CS 482/682 Final Project Report

News Title Classification

Jin Yong Shin, Jae Goan Park

Abstract: We tried to solve the multi-class labeling problem in textual classification. We demonstrate multiple approaches to obtain the lowest classification error rate and show where we have succeeded and failed.
Github: https://github.com/jaegoan9/News-Classification-Project

Section 1: Problem Statement

For our Deep Learning final project, we chose to explore 'Text Categorization' in both the realm of Machine Learning and Deep Learning. There are tons of text datasets on the world wide web, but we decided to choose the dataset of news headlines available on Kaggle. This dataset consisted of over 420,000 examples of news headlines and their respective categories along with their sources (links). There was a total of four categories: Business, Science & Technology, entertainment, and health. Our task in this project was to obtain the lowest possible classification error rate when classifying each news article title with a multi-class classifier.

We were not entirely familiar with text classification via deep learning as most of our experience in this course have been with image classification but came across several interesting articles that yielded somewhat mediocre text classification results. "Convolutional Neural Networks for Sentence Classification" by Harvard PhD student Yoon KIm, turned out to be popular literature among deep learning fanatics with over 1900 citations. This particular piece dealt primarily with training on convolutional neural networks. He had varying degrees of success across several different text datasets (MR, SST-1, SST-2 etc). What is interesting to note is that in his paper where he discusses model with convolutional networks, he himself was not able to achieve high accuracy with textual classification that tried to solve the problem of multi-class classification. Accuracy for binary text classification was much higher (nearly double) the accuracy of multi-class classification. We used his article and some personal opinions deep learning experts had about text classification but were not able to find a gold standard solution specific to our dataset as we could not find deep learning case studies for this particular dataset.

Section 2: Methods

Machine Learning

We first approached the problem with machine learning techniques to evaluate several predefined sklearn classifier models. As with many other text classification problems(sense disambiguation etc), we decided to use a "bag of words model" to create a vocabulary set for every single example given in the news title dataset. We initially read in the entire text corpus in order to use python's built-in PorterStemmer to obtain a unique set of vocabulary. Stemming would remove tense from words and therefore words that have the same meaning but were in different tenses would be transformed into equivalent phrases. We then did a frequency count of all the unique words across the collection of title corpus. This resulted in creating a vocab set of nearly 50,000 words, and clearly this was not a reasonable size to even create a full feature set for nearly 420,000 examples. We eventually decreased

the vocabulary set down to around 10,000 after manually removing unnecessary quotation marks and removing words (mostly proper nouns with weird encodings) as this helped collapse and eliminate some meaningless words. We were not able to use the entire dataset as our train-validation data as it would simply take too much time, and there was no guarantee the computer memory would be able to handle the massive amount of data being generated and saved as feature vectors (more than 10000 x 420000 assignments). As a result, we decided to segment the feature vectors into chunks of 1000 examples and dump them in json files from which we would read in the data later to fit our ML models. Note that we did not use full dataset even with json dump since the size of the files would be too big (10GB for entire thing) and would take too long to load the data.

We ended up using *lemmatization* instead of porter stemming as this returned the base (root) or the dictionary form of the word, and this reduced the vocab set size further (although the difference was minimal). We did a train-test split with 33% attributed to the test data (random split) and used a total of 60000 examples for all three ML models (SVM takes a lot of time even with multiple virtual cpu's so we used a portion of the entire dataset).

Deep Learning

We also used the pre-processed data (nlp techniques) explained in the above ML method to input to the neural nets instead of using nltk's word2vec and embedding each example. We believe that using a word to vector model (and not a bag of words model) will actually yield very low classification accuracy for a deep learning approach in case of a multi-class classification problem. Since we had high dimensions due to using a bag of words model (nearly 10000 distinctive features), we created two linear layers (4 outputs including the last output) to reduce the dimensions and made it so that the one with the highest probability will be chosen as the final class label at the end per example. For our deep learning approach, we used a total of 80000 examples to split into train and test for evaluation.

Section 3: Results

Machine Learning Approach

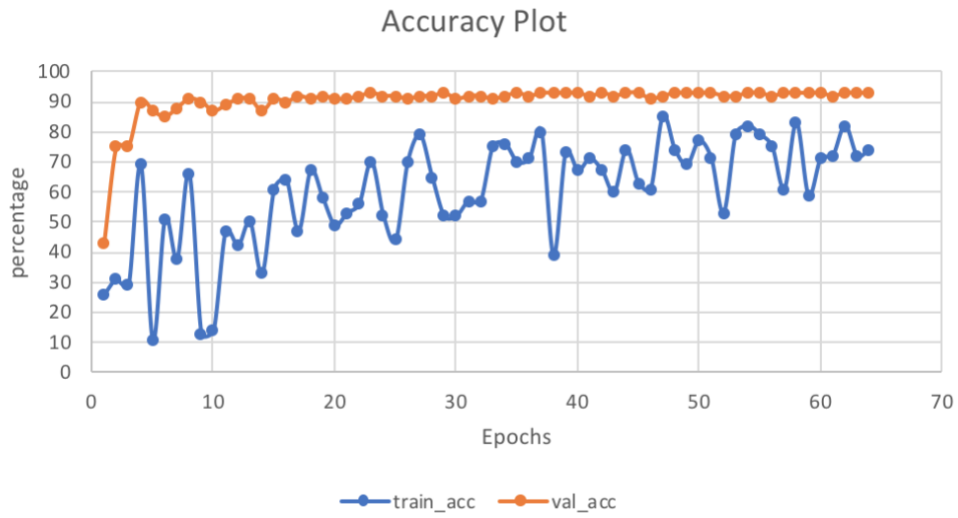
To evaluate our model, we fit our features onto three classic machine learning classifiers: Decision Tree Classifier, Multinomial Naïve Bayes Classifier, and the Support Vector Machine. The results were as follows:

Decision Tree (90.747 %), Multinomial Naïve Bayes (34.460 %), SVM (34.571%)

SVM ends up doing O(n_features * n^2_observations) operations which is considerably more than the other algorithms. Fitting on the ML models were all done on a 20vCpu machine via google cloud (thanks to Prof. Hager). While the Decision Tree Classifier and Multinomial Naïve Bayes algorithms took around 20 minutes each to fit, the SVM algorithm took nearly 10 hours to complete.

Deep Learning Approach

For our deep learning approach, we trained and tested on a bigger set of data (80000 examples which is 20000 more than ML approach) as the computation speed was much faster than that of the ML approach. Below are graphical representations of the results.



Best Train Acc: **83%, Best Val Acc: 93%**

Best train_loss: 2835.124756, Best Val_loss: 6096.544434

Unfortunately, when we tested our saved deep learning model on a **random segment** of the data set we hit an **accuracy rate of 39%.** We still consider this to me better than random chance as we had a total of 4 classes and random chance would mean getting 25% accuracy.

Section 4: Discussion

From concluding this experiment with some satisfactory, and some not so satisfactory results, we realized the importance of the "No Free Lunch" theorem Professor Hager taught to us in lecture. There simply is no algorithm that is dominantly better than the other. Depending on the scenario: size of data,

data type etc., we will see accuracy rates move up and down across depending on the algorithm being used. We initially approached this multi-class text classification with a machine learning approach, using three built in sklearn algorithms to evaluate them on the pre-processed data set. We concluded that the result for SVM was inferior to that of the decision tree classifier, as SVM is usually not efficient with big number of features and is performs poorly with multiple classes. In our hybrid ML/DL approach, assuming the using word to vector model with embedding will yield lower accuracy results for multi-class textual classification based on Yoon Kim's paper, we decided to pre-process our textual data by using a bag of words model to count word frequencies and divide them by total corpus frequency per word in order to regularize the features. This method had its pros and cons. Having cleaned up the data prior to training our model on the set of features seemed to yield high training and validation accuracy in both the ML and DL approach taken in our experiment. However, we concluded that our model created in the DL approach tended to perform rather poorly with a random segment of data found in the original dataset. When our saved model was evaluated on a random set of data that was not part of the testing or training before, we were only able to hit 39% tops with our approach. We agreed that this is most likely due to overfitting in the training process and that our model may have essentially done more memorizing than making an educational guess when it came to the final evaluation.

This may not be so surprising having seen that Yoon Kim has been only able to hit mid-forties to high forties for his accuracy rate when he tested his convolutional net model on the SST-1 dataset which was trying to solve the multi-class classification problem. We still consider our final deep learning model result to be satisfactory as it was better than random chance (25%), but feel that with more time we could perhaps take a different path with processing the data before evaluating our data on it. For instance, convolutional layers would only make sense if we used a "word2vec" model with embedding as spatial features are key to using CNN's effectively. In our case, because we used a bag of words model, the ordering of the words within each example was insignificant, therefore sending the input through conv layers would not be meaningful. At the same time, because we observed Yoon Kim's performance wasn't exactly stellar with CNN, we wanted to take a different route, and attempt to solve the classification problem with a hybrid ML, DL approach. Although our results weren't any better, we believe that there are still many approaches that could be taken with pre-processing the data to yield higher accuracy results when taking a deep learning approach to solve the problem. Not only will this approach have much faster computation speed than fitting machine learning algorithms but has more flexibility and potential in solving the general text classification problem. Perhaps we will attempt to contact the TA's or Prof Hager the following week to follow up on a new method to try before the final presentation.

We've learned that there are still so many undiscovered ways in which one can pre-process data prior to evaluating a classification model on the data. With not too much known about the specifics about how the magic behind deep learning works, it could be a matter of weeks or even days before we see another article that may show higher accuracy results with multi-class text classification models. Although a strong understanding of the fundamentals of probability and statistics is important to succeeding in the realm of deep learning, I also believe that there is nothing more important than experimentation. Keep training, keep testing, and keep changing the parameters.

References

Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).