Sentiment Analysis of Yelp Review Dataset

Fatima K. Suleiman, Souvik Roy, and Shaundell Dubay

***Abstract*— This paper implements sentiment classification on a Yelp review dataset to rate the intensity of the sentiment. The paper performs classification using a variety of classifiers, such as linear Support Vector Machine (SVM), Extreme Gradient (XG) Boost and Long-Short-Term-Memory (LSTM). In addition, experimental investigations of preprocessing techniques, methodologies and chosen parameters, are conducted in order to test the role of various natural language processing features. Examples include concepts related to class imbalance and stratified sampling, domain specific classification, Vectorizer performance and hyperparameter tuning.**

I. Introduction

In today's age of digital communication, review sites have become an important platform for sharing the consumer experience. Whether the consumer is interested in buying a product, considering a service or anticipating an experience, reviews from other users can provide important insight. As review sites are easily accessible, it is common for people to first investigate the opinions of others about the establishment, service or product that they are considering. Businesses are also interested in understanding how their customers feel about their service, product or even how they can improve their customer relations. To achieve this level of awareness, it is not practical to read every review, therefore, automatic detection is a necessity. Semantic analysis is a process that can be used to identify the feeling or opinion of a given text passage. The challenge of automatic classification of sentiment from review text requires natural language processing (NLP), text analysis and other computational methods.

Yelp is an American corporation that helps people locate local businesses based on a community of reviewers and reviews. Yelps platform allows customers to write a review documenting their experience or opinion, accompanied by a star rating between 1 and 5. With the rapid increase of traffic on their site, Yelps review dataset provides a diverse repository of sentiment.

II. Literature Review

In [10], sentiment analysis was performed to determine sentiment polarity of Yelp reviewed restaurants. The review text was used as the corpus and the star rating was the feature label for positive or negative sentiment; ratings of 4 or higher were considered positive and 3 or lower was considered negative. Initial preprocessing included separating words, removing punctuation and stemmed and lemmatized using Pythons NLTK package. The main approach was linear Support Vector Machine (SVM) operating with two statistics: bag-of-words (BOW) with count frequency and the term frequency inverse document frequency (TFIDF). SVM can perform well with text classification tasks as they are able to generalize with high feature dimensions and can therefore remove the need for feature selection [2]. The results show that both methods produced a test accuracy of around 88%. On the other hand, [1] implements linear SVM on a BOW with bigrams so that features can capture partial information of the word order that may be useful in determining sentiment. The results show a test accuracy of 59.8% on the 2013 Yelp dataset.

The authors in [8] look at citation analysis and sentiment analysis using Long-Short-Term-Memory (LSTM) models. Specifically for sentiment analysis, three classification categories are defined: negative, neutral and confirmative. A baseline model of LSTM and SVM with a linear kernel is used for comparison of other LSTM variations. The baseline models utilize TF-IDF statistics, as well as 1-2-3-grams for feature set development. In addition, the Adam optimizer and Word2vec were implemented for the baseline LSTM and hyperparameter tuning was done for the linear SVM. For experiments, stratified sampling is carried out in order to preserve the distribution of classes in the Yelp 2013 dataset. Results show that SVM and LSTM have a test accuracy of 53.89% and 66.42%, respectively. The work presented in [9] aims to achieve sentiment classification of long sentence text by using a LSTM model to capture sequence text information. Yelp 2014 and Yelp 2015 reviews are used as the large-scale datasets and the objective is to classify the text passage as negative, neutral or positive. Using NLP pipeline, sentences are first split into words and phrases; then summed scores of words or phrases in a given sentence represents the polarity. For the experiments, the LSTM batch size is set to 64, the learning rate is 0.01 and Adagrad optimizer is chosen. The classification accuracy for Naive Bayes (NB), SVM and LSTM are 61.3%, 58.9% and 61.7%, respectively. The work in [2] and LSTM Approaches, performs sentiment analysis on the Yelp dataset using Multinomial Naive Bayes (MNB), SVM and LSTM for classification. The review texts and star ratings were used, where negative was considered 2 and lower, positive was 4 and higher, and 3-star reviews were dropped. In addition, the idea of balanced and stratified distribution of classes was analyzed. Preprocessing was similar to other literature, and for MNB and SVM: BOW and TF-IDF were compared. In the LSTM model, pre-trained word2vec embedding was used as well as the following hyperparameters: NAdam optimizer, batch size equal to 64 and ReLu activation. Experimental results show a slight increase in accuracy when using TF-IDF and decreased accuracy when using a balanced distribution. Remaining results are as follows: SVM TF-IDF, MNB TF-IDF and LSTM have test accuracies of 94.17%, 89.70% and 95.89%.

