IRE Major Project Report

Humour Detection in Yelp Reviews

Ankita Jain (201505536)
Anurag Gupta (201301109)
Shraddhan Jain (201330174)

Problem Statement:

Our problem is to predict community- determined humor in Yelp reviews. Yelp reviews can be voted as funny or non funny,and given a review, it is our task to label it as funny or not, using machine learning techniques.

So our problem is to classify reviews is a binary classification problem in terms of machine language.

This would help distinguish between informative and sarcastic reviews and save time of the consumers and this is the main reason behind classifying reviews .

Paper - Humor Detection in Yelp reviews - de Oliveira, Alfredo Rodrigo
Link to paper: https: //cs224d.stanford.edu/reports/OliveiraLuke.pdf

Approaches used for classification :

- Shallow algorithm - SVM on bag of words
- Deep Learning
    - Deep Feedforward Networks on word vectors
    - Convolutional Neural Networks

Modules implemented:

- **Dataset Preprocessing and Tfidf vector**:

Downloaded the dataset for yelp reviews from their site.
Cleaned the json file and extracted reviews and their funny votes.

1) Reviews with votes >= 4 considered to be funny
2) 50,000 each of funny and not funny reviews make the tagged dataset

These reviews and tags are used to create tfidf vector using scikit-learn library.This is an open source machine learning library in python which has TfidfVectorizer and fit_transform inbuilt functions to create tfidf vector(bag of word representation of corpus).

Observations:

Number of funny reviews with 3 or more: 94224

Number of funny reviews with 4 or more: at least 50K

Number of funny reviews with 5 or more: 38182

So we have choosen reviews with at least 4 number of funny votes to be considered as funny to train the network.

● Shallow Algorithm:

In this we used support vector machine(SVM) to classify the corpus reviews as funny or non funny.

Support vector machines (SVMs) are a set of supervised learning methods used for classification. SVM take corpus's bag of word representation as input and accordingly trains the machine.

For this we used scikit-learn library to divide data into train and test samples using train_test_split method. After this we trained SVM on training data and used the SVC (Support Vector Classifier) class from scikit-learn. We give it the parameters *C*=1.0, *gamma*=0.0 and choose a linear kernel.

The training process of SVM is 5-fold cross validation.

SVM accuracy for 5-fold: 85%

● Word vector implementation:

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks, that are trained to reconstruct linguistic contexts of words. To construct this the network is shown a word, and must guess which words occurred in adjacent positions in an input text. The order of the remaining words is not considered important (bag-of-words assumption).

After training, word2vec models can be used to map each word to a vector of typically several hundred elements, which represent that word's relation to other words. This vector is the neural network's hidden layer.

To implemnt Word2vec **gensim** python library is used and it provides **Word2Vec** method to do so. The input to these is the sentances which is a list of list of each review tokens. For tokenization of review **nltk** package is used.

● Deep Feedforward Networks on word vectors:

To implement a feed forward network word vector representation of data is used as input. We implemented a 3-layer feedforward network with one input layer, one hidden layer, and one output layer.
The number of nodes in the input layer is determined by the dimensionality of our data set. Similarly, the number of nodes in the output layer is determined by the number of classes we have, also 2. The activation function for hidden layer transforms the inputs of the layer into its outputs. The activation function for the output layer is the softmax, which is simply a way to convert raw scores to probabilities.

To train our neural network in forward direction and also to backpropogate the error we used pybrain python library.

Following results are obtained by training deep feedforward network:

Epoch:26 Total error: 0.101353500873
Epoch:27 Total error: 0.101210977137
Epoch:28 Total error: 0.101513547716
Epoch:29 Total error: 0.101403366638
Epoch:30 Total error: 0.10133227253

After 30 epochs, we got an accuracy of 89.87%

● Covolution neural Network:

In our CNN The first layers embeds words into low-dimensional vectors. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time. Next, we max-pool the result of the convolutional layer into a long feature vector, add dropout regularization, and classify the result using a softmax layer.

For implementing this we used Tensorflow python library. TensorFlow is an open source software library for numerical computation using data flow graphs. In this Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors)

communicated between them.

The code took time to run, and we couldn't get the accuracies.

Challenges Faced

○ Humour in a statement is a very subjective feature

○ Cannot extend the model out of domain since we need labeled corpus to learn the classifier

○ Takes time for execution over 1,00,000 datapoints and is difficult to tune parameters again and again

Technologies used -

○ NLTK

○ Gensim

○ Scikit module for SVM

○ PyBrain

○ Lasagne

○ Keras

○ Tensorflow