International Journal of Sciences: Basic and Applied Research (IJSBAR)

ISSN 2307-4531
(Print & Online)

http://gssrr.org/index.php?journal=JournalOfBasicAndApplied

A Fine Tuned Universal Language Model Fine-Tuning (ULMFiT) Approach for Airline Twitter Sentiment Analysis

Kevin Njagi[a]*, Zhang Zuping[b], Gilbert Marisa[c]

[a,b,c]*Central South University, Lushan road Changsha Hunan 410083,China*

[a]*Email: murimikevo@gmail.com*

[b]*Email: zpzhang@csu.edu.cn*

[c]*Email: gilbertmarisa161@gmail.com*

Abstract

More researches have shown that real-time Twitter data can be used to predict market movement of securities and other financial instruments. The Universal Language Model Fine-Tuning (ULMFiT) is a new approach which is based on training a language model and transferring its knowledge to a final classifier. We propose to fine tune the ULMFiT model by optimizing the parameters and training the model in a deterministic approach to increase the reproducibility. In this paper, we performed multi-class classification using Fine-Tuned ULMFiT, Naive Bayes, SVM, Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbors, on the Twitter US Airline data set from Kaggle. A model is built firstly for six major U.S. airlines that performs sentiment analysis on customer reviews so that the airlines can have fast and concise feedback. Recommendations is made secondly on the most important aspect of services they could improve given customers complains. Significant accuracy has achieved, which shows that our models are reliable for future prediction. Also, the accuracy of different models is compared, and results show that Random Forest is the best approach.

Keywords: Machine learning; Sentiment analysis; ULMFiT.

---

* Corresponding author.

1. Introduction

In recent years, Twitter has become the main online customer service platform. Hence a company's reputation on the Twitter website is of high value and importance especially in case for airlines. This is assuming the fact that commuters and travelers usually tweet their experiences from their previous flight. In fact, study has shown that replying to tweets has revenue generating potential, imbibes mores satisfaction than other customer service portals, and perhaps most importantly, satisfied Twitter users spread the word[14]. In this paper, we use the tweets to learn about people's flight experiences and give airline companies suggestions on how to make their trip more enjoyable.

The data set contains about 15,000 tweets, collected from February 2015 on various airline reviews from Kaggle website's already available dataset[?]. Every review is labeled as either positive, negative or neutral. First, we want to build a model to perform sentiment analysis on the data set. Our model is based on Universal Language Model Fine tuning model (ULMFiT) [11] for Text Classification .Our approach using fine-tuned ULMFiT improves the accuracy and the time it takes to achieve the results considerably.

ULMFiT is an effective transfer learning method to the NLP tasks. The ULMFiT method outperforms the state-of-the-art on six text classification tasks, reducing the error by 18 % − 24% on the majority of datasets. ULMFiT architecture consist of five layers, they are: embedding layer, LSTM1, LSTM2, ReLU, Softmax. Between the last LSTM2 and ReLU, ULMFiT uses a vector representation composed by last LSTM2 state, average, and max-pooling of the sequence of LSTM2 states. They use average and max-pooling to select the most important information in the sequence of LSTM2 states, but not always the most important information is the average neither maximum values.

Second, more interestingly, we want to assign a reason to each negative response, such as late flight, lost luggage, etc. In our data set, about 80% of the negative reviews has a negative reason label, yet the rests are labeled as" can't tell". Our goal is to assign a label to this unspecified group. By knowing every review's negative reason, we can give specific suggestions to different airline companies on how to improve their service.

2. Background and related work

Nowadays, developing and testing different models for a natural language processing problem is an interesting and challenging task. One of the significant challenges of the natural language processing approach is the massive data requirements. Transfer learning techniques deal with this problem transferring the knowledge already learned in a model and use it for another task in a new model. Therefore, deep learning applications rely on the use of transfer learning due it enables to train models in an entirely new problem with fewer samples. Transfer learning is a widely used technique in the computer vision community. In most applications, the image features are learned by the model in a general task such as object detection using the ImageNet [15] dataset and used in different domains. Some works as [16,17] fine-tune the last layer or several layers of a pre-trained model and use the pre-trained information in different contexts. The work developed in [18] study the impact of dataset complexity on transfer learning over CNN.