III. Problem Definition

The Yelp Challenge Dataset [5] is a subset of the businesses, reviews, and user data information from a crowd-sourced forum that reviews local services and establishments. The review dataset consist of 5,261,668 review text, user-id, star ratings, useful votes and the business-id, which relates to business categories, location etc.

The objective of this work is to implement semantic text analysis in order to determine the sentiment of the review and rate the intensity of the sentiment on a scale of 1-5 which corresponds to the star rating of the review.

IV. Methods and Approach

A. Data Preprocessing

For this project, only **105,235** (**2%**) of the entire review text samples were analyzed due to hardware constraints. These samples were obtained from the entire dataset by stratified sampling to ensure that the actual sample distribution amongst the 5 classes was preserved. The following details the preprocessing techniques that were applied to the text reviews.

1) Punctuation and Stop words Removal: Stop words are commonly used words that appear frequently in texts for example: the, a, in. These words together with punctuation marks and symbols are filtered out when processing natural language text because they do not add to the meaning, especially for sentiment analysis application. Because the Yelp dataset is from businesses situated in the USA, it is assumed that all the reviews are in English and the stop words to be removed from the review text were evaluated in English as well.

2) Text Normalization: Text Normalization is the process of converting a group of text with the same meaning into a standard form for efficient processing and analysis. Stemming and Lemmatization are two main types of word form normalization used in natural language processing to obtain the root form of a word. Stemming mainly splits a word from its suffix and/or prefix while Lemmatization returns the dictionary root form of the word. Lemmatizers work best with words generated by inflection variance while Stemmers are good at handling derivational variance [7]. For this project, we implemented both a Porter Stemmer and WordNet Lemmatizer with an adjective part of speech tag (pos tag) to better capture the qualifiers that usually denote sentiment. The Lemmatizer was applied before the Stemmer.

3) Featurization Techniques: Featurization refers to the process of transforming textual data into the numerical format required for machine learning. The two broad featurization techniques that were considered for this project are Bag-of-Words and Word Embedding.

1) **The Bag-of-Words:** This approach creates a vocabulary vector where each entry/word corresponds to the features of the review dataset. In building our Bag-of-Words vocabulary, we ignored terms that had a document frequency lower 1%. The two popular Bag-of-Word vectorizers are **CountVectorizer** and **TFIDFVectorizer**. The *Countvectorizer* counts the number of times a word occurs in a text sample while *TFIDFVectorizer* utilizes both the count of a word in the sample as well as the frequency of the word in the entire corpus of the dataset. An important characteristic when implementing these vectorizers is the **N-gram** which defines sets of consecutive words that defines a feature of your data set. Features of unigram (N-gram=1) and bigram (N-gram=2) were combined for better accuracy because bigram allows for the preservation of the order of some commonly joined words in the text [1].

2) **Word Embedding:** This generates a vector representation of the words that capture its semantics, context, analogies and similarities [6].The Word Embedding that is used in this work is **Word2Vec**, a pre-trained two-layer neural network that predicts a dimensional vector space for a word based on its linguistic context. *Word2Vec* tends to create clusters as the resulting location of a word in the feature space is a result of its meaning and grammar. It also employs cosine similarity to ensure words with similar context occupy close spatial position [6].

