Capstone Project Proposal

Pratik Joshi
September 30th, 2017

Domain Background

Deep Learning and Natural Language Processing are two extremely exciting fields today. Deep learning has dominated the process of pattern recognition, and has immense applications in text analysis. Definitive progress on NLP is said to have started in the 1950s, but there have been detected earlier works. However, the development of recurrent neural networks in the 1980s, and of word embeddings(first explained in "A Neural Probabilistic Language Model" by Bengio) by Yoshua Bengio in the 2000s set unprecedented standards to the field of NLP. Recurrent neural networks(RNNs) are of great importance in the field of machine translation, text prediction, and creation of language models.

The field of NLP can greatly improve our understanding of similarity in languages. It can have immense applications in communication, wherein it can reduce the barrier of communication between people speaking different languages. It can help us carry out sentiment analysis of different texts, and help us understand how language is linked to emotions.

Problem Statement

In this project, I will be attempting to classify movie reviews by sentiment. Sentiment analysis is a largely commercial field in which companies like Netflix, and Amazon Prime are heavily invested in. It helps them prioritize positive movies for users and make the experience of browsing much more convenient for the users.

I will be using reviews from Rotten Tomatoes to classify movies into 5 classes:negative,somewhat negative,neutral, somewhat positive,positive. I will be using NLP and deep learning techniques to create a neural network that will carry out this multi-class classification.

Datasets and Inputs

The dataset I will be using is part of the Kaggle competition “Sentiment Analysis of Movie Reviews”(Link:https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data). There are two csv files, a training dataset and a testing dataset. Each of these datasets contain tab-separated files with phrases from the Rotten Tomatoes dataset.

Both the training and testing dataset contains the columns PhraseId,SentenceId, and Phrase. The SentenceId denotes the sentence number, the PhraseId denotes the phrase

number in a sentence, and the Phrase itself is a column containing the phrase string themselves. The training set also has the associated sentiment label attached to it, with the following integer values:

0 - negative
1 - somewhat negative
2 - neutral
3 - somewhat positive
4 - positive

I will be conducting preprocessing on the data before feeding it as input to the neural network. I will use gensim to convert the phrases into word embeddings using Doc2Vec. This converts words into corresponding vectors, and in addition to that, it converts the vectors into an aggregated sentence vector which can be directly fed as features in a classification algorithm. In addition to this,I will be one-hot encoding the output labels, depending on the sentiment class it belongs to.

Solution Statement

I will be creating a neural network using Keras to classify the given phrases according to their sentiment. The Doc2Vec functionality requires a word count dictionary. Hence, I will be running through the phrases and filtering out unique words and listing their counts. After this, I will feed the embeddings through to the neural network. The neural network will be fully connected.

If time permits, I will be plotting a t-SNE graph to display the word/sentence embeddings of the various phrases. This will help the observer to visualize which sentences are similar in sentiment.

Benchmark Model

The benchmark model for this project will be the leader(Mark Archer) on the public leaderboard of the Kaggle Competition(https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/leaderboard). The accuracy on the test set was 76.5% for the model, and I will be trying to come close to that accuracy.

Evaluation Metrics

The main evaluation metric in this project will be the categorization accuracy. However, I cannot effectively calculate the test accuracy without running my solution on the Kaggle Evaluator. As a primary benchmark, I will instead do a k-fold cross validation and improve my model based on the cross validation score. In the end, I will also feed the model and check out my categorization accuracy on the Kaggle Evaluator.

The categorization accuracy is : (Number of correctly classified datapoints)/(Total datapoints)

Project Design

The neural network that I will be making is currently a black box. I will be doing a lot of trial and error to optimize my accuracy. However, this is a general outline of my project:

1. **Preprocessing**
    a. Carry out a word count of all unique words of all phrases by appending all phrases to one huge body of text
    b. Using Gensim's Doc2Vec, convert the phrases into word embeddings.
    c. One-hot encode the labels for the neural network.
2. **Training the Doc2Vec Model**
    a. Using the above data, train Doc2Vec to generate vectors for each phrase
    b. Create a list of vectors as features, and one-hot encoded labels as output labels.
3. **Neural Network**
    a. Will be fully connected.
    b. Input layer contains same number of nodes as number of features
    c. Contains softmax output layer of 5 nodes, one for each class.
    d. Run same model with saved best weights for each cross validation set in the k-fold and determine categorization accuracy.
4. **Final Test Set Run**
    a. Run model with Kaggle Evaluator to determine final test categorization accuracy.