The most common form of transfer learning in NLP is to use pretrained word embeddings from well know methods like GloVe [19], Word2Vec [20], FastText [21]. Some works as [22,23,24] design a transfer learning method based on a multi-task learning architecture. Multi-Task architectures share weights of different tasks. Thus, the knowledge of different tasks is shared. A recent method which goes beyond transfer pretrained word embeddings is the Hypercolumns [25] that uses embeddings at different levels. Another approach to transfer learning in NLP is Fine-Tuning, this method consist of training a model in one task and then fine-tuning the model to the target task. In NLP, the Fine-Tuning approach presents good results on transfer information between similar tasks, as presented in [26,27].

In the field of airline service is similar to the product review sentiment analysis in such way that both are classifying the feedback of a purchasable product. However, there are still many differences in terms of domain focus and concentration of the result. For instance, classifying book reviews from Amazon, 'This book is worth reading.' is expressing a positive opinion regrading to the book. The word 'book' is a noun in this sentence, but in airline service the word 'book' has a higher chance to act as a verb, such as 'I am not be able to book this flight'.

Twitter platform for travel companies. In airline service domains, for instance, many airline companies have provided 24/7 customer service using Twitter, which increased the responsiveness especially when dealing with complaints and inquiries. Twitter also enabled companies to notify latest promotion products to customers and receive direct customer opinions regarding the deal when they retweet or comment on it, which is another advantage of using the Twitter platform. In this context, implementing automatic sentiment analysis techniques can be very effective comparing with using human resources when marketing analysis tasks are required for a specific promotion. Additionally, many consumers tend to consider the feedbacks and opinions posted by other Twitter users who have experience with the products and services before purchasing. This can significantly influence the consumers' decisions on which airline company they are about to choose.

Since tweets texts are usually short and verbal, the same problem presents in our data set as well. However, even though the tweets are short, there are strong indicative words. Specific words can be used as indicators for spam/ham emails and achieve good test accuracy. Therefore, we believe that tweets review, without many negating negatives, can be predicted well using the frequency vector representation and even better by using transfer learning. To prove this, we will use 9 different models.

3. Approach

Twitter, for the past decade, is one of the most used microblogging websites in the world. With Twitter, people can easily express their opinions for the whole world to see, if person allows for their tweets to be public. Twitter has even allowed a more direct contact between businesses and the public, such that they can advertise or speak directly to customers. Also, the consumer can share their feelings towards a specific company, whether it has positive, negative, or neutral sentiments. This can either help the company, or hurt the company depending on the tweet. Since there are millions of users on twitter, and billions of tweets tweeted a day, it becomes difficult to people to search through the tweet to understand the sentiment of its consumers [3]. However, we

can use machine learning to train and evaluate models to recognize the sentiment of a tweet, to help better and efficiently understand what the consumer wants.

3.1 Task

For our research, we are analyzing tweets related to various airlines to understand the people's sentiment when using their services. The dataset is used from Kaggle repository. We can see looking at the graphs below that most of our tweets have negative sentiments towards airlines with a 63%, while only 21% has neutral sentiment and 16% has a positive sentiment. Using these tweets, we first need to preprocess them to obtain the features we want that would affect the sentiment. Then we use a machine learning model, such as Naive Bayes, Random Forest to name a couple, and train the model using some of the preprocessed tweets. We will then use the fine-tuned ULMFit model that we propose in this paper. Finally, using the testing dataset of preprocessed tweets, we evaluate and compare our results to the predefined sentiment and obtain the models accuracy, precision, and recall rate. Once complete, we can finally compare each model and see which is best suited for twitter sentiment analysis.

