Machine Learning Nanodegree
Capstone Proposal

Parth Singh

Feb 11, 2018

1 Domain Background

Analysis and recognition of sentiments is a fundamental task designated under natural language processing. It has found a broad spectrum of utility including recommendation systems, processing reviews and social media comments.

Vector representations of information, more specifically words, such that the constructed vector space behaves in way that is useful for analysis is crucial to the idea of vector embedding. And when this idea is applied to words or phrases it is called word embedding. Word embedding is now a popular and effective technique that creates input which is suitable to operate upon by a model such an LSTM (Long Short Term Memory) network. One such embedding commonly available is the Word2Vec model and it performs well in trying to capture semantic closeness between words, and it is not very cumbersome either.

1.1 Motivation

I have generally been fascinated with the development of natural language processing and its applications off late. Personally I wanted to dive into the techniques being employed in order to keep myself abreast with tools such as RNN. Choosing a dataset which has been worked on before will be a good start for me and I will be evaluate my performance appropriately I believe.

2 Problem Statement

I would like to try my hand at the the imdb-movie-review sentiment analysis problem. The problem essentially requires a binary classification of reviews into either positive or negative categories. There have been many attempts at solving this problem including a Kaggle competition, but a highly cited paper[1] by Mass et al. describes the Word2Vec model and also the authors made the dataset publicly available.

3 Datasets and Inputs

The dataset as mentioned was curated by Mass et al. and is easily accessible[2]. They collected 50K IMDB movie reviews. A rating of < 5 implies a negative sentiment while a rating of ≥ 7 implies a positive sentiment. No movie has more than 30 reviews in the dataset in order to not introduce a bias where some movies may have overwhelmingly numerous reviews and it may also lead to a correlation of reviews as reviewers may begin to get influenced by past reviews they read. There are 25K positive reviews and 25K negative reviews among the set. Neutral reviews were not included in the dataset. I will split the dataset into training and testing sets.

[1] http://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf
[2] http://ai.stanford.edu/~amaas/data/sentiment/

4 Solution Statement

5 Benchmark Model

A completely random model would give an accuracy of 50 percent based on the kind of dataset which has been curated. Obviously this would be the bare minimum that my model should achieve to be even classified as something that has "learnt" something. A slightly better model to benchmark against would be the Naive Bayes classifier after applying the appropriate pre-processing, perhaps bag of words and tf-idf.

6 Evaluation Metrics

Of course the most obvious evaluation metric would be accuracy. Here a random guesser would obtain an accuracy of 50 percent. But a more complete evaluation metric would be F-score, also separately evaluating the recall and precision, in other words inspecting the confusion matrix.

7 Project Design

7.1 Programming Language and Libraries

- **Python 2.**
- **scikit-learn**. Open source machine learning library for Python.
- **Keras.** Open source neural network library written in Python. It is capable of running on top of either Tensorflow or Theano.
- **TensorFlow.** Open source software libraries for deep learning.

7.2 Preprocessing

It will be necessary to preprocess the data in a way that will suit the specific type of model I want to develop. But after some exploration I have found that the keras library actually carries with it an easy statement to import the imdb dataset. I will further use the NLTK library to convert comments into sentences then further into words and removing stop words. I plan to also lemmatize the words

7.3 Vectorization

Once we have a list of lematized words we create the vocabulary of the training data by first creating a bag of words. Each words gets assigned a numeric index based on its frequency. Then I plan to use two techniques two techniques other than the benchmark one (Naive Bayes) to test solutions.

- **LSTM**: This will follow the approach of using word2vec implementations which helps decide how similar two words are semantically based on their cosine similarity. Then I will use the LSTM (RNN) architecture to try and decipher whether a review was positive or not
- **NN**: Another method I would like to try is a more vanilla deep neural network or a Convolutional Neural Network based on appropriate preprocessing to compare the results and training time between LSTM and this method.