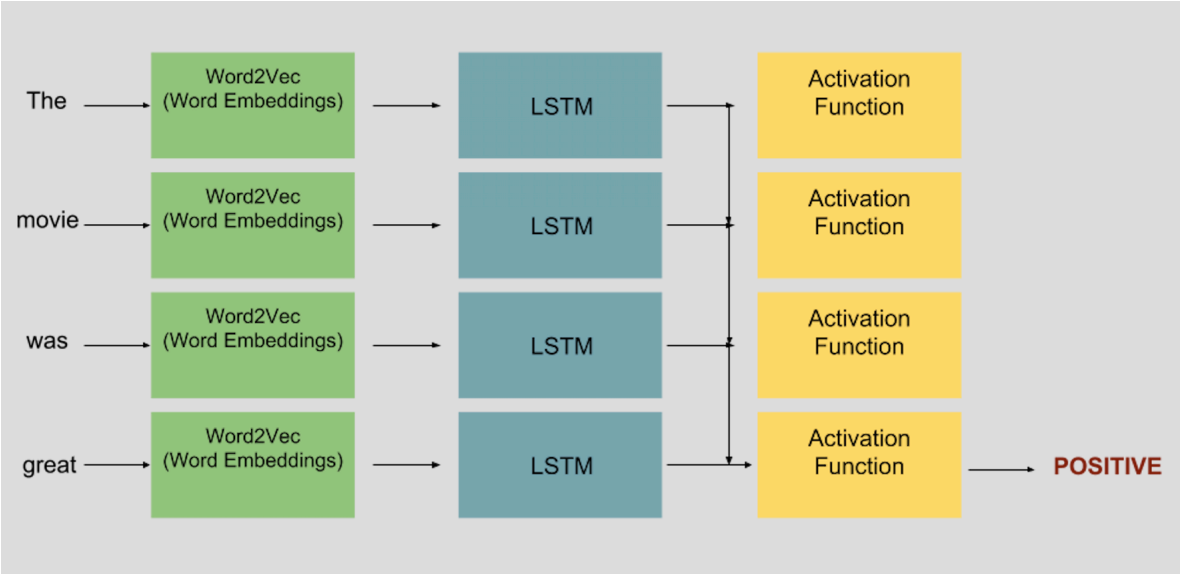
gate and a forget gate. The cell is primarily responsible for remembering values over arbitrary time periods. The gates compute an activation using the logistic function. The input gate controls how much information flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

Each gate takes in an input X(t) and h(t-1) as the inputs and perform some computations to determine the next state. Each state in then fed into different pipelines and the final output (h(t)) is calculated.



Building model

This is how the high level architecture looks like:



We will pass in the words to the embedding layer. Instead of using the pre-trained **Word2Vec** model, we will be creating our own embedding layer. Listing down the series of step that will be followed to build the model.

1. Data Preprocessing
    1. Cleaning data
    2. Converting words into vectors
2. Splitting training set to train and validation set
3. Build the LSTM Graph
4. Training
5. Testing

In addition to the custom embedding layer that we created, we will also try and run it with **Word2Vec** model and compare the results.

References

LSTM Wiki : https://en.wikipedia.org/wiki/Long_short-term_memory

Movie Dataset Credits : http://ai.stanford.edu/~amaas/data/sentiment/

O'reilly LSTM : https://www.oreilly.com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow

RNN Wiki : https://en.wikipedia.org/wiki/Recurrent_neural_network