3.2 Data Set

The sentiment analysis labels are positive (20%), negative (60%), and neutral (20%). The negative reason labels are bad flight (7.45%), canceled flight (9.62%), customer services issues (39.8%), damaged luggage (0.84%), flight attendant complaints (6.05%), flight booking problem (6.19%), late flights (1.99%), long lines (19.97%), and lost luggage (8.23%).

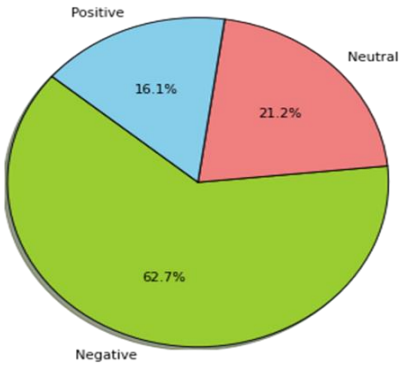

Figure 1: Pie chart for the percentage of sentiments in our dataset.

3.3 Preprocessing Steps

Before we can start training, the data must be preprocessed since there are words that are the same, but are written differently, such as WORD and word. They are indistinguishable to us, but to a computer they are not. We want to have as few, but necessary words as possible [4]. To do this, we use regular expressions (regex) on every tweet. While processing each tweet, we convert every character from upper case to lower case, convert URLs into the word URL, convert mentions or others username into the word AT USER, remove excess

whitespace, remove all instances of '#' only keeping the word following it and then stripping the tweet. We also eliminate character repetition such that when someone uses the word LOOOL, LOOOOOL, or however many 'O's they can fit, it will all be converted to 'LOOL' since it the intended spelling of some words require two of the same characters in repetition, such as the word 'hello'. This does mean that 'LOL' and 'LOOL' are two different features, but it would be extremely tedious to go through each feature and differentiate between the words that do and do not require the repetition and remove the unnecessary features.

When obtaining our features, we first preprocess, as previously mentioned, and then we tokenize each word for all tweets. Going through each word, we remove each character repetition and punctuation. Finally, we compare each word in the tweet with stop words from the NLTK corpus, AT USER and URL. The stop words are words with no sentimental value, such as the word the, and therefore have no reason for being a feature. Also, the only features that are accepted are those that start with a letter so that there are no nonsense words. Finally, after checking every word, we get our features. For the tweets we want to train, we use the mentioned process and attach the appropriate sentiment value to it.

3.4 Dictionary

The dictionary is made based on the training data and all sentences are broken down into list of words: (1) Delete common words such as a, an, to, etc. with high frequency but little semantic usage. (2) Stem words, such as "thanks" and "thank" as one word. (3) Delete low frequency words to reduce the size of dictionary for calculation efficiency.

3.5 Bags of Words Representation (Feature Extraction)

As features in our dataset is words i.e. text so we need to represent in numbers which can be used as feature for various models. Bags of words is used to represent the text. It is way to extract features from text to use in the modelling. It describes the occurrence of words within a text/sentence.

Three ways to represent in numerical features:

- Tokenizing gives an id for each possible token, for example white spaces and punctuation as token parameters.
- Counting the occurrences of words or tokens in each document.
- Normalizing and weighting the importance token and diminishing the weights of stop words

Samples are defined as follows:

- Each individual token occurrence frequency is treated as a feature.
- Vector of the features or tokens.

Vectorization is the general process of turning a collection of text documents into numerical feature vectors. In our project we have used 2 grams of representation to increase the accuracy.

For example: “This book is not good” will have almost similar score value as “This is good”. As each word is taken, to increase the probability we can take 2 words as single feature like “not good”, “is good” will give us score value different. Score value is the number to which each word is assigned to. Stop words have less value compared to other values.