B. Classifier Algorithms

The most common classifiers used in sentiment analysis is Support Vector Machine (SVM [10]. SVM is an algorithm designed to identify the hyperplane that best separates the different classes. While the performance of SVM for sentiment classification is well documented, the performance of other machine learning algorithms for this application is not easily found in the literature. For this reason, the following classifiers were also implemented for this project: Naive Bayes, K-Nearest Neighbor (KNN), Decision Tree, Random Forest, Gradient Boost, Extreme Gradient Boost (XGBoost).

With the aim of improving the performance of the individual classifiers mentioned above, an ensemble voting classifier was implemented by combining the Naive Bayes, Random Forest and XGBoost classifiers.

To compare the performance of classical classifiers with that of artificial neural networks, two neural network models were also implemented: Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN). MLP is a feed-forward neural network which uses backpropagation to determine the gradient for updating the weights at each neuron. LSTM is a neural network that utilizes a feedback loop to learn the relationship between sequential data such as the order of words in a review text [2]. LSTM solves the vanishing gradient problem faced by most neural network by preserving the back-propagated errors through time and layers.

C. Testing

The following are methods used to facilitate the performance evaluation for our classifiers and to prevent the possibility of overfitting.

1) Stratified Train-Test Split: The dataset is split into 80% training data and 20% test data. The split is stratified such that the classes are equally balanced in both the training and test sets. The training dataset will be used to fit the classifiers while the test dataset will be used to evaluate the capabilities of the classifier to accurately predict the sentiment rating.

2) Stratified k-fold Cross Validation: This involves dividing the dataset into **k** subset of equal sizes, the classifier is trained on **k-1** subsets with the left out subset used to validate (test) the classifier. This is repeated k times with each subset acting as the validation subset only once. The evaluation metrics are then averaged over all the k-folds. For our dataset, the **k** subsets are stratified i.e. the sample distribution amongst the classes is uniform for all the folds.

D. Hyperparameter Tuning

Each of the classifiers with the exception of Naive Bayes depends on its hyperparameters to control how it learns the training dataset. It is ,therefore, necessary to optimize the hyperparameters utilized by the classifier. Given our limited available hardware, a range of values was specified for the important hyperparameters for each of these classifiers and a grid search with k-fold cross validation was conducted to obtain the hyperparameters that will increase the prediction accuracy of the classifiers compared to their performance with the default hyperparameters.

E. Resampling Techniques

From Fig. 5 below, the imbalance of our dataset can be readily observed with class of ”5” rating has almost three time the number of data samples found in the classes of ”1” and ”3” ratings. Having this data imbalance problem can create a bias problem in the classifier models. Therefore, balancing the amount of samples in the class is necessary.

The class imbalance can be corrected by resampling the dataset using either an oversampling (upsampling) or undersampling (down-sampling) method. For oversampling, synthetic samples are added to the class with lower number of samples to match the dominant class. The two oversampling methods used for this dataset are as follows:

1) SMOTE Oversampling: SMOTE (Synthetic Minority Over-sampling Technique) uses k-nearest-neighbour to oversample the observations of the minority class to create synthetic samples along the lines connecting the samples of the minority class. The synthetic samples are similar but modified versions of the minority class samples.

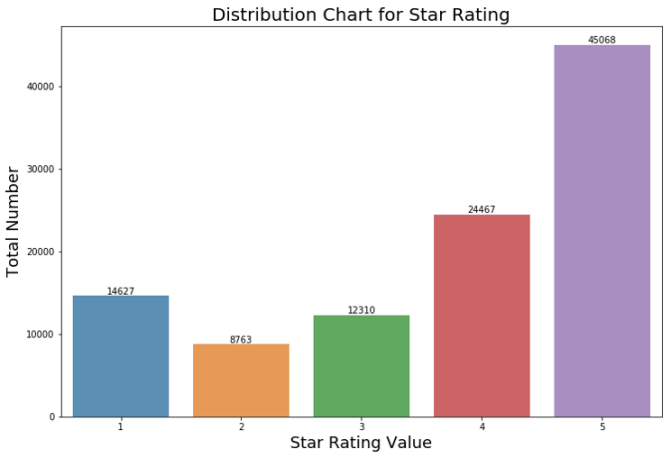

Fig. 1. Sample Distribution across the Sentiment Classes

2) ADASYN Oversampling: ADASYN (ADAptive SYNthetic) is an improved version of SMOTE in that it allows for variance by adding random values to the generated samples, this prevents linear correlation between original and synthetic sample.

