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.

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 420000 examples of news headlines and their respective categories along with the sources (links). There was a total of four categories: Business, Science and 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. Historically, “Convolutional Neural Networks for Sentence Classification” by Yoon Kim, a PhD student at Harvard, turned out to be popular literature among deep learning fanatics with over 1500 citations. His work primarily dealt with training on convolutional neural networks. He had varying degrees of success across several different text datasets (MR, SST, 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, 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 considered equivalent. 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 eradicate 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.

We ended up using lemmatization instead of porter stemming as this returned the base 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 high vcpu’s so we used a portion of the entire dataset).

Deep Learning

We are also using 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. We created four distinct neural nets (4 outputs including the last output) 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.

We had high dimensions due to using a bag of words model (nearly 10000 distinctive features), so we used two linear layers to reduce dimensionality and then activated the input and got 4 outputs. We finally compared probability of four outputs and grabbed the highest probability and assigned the label connected to the highest probability to each particular example.

Section 3: Results

Machine Learning Approach

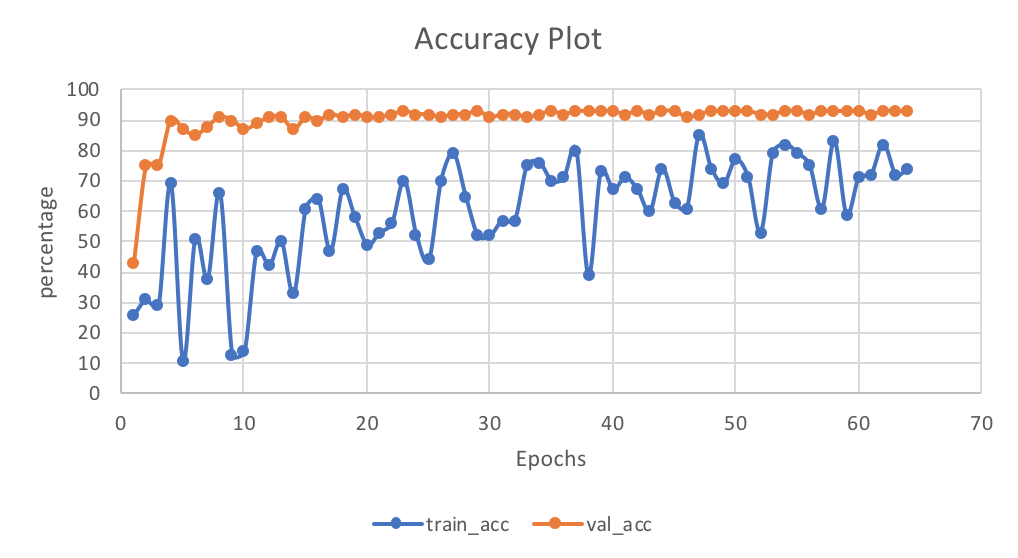
To evaluate our model, we fit our features onto three classic machine learning classifiers: Decision Tree Classifier, Multinomial Naïve Bayes Classifier, and the Support Vector Machine. The results were as follows:

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

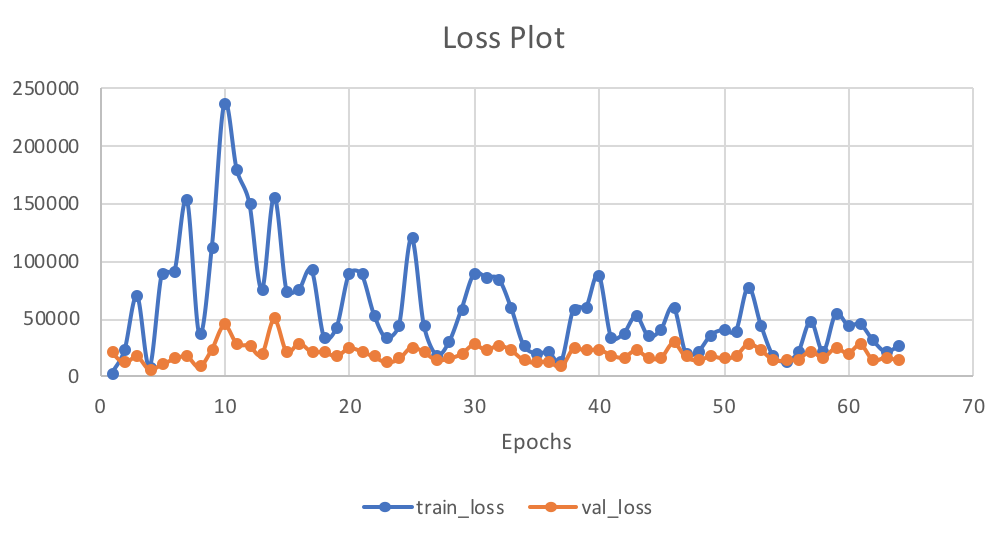
SVM ends up doing O(n\_features \* n^2\_observations) operations which is considerably more than the other algorithms. Fitting on the ML models were all done on a 20vCpu machine via google cloud (thanks to Prof. Hager). While the Decision Tree Classifier and Multinomial Naïve Bayes algorithms took around 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 different algorithms. 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. 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 mode to count frequencies and dividing them by total corpus frequency per word in order to regularize the quantitative measures.

* Analysis of results
* Known limitations and possible future extensions
* Two or three things you learned during this project that you didn’t know beforehand
* Any advice you would give to next year’s DL students (and instructors!).

References

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