4. Models

4.1 Naïve bayes

One of many ways to solve the twitter sentiment analysis classification problem is by using Naïve Bayes. Naïve Bayes uses Bayes Theorem, which for our classification problem, gives us:

$$P(label|features) = \frac{P(label)*P(features|label)}{P(features)} \qquad (1)$$

The algorithm assumes that all features are independent of one another and are hence naive. The Naive Bayes algorithm from the Natural Language Toolkit (NLTK) normalizes the numerator for each label rather than computing P (features). In short, using the tweet as input, the algorithm will determine from each word which is the most appropriate

We test our tweets by comparing each word with those features obtained from our training set and determining using the Naïve Bayes method to obtain the most likely sentiment. The data used has more tweets with negative sentiment than either neutral or positive, so it’s more likely that it will predict negative tweets correctly.

We can obviously see that with smaller test sizes, the predictions are more accurate, yet there’s not much of a difference. We can also see that both positive and neutral recall only slightly vary as the test size increases. Negative recall, positive precision, and neutral precision get worse as the test size increases, while negative precision overall improves. Given the fact that most of the tweets have a ground truth label of negative, it makes sense that such

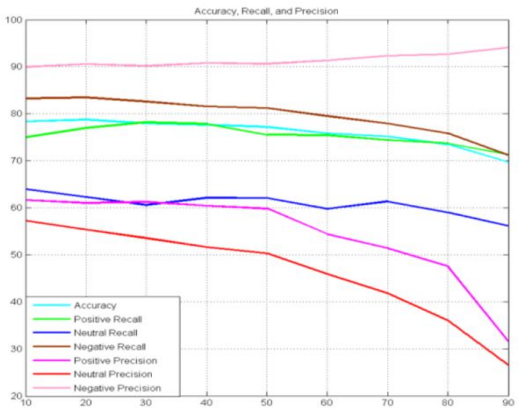

Figure 2: Comparing the results of different sized training and testing data to see how well it can predict the data correctly

Events occur with smaller training data. Naive Bayes has some difficulty predicting neutral tweets while easily guessing negative tweets, though this is most likely due to there being a bias. If the dataset contained an approximately equal amount of positive, negative and neutral tweets, then surely the Naïve Bayes algorithm would have better predictive capabilities.

4.2 Support vector machine

Support Vector Machine is a supervised machine learning classifier that can implemented for regression analysis as well as binary classification. We are here using the classifier in our work to separate out tweets into positive or negative sentiments. We are using RBF Kernel in our project. SVM uses the same input and implementation package as Naive Bayes. We then calculate the accuracy of the SVM model. Following is the equation of the line that classifies our model.

$$K(u,v) = u^T v = exp\left(\frac{-||u-v||^2}{2\sigma}\right) \quad (2)$$

The simple estimation of the decision boundaries to result as a function of the predictor variables is the main idea behind SVM. The input values are mapped on to one or other side of the boundary using this estimated line. The problem of overlapping classes, and the decision boundaries linearity is overcome by the SVM Classifier.

4.3 Logistic regression

Logistic regression is implemented to solve classification problems where we want to predict the result to be either 0 or 1. We usually use a sigmoid function as defined in the equation:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad (3)$$

We can see that even if both Naïve Bayes and Logistic regression are linear classifiers, Naïve Bayes is a considered to be a generative classifier that will try to predict the likelihood term under the assumption of conditionally independent between features, however, this assumption less happened in the real word problems, but by contrast, Logistic regression is a discriminative classifier that use a Logistic function to get the likelihood directly. In addition to this, Logistic regression also covers the case of a binary dependent variable. As our dataset will be classified in a positive and negative category, it is expected to have better performance than Naïve Bayes, which can be clearly seen in the conclusion section.

4.4 Adaboosting

Adaboosting is an ensemble technique which takes number of weak learners and attempts to create a strong classifier. It is used to boost the performance of decision trees. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. At each boosting iteration weights w (1), w (2), w(n) are applied to each of the training samples. Initial value of those weights is all set to w(I) = 1/N, the first step simply trains a weak learner on the original data. For each successive iteration,

misclassification rate also known as error is calculated and the sample weights are modified based on that rate and the algorithm is applied again to the reweighted data. At each iteration weights of incorrect samples which we got from previous step, whereas weights of correctly classified samples are decreased. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence.