In case of undersampling the classes with higher number of samples are decreased to the class with lowest number of samples. The two undersampling methods used for this dataset are described below:

3) Random undersampling: Random Undersampling samples the majority class in a random but uniformly way to obtain a class size that matches that of the smallest class.

4) NearMiss-1 undersampling: NearMiss-1 calculates the distances between all instances of the majority class and the instances of the minority class after which it removes the majority class samples with higher distance to the k-nearest points in minority class.

V. Results and Discussion

The results of various models evaluated for our dataset will be discussed in this section. The major Python libraries used in obtaining our results are scikit-learn, NLTK, Keras and imbalanced-learn. As mentioned in the above sections that, we have used 10 classifiers to classify the reviews and predict the rating of the reviews. There are two different type of review prediction that has been considered for this project. One of them is for rating 1,3 and 5 and other is for 1,2,3,4 and 5. That means these are 3 class problem and 5 class problem. The metric that is used for comparison between the models are cross-validation accuracy and F1 score with testing accuracy as well. There is one more metric that is used is confusion matrix to analyze the detail of classification for each class.

A. Experiment 1: Five Class with Deafult parameter

The mentioned metric used for comparison of performance of models will shed some light on how the class distribution affects the model performances. From the Fig. 5 with respect to unbalanced data-set and using bag-of-words method with TF-IDF feature list the accuracy and F1 score of all the classical algorithms and Multi-Layer Perceptron (MLP) are listed here. The evaluation has been done without any hyperparameter tuning of the models and Linear SVM model has performed better than other with 63.43% of accuracy as highest among them. Gradient Boost and Ensemble learning have more or less the same accuracy score with second best performing models. Also the result of Linear SVM are better than comparing with literature [9] but under-performed comparing to [10].

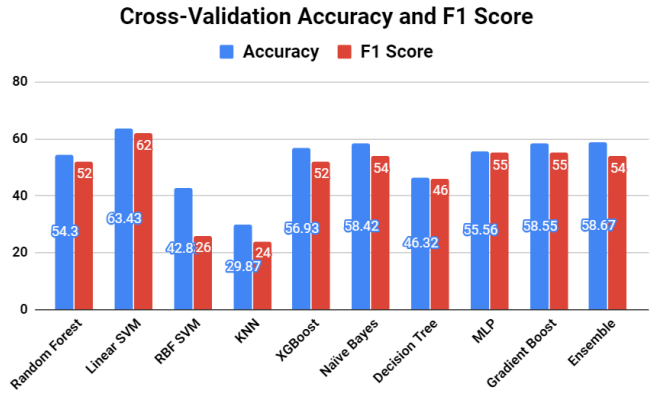

Fig. 2. Cross Validation Accuracy and F1 Score for Experiment 1

The least performing model over here is KNN with an cross validation accuracy of 29.87% followed by 42.8% for Radial Basis function SVM classifier. With this accuracy results, it was understood that we need to check further into confusion matrix as where exactly is more mis-classification error that occurred due to sentiment uncertainty.For this purpose, confusion matrix of SVM was analyzed and we found that the rating ”2” is majorly mis-classified as rating ”1” and rating ”4” is majorly mis-classified as rating ”5”. At the same time rating ”1” and ”5” have the highest prediction accuracy.

B. Experiment 2: Three Class with Default and Tuned Parameter

To overcome the problem of mis-classification, we have done the same experiment but for rating ”1”, ”3” and ”5”. This time the cross validation accuracy for Linear SVM has been increased by 22.7 and it performed better with 86.13% than previous score of 63.43%. Comparing to the literature’s [10] [8], the Linear SVM performed 46.2% higher but it did not exceed he performance of [9].