Table 1: Parameter Tuning

n_estimators(default estimator)	Accuarcy
100	74.48770%
200	75.10246%
300	76.05874%
400	76.43442%
500	76.33197%

N_estimators controls the number of weak learners also known as decision stumps. Contribution of the weak learners in the final combination is controlled by learning rate. Base Estimator parameter is used to specify different weak learners.

N_estimators and the complexity of the base estimators are the main parameters to tune to obtain good results

Table 2: Adaboost with Decision as base estimator

N_estimator(decision tree)	Max depth of tree	Accuracy
2	15	~73.66%
10	15	~74.28%
100	15	~70.41%
10	14	~74.11%
10	16	~74.86%

As no of estimators is increased, accuracy also reaches till a point and it starts decreasing after a certain point in our case it is 300. As learning rate is varied, accuracy varies there is no pattern with the accuracy. Same is the case with max_depth. Major impact on the accuracy is due to N_estimators.

4.5 Gradient Tree Boosting

Gradient tree boosting is dependent on three below factors.

Loss Function: It can be squared error or logarithmic loss.

Weak learner for predictions: Decision trees are constructed in a greedy manner, choosing split points based on purity scores like Giny

An additive model: Decision trees are added in iterative manner and gradient descent procedure is used to minimize loss.

N_estimators: controls the number of weak learners

Learning_rate: it used to control the overfitting of the data.

Table 3: Training Gradient Tree with different values of parameter

N_estimators	Learning rate	Max depth	Accuracy
100	1.0	3	76.2978142077%
200	1.0	3	77.1516393443%
300	1.0	3	77.2540983607%
400	1.0	3	77.1857923497%
300	2.0	3	69.8428961749%
300	0.1	3	73.1315677889%
300	1.0	4	77.2540983607%
300	1.0	5	76.912568306%
100	1.0	4	76.6393442623%
100	1.0	5	75.9221311475%

As no of estimators is increased, accuracy also reaches till a point and it starts decreasing after a certain point in our case it is 300. As learning rate is varied, accuracy varies there is no pattern with the accuracy. Same is the case with max_depth. Major impact on the accuracy is due to N_estimators.

4.6 Decision tree

The Decision tree classification approach is a technique that making classification predictions by carrying out a series of true or false decisions. The decision tree approach is the foundation for the implementation of Random forest. A Decision Tree is a flowchart-like tree structure, in which each internal node represents a test on an attribute and each branch represents an outcome of the test, and each leaf node represents a class.

4.7 Statistical language model

The Statistical Language Model, also known as SLM, is a model used to estimate the distribution of natural language. The model uses probability distribution to show how frequently a string is encountered and, based on the strings; it will make a prediction on what word would come next. The big thing with natural language is that it is always

Changing Words and phrases used in a specific way can mean many different things. That is why we must look at the whole sentence structure and not just one word within a sentence to try and guess what the sentence is trying to convey. It assigns a probability to the whole sentence and also a probability to the likelihood of what

word will show up after a given sequence of words are given.

The model begins with getting the tweets and breaking it up into words. The words are then analyzed to see what words are commonly used within the batch of tweets. The commonly used words are used as stopping points as the model grabs list of words from the tweets.

Table 4: list of the most commonly used words within the tweets in the given file

Word	Frequencies	Word	Frequencies
Have	1746	That	1722
To	8627	The	6030
I	5393	Flight	4762
A	4457	You	4082
For	3975	On	3763
And	3693	My	3265
Is	2808	In	2521
It	2494	Of	2099
me	1908	Your	1899

The model removes the stopping words that have little to no effect to the sentence structure and it will make the data smaller to search through. This will also assign a probability to a sequence of words. The words that are important will have a higher probability next to ones that are not as important. The features part of the model will be a list of words that will get check to see what label it falls under. The labels are positive, negative and neutral. This model also uses MLE (maximum likelihood estimation) to calculate the probability of receiving a prediction based on what set of data it was given. Just knowing some predictions will help the model get a good estimate for a bigger sample size, but only if the predictions are normally distributed. The scores I got after running the model 6 times for each alpha value in table 5. The prediction value gets better when the change of alpha is smaller.

Table 5: Alpha value from point 0.1 to 0.0001

Alpha value						
0.1	98.6	93.54	94.52	94.63	95.21	94.9
0.5	97	88.17	92.45	92.54	87.33	89.88
0.01	98.8	94.78	95.02	95.59	96.66	95.66
0.0001	98.8	94.96	95.18	96.22	97	95.85

4.8 Random Forest

The random Forests regressor is implement for fitting many classifying decision trees on sub-samples of our data. This classifier is expected to improve the accuracy and control overfitting by averaging. The introduction

of the Random Forest Algorithm was developed as an approach to be built over decision trees. We use the bagging approach as described in the approach section. From a dataset D, we build each time an eligible Decision Tree, having N instances and A attributes, we build a subset d. This d is sampled along the with replacement techniques for the training dataset.

4.9 ULMFiT

Universal Language Model Fine-tuning for Text Classification (ULMFiT) is a Transfer Learning Method used in (NLP) [11]. ULMFiT pre-trains a language model using data from a general domain corpus. Then, it transfers the learned information from a language model to a text classifier. ULMFiT has three steps, present in figure 3: (a) LM pre-training, (b) LM fine-tuning and (c) Classifier fine-tuning.

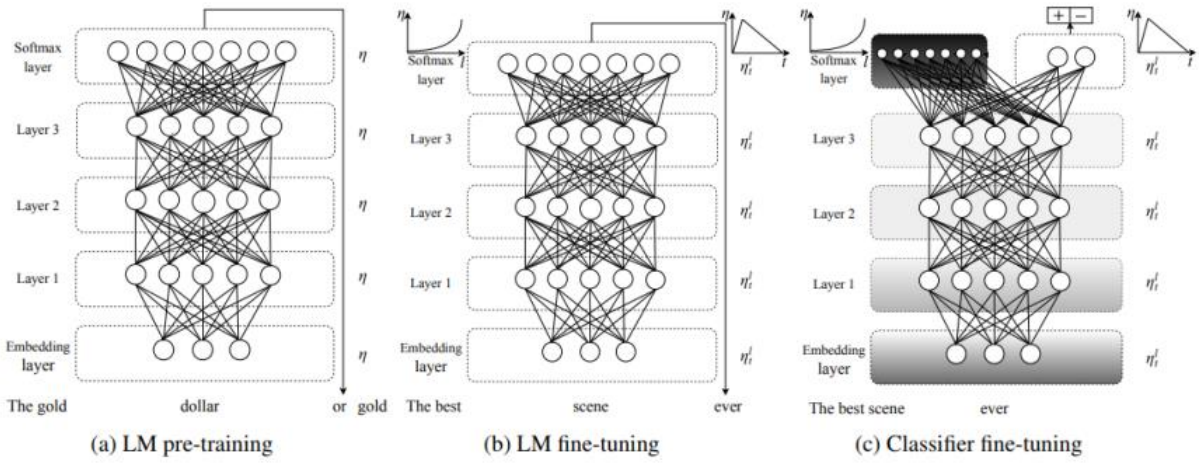

Figure 3: Step by step of ULMFiT

4.9.1 LM pre-training

The general domain language model is trained in the Wikitext 103 dataset composed by 25,595 pre-processed articles from Wikipedia and 103 million words written in English. The computational cost at this stage is relatively high but spent just once, and then used for other models.