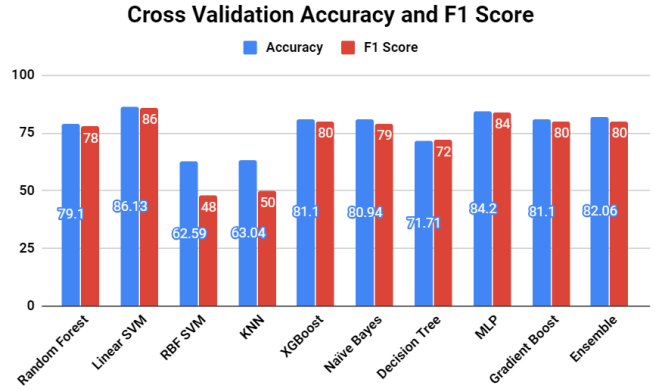

Fig. 3. Cross Validation Accuracy and F1 Score for Experiment 2

Since we have a better model to tune it for increasing the accuracy. We used CVGridsearch for finding out the best parameter for for all the models and then test the model with the dataset. The below table shows the comparison between the test and cross validation accuracy. From the previous cross validation accuracy and the tuned cross validation accuracy table we can observe that it has increased bsmall

value for Random Forest, Gradient Boost and Ensemble classifier but for RBF SVM it has increased to 86.28% from 62.59%. So, overall the accuracy has increased. There is one more important observation that the test accuracy is higher than the cross validation accuracy by just a small amount. But it does not mean that the model has overfitted.

	Cross Validation	Test
Random Forest	**81.53**	**81.49**
Linear SVM	**85.99**	**85.51**
RBF SVM	**86.28**	**86.12**
KNN	**62.91**	**63.12**
XGBoost	**80.05**	**79.99**
Naïve Bayes	**80.94**	**80.95**
Decision Tree	**74.1**	**73.94**
MLP	**84.91**	**84.66**
Gradient Boost	**84.23**	**84.04**
Ensemble	**82.08**	**82.15**

Fig. 4. Cross Validation and Test accuracy Score for Experiment 2 with Tuned Classifier Hyperparameters

C. Experiment 3: Domain Specific Classification

SVM has been proved as the best classifier so far in the last two experiment results. Therefore, domain specific analysis is done from the dataset. ”Food” is chosen as the domains as it contain quite high number of samples so that enough data samples are available for training model. In this experiment, our assumption was domain specific words are only there in the review, which might influence the model performance. So, Linear SVM is chosen as the model to compare.

	Mixed Category	Food Category	Difference
Cval Acc (%)	**85.99**	**84.93**	**-1.06**
Test Acc (%)	**85.51**	**85.00**	**-0.21**

Fig. 5. Accuracy of Domain Specific (Food) for Linear SVM

D. Experiment 4: Inverse Document Frequency (IDF) Effect

In this experiment, we are interested in looking the effect of Frequecy-Inverse Document frequency (TF-IDF) compared to countVectorizer in model performance. TF-IDF takes into account how much unique words and repeated common words does our data sample consist of in the dataset. From the Fig. 6 it is very clear that TF-IDF has performed better than CountVectorizer and there is quite significant increment in test accuracy.

	XGBoost	MLP	Random Forest
Count	**77.55**	**85.48**	**74.98**
TFIDF	**79.99**	**84.66**	**81.49**

Fig. 6. Comparison of accuracy of XGBoost, MLP and RF with CountVectorizer

E. Experiment 5: Class Imbalance

In this experiment we evaluated the four resampling techniques to prevent our model from being biased to the class with the most samples i.e. rating ”5”. The result from implementing the over sampling techniques, SMOTE and ADASYN are detailed in Fig. 7 while the results from the under sampling techniques, Unbalanced and Random are detailed in Fig. 8.

	Unbalanced	SMOTE	ADASYN
Test Acc(%)	85.51	83.57	83.74

Fig. 7. Comparison of the SMOTE and ADASYN Oversampling Test Accuracy

From Fig.7, we observed that there is decrease in Test accuracy compared to unbalanced class which is consistent with the result obtained by Flores et al [3] decreased from 84.64% to 82.22%. Also from the detailed analysis we observed that oversampling preserve the accuracy of major class.

	Unbalanced	Random	NearMiss-1
Test Acc(%)	85.51	82.41	82.58

Fig. 8. Comparison of the Unbalanced and Random Undersampling Test Accuracy

From the Fig. 8, we observed that there is decrease in Test accuracy compared to unbalanced class as in case of oversampling. And we also observed that undersampling is better able to predict the neutral class but the penalty of both sampling method reduces the accuracy of majority of classes.

F. Experiment 6: LSTM Models

In this section, LSTM model is being used to find out if he LSTM model have more performance than any other classical models. We tested with LSTM layer and batch size with a dropout of 20% was selected to avoid overfitting problem. The optimizer we use is adam, which is a variation of the popular Adam optimizer. The optimal LSTM layer size was found to be 128 with embedding layer of size of X_train. The output layer of our LSTM model is a Softmax function which is used to condense the output value of our network into a probability of classifying the review as number of classes. The number of epochs during model training was set to 10 as the consecutive training loss does not change less than 1% after that.

There are two models that have been tested for comprise. Model 1 with one LSTM layer and Model 2 with two LSTM layer of size 128 and 60 respectively. For this study, we permit the the length of training sample of most commonly occurring words in

our vocabulary. Once the words in the reviews are converted into their corresponding integers, for the word embedding approaches, we can prepare an embedding matrix which contains at index i, the embedding vector (Word2vec embedding) for the word at index i. The embedding matrix is then loaded into a Keras embedding layer and fed through the LSTM.

1) LSTM with TFIDF Vectorizer: In this experiment LSTM model is tested with two different models and TFIDF is used as embedding layer in this case. From the Fig 9 we can observe that the model testing accuracy for both the models remains the same, so there is no effect of increasing more LSTM layer in the model as it has no effect.

	LSTM Model 1	LSTM Model 2
Testing Accuracy	**85.04**	**85.04**
Testing Error	**32.49**	**32.50**

Fig. 9. Comparison of Testing accuracy of LSTM Model 1 and Model 2 with TFIDF Vectorizer

Also from the figure, there is no decrease in training loss even after 10 epochs and the training accuracy also does not decrease by any significant value.

2) LSTM with Word2vec: In this experiment we have used Word2Vec as word embedding layer for the same dataset as in above subsection. Both LSTM model used here are not changed and have same configuration, so that comparison can be done. From the Accuracy chart Fig Fig. 11, we can clearly observe that LSTM model 1 performs better than model 2 with decrease in training loss gradually. The model 1 accuracy is also increasing from epoch 1 till epoch 10 and same for Model 2. Model 2 gradient is less positive than model 1 that's why it increases slowly. And Model 1 has done well from initial epoch a d this suggest that LSTM Model with one layer works perfectly for our dataset. n

	LSTM Model 1	LSTM Model 2
Testing Accuracy	**86.48**	**86.97**
Testing Error	**29.77**	**28.51**

Fig. 10. Comparison of Testing accuracy of LSTM Model 1 and Model 2 with Word2Vec

The accuracy chart and training loss char indicates that after 9 epochs there is less than 1% change in the training loss between 9^{th} and 10^{th} epochs. This suggest that model has reached global minimum. The Word2Vec word embedding is very much efficient in vectorizing the words with combination of unigram and bigram. t

In the end, we can say that LSTM has performed better than Linear SVM with Word2Vec. This suggest hat LSTM can handle sentiment uncertainty whereas SVM is not that much efficient. Linear SVM performed well when there is three class problem as compared to five class problem. And LSTM has proven to be best than any other used classifier in this study for five class problem.