4.9.2 LM fine-tuning

From the pre-trained LM, the fine-tuning of the language model using the dataset of the classifier. This is an essential part because the word distribution is different among different datasets, so it is necessary to do new training. The training phase is divided into two stages:

- Discriminative fine-tuning

Each layer uses a different learning rate. Initially, the learning rate of the last layer is defined, then is calculated for the previous layers according to the following

$$\theta_t = \theta_{t-1} - \eta . \nabla_\theta J(\theta) \quad \text{(i)}$$

$$\theta_t^l = \theta_{t-1} - \eta^1 . \nabla_{\theta^t} J(\theta) \quad \text{(ii)}$$

$$\eta^{l-1} = \eta^l / 2.6 \quad \text{(iii)}$$

According to Howard and Ruder in [11] the value 2.6 is defined after various experiments.

- Slanted Triangular Learning Rate

Initially, the maximum point of the learning rate is defined. The initial learning rate is the minimum value, and it has to grow until it reaches that maximum. After crossing that point, it starts to decrease. At the same time that the maximum value of the learning rate is set, the percentage of decreasing by epochs is also defined.

4.9.3 Classifier fine-tuning

Here, the final classifier is trained.

The authors remove the softmax layer of the language model. Then, they add a ReLU and softmax layer to solve the problem. Thus, the embedding layers and LSTM are pre-trained for the language model, so the ReLU and softmax layers are the only ones trained from scratch. Given a new sentence S, we gather its sentence representation H = [H1; H2; :::; Hn] obtained from the last language model LSTM and calculate its final representation using this equation

$$H_C = [H_n, maxpool(H), meanpool(H)] \quad (4)$$

The ULMFiT authors chose to use average and max-pooling layers because the essential part of a sentence may have only a few words, so we can lose information if we use only the last state Hn.

In our model we will use the following hyper parameters that are fine-tuned from the original model

We use the original language model available from the ULMFiT authors Howard and Ruder. It consist of an embedding size of 400 units, 3 layers with 1,150 units per layer and trained with a Backpropagation Through Time (BPTT) [12] with a batch size of 70. The authors used a dropout of 0.4, 0.3, 0.4 and 0.05 for the layers, RNN layers, input embedding layers, and output embedding layers, respectively.At the classifier step, we follow the original ULMFiT approach using a hidden layer of 50 units and an Adam optimization method with β1 = 0:7, β2 = 0:99. Our attention layer has 400 units, and we use a batch size of 48, a base learning rate of 0.004 and 0.01 for finetuning the LM and the classifier respectively. In the same way as the ULMFiT authors, we use 15 epochs to fine-tune the language model and 50 epochs for the classifier step. An important difference between our training method and the original ULMFiT is the use of a deterministic approach by setting the seed value to 40, and this is important to increase reproducibility. Following the original ULMFiT approach, we use bidirectional language model [13] too. It consist of fine-tune a forward (original sequence of sentence S)

language model and a backward (sentence S inverted).

5. Discussion

A general conclusion is that classifiers' results vary according to the dataset, however, it is noticeable that kNN, RF, and DT were almost better than other classifiers on most of the experiments. Another major conclusion is that feature selection (levels 1 and 2) almost enhanced every performed experiment on any dataset, except for rare cases where using the whole feature set was better (i.e. Naïve Bayes, and Adaboost on the "American" dataset).

The fine-tuned ULMFiT model has provided its best performing hyperparameters to achieve best performance. While this parameters may work on other datasets they were not the best for our dataset and we decided to fine tune the model to fit out our dataset. We explored three regularization techniques: we fine-tuned the base learning rate of the language model, we also fine-tuned the learning rate of the classifier, and finally on our training model we change the seed value to enhance replicability. Based on our results we found our model produced better results than the standard ULMFiT model and the model is not overfitting on dataset because of the dataset imbalance.

6. Precision

Table 7: Accuracy in the two-class dataset

Model name	**accuracy**
Fine-tuned ULMFiT	0.931
Random Forest Classifier	0.809
Ada Boost Classifiers	0.785
Gradient Tree Boosting	0.765
Decision Tree Classifiers	0.757
Logistic Regression	0.645
SVM	0.645
Naïve Bayes	0.572

7. Conclusion

In this paper we have studied the sentiment analysis based on the feedbacks of travelers regarding airline companies. Our proposed approach showed that both feature selection and over-sampling techniques are equally important about to boosting our results. The use of the ULMFit technique has returned the best subset of features and reduced the computations needed to train our classifiers. Whereas, SMOTE has reduced the skewed distribution of the classes found in most of our smaller datasets without causing overfitting. Our results are a compelling evidence that the proposed model has high classification accuracy in predicting instances form the three classes (Positive, Negative, and Neutral).

As can be seen, some of the applied classifiers have outperformed the others. For example, Random Forest and Decision Tree have shown a high prediction level, and stability when applied on all datasets. While K-NN and Linear SVM have shown an acceptable level of performance regarding all the evaluation metrics. On the other hand, Kernel SVM has shown poor results in comparison with other classifiers

This research has had some limitations since the model had a problem in overfitting the model when applied on this dataset over-fitted when applied on large epochs at such we had to use the set no of 15 epochs in our study. This could be a big problem if we wanted to train the model on a large dataset and it over-fitted, as this would provide the results but miscalculated results.

8. Recommendations

Apart from measuring the performance of different models, in future, we plan to extract what kind of works exactly enables us to predict the emotions and sentiments of what people share as Tweets, more accurately. We also would like to have hyperparameters for each model during each iteration for fine tuning. Additionally language modelling could also be included, instead of using the AWS_LSTM another pre-trained model could be used, maybe one more specifically trained for tweets or sentiment. Data preprocessing could also have been done differently.

References

[1] Esi Adeborna, Keng Siau, “An Approach To Sentiment Analysis-The Case of Airline Quality Rating,” PACIS 2014 Proceedings. Paper 363.

[2] Hung T. Vo, Hai C. Lam, Duc Dung Nguyen, Nguyen Huynh Tuong, “Topic Classification and Sentiment Analysis for Vietnamese Education Survey System”, Asian Journal of Computer Science and Information Technology 6:3, May (2016).

[3] Soumi Sarkar, Taniya Seal, “Sentiment Analysis- An Objective View”, Journal of Research, Volume 02, Issue 02, April 2016.

[4] Ann Devitt, Khurshid Ahmad, “Sentiment Analysis and The Use of Extrinsic Datasets in Evaluation,” International Conference on Language Resources and Evaluation,2008.

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J Mach Learn Res 15, 1929-1958. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[6] Ian Goodfellow, Yoshua Bengio & Aaron Courville (2016). Deep Learning. MIT PressK. Elissa,

[7] Zagoruyko, S., & Komodakis, N. (2016). Wide Residual Networks. Proceedings of the British Machine Vision Conference 2016. doi:10.5244/c.30.87

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90

[9] Scikit-learn: http://scikit-learn.org/stable

[10] http://www.itl.nist.gov/div898/handbook/apr/section4/apr412.html

[11] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, 2018, pp. 328–339

[12] M. C. Mozer, "A focused backpropagation algorithm for temporal," Backpropagation: Theory, architectures, and applications, p. 137, 1995

[13] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semisupervised sequence tagging with bidirectional language models," arXiv preprint arXiv:1705.00108, 2017.

[14] S Kamal, N. Dey, A.S Ashour, s. Ripon, V.E. Balas and M. Kaysar, "FbMapping: An automated system for monitoring facebook data",Neural Network World, 2017.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. Ieee, 2009, pp. 248–255.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in International conference on machine learning, 2014, pp. 647–655.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.

[18] M. D. d. S. Wanderley, L. d. A. e Bueno, C. Zanchettin, and A. L. Oliveira, "The impact of dataset complexity on transfer learning over convolutional neural networks," in International Conference on Artificial Neural Networks. Springer, 2017