VI. Conclusion

We presented results for sentiment analysis on Yelp user review dataset. We have 10 different models for finding the best performing model that could beat the results with [10] [4] [9]. We presented a

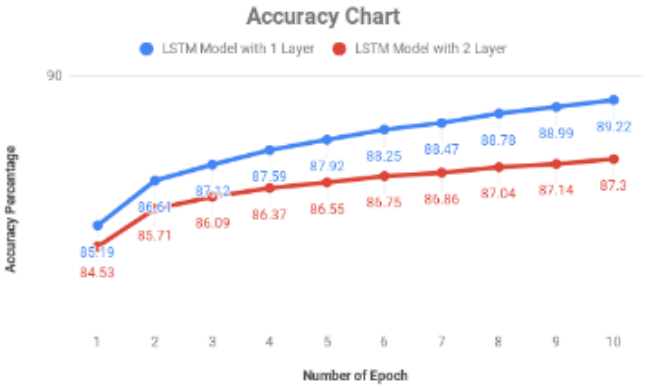

Fig. 11. Comparison LSTM Model 1 and Model 2 training accuracy

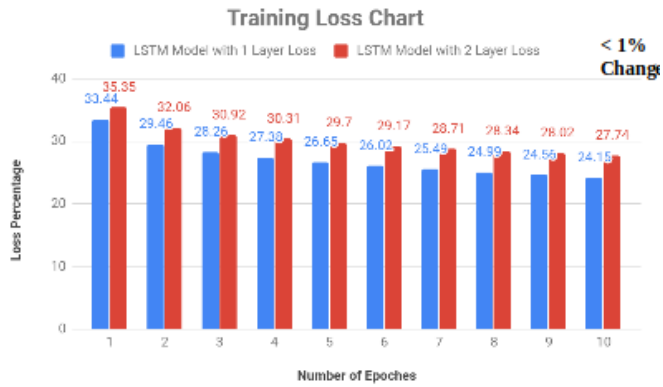

Fig. 12. Comparison LSTM Model 1 and Model 2 training loss

comprehensive set of experiments for both five class and 3 class problem. Set of 6 different experiments has been done in this study with bag-of-words and word embedding. In the first set of experiments we have compared the classical algorithm with default parameter for 5 class problem and default hyperparameter for 3 class problem as first step. Here Linear SVM performed better than any other models. We have also compared different oversampling and undersampling technique to overcome unbalanced dataset issue, but the results were not better than original dataset as found in [3].

We also compared our results across two different dataset distributions, a balanced distribution and one which follows the original distribution. While the greatest accuracy was achieved on the original distribution using TF-IDF rather than domain-specific embedding. The fact that the LSTM models achieved greater accuracy scores than the baseline model Linear SVM highlights their ability in NLP tasks. The LSTM models are able to learn more subtle relationships which the baseline models fail to pick up on as evident in their comparative test accuracy scores.

References

[1] P. Bojanowski A. Joulin, E. Grave and T. Mikolov. Bag of Tricks for Efficient Text Classification. *Computing Research Repository (CoRR)*, page arXiv:1607.01759, Aug 2016.

[2] James Barry. Sentiment analysis of online reviews using bag-of-words and lstm approaches. In *AICS*, 2017.

[3] A. C. Flores, R. I. Icoy, C. F. Pea, and K. D. Gorro. An evaluation of svm and naive bayes with smote on sentiment analysis data set. In *2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, pages 1–4, July 2018.

[4] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. Exploiting document knowledge for aspect-level sentiment classification. *arXiv preprint arXiv:1806.04346*, 2018.
[5] Yelp Inc. Yelp open dataset. Yelp, 2018.
[6] Dhruvil Karani and Dhruvil Karani. Introduction to word embedding and word2vec, Sep 2018.
[7] Haibin Liu, Tom Christiansen, William A Baumgartner, and Karin Verspoor. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *Journal of Biomedical Semantics*, 3(1):3, 2012.
[8] M Munkhdalai, J Lalor, and H Yu. Citation Analysis with Neural Attention Models. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis (LOUHI)* ,, pages 69–77, Austin, TX, November 2016. Association for Computational Linguistics.
[9] Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49 – 57, 2018.
[10] Boya Yu, Jiaxu Zhou, Yi Zhang, and Yunong Cao. Identifying Restaurant Features via Sentiment Analysis on Yelp Reviews. *arXiv e-prints*, page arXiv:1709.08698, Sep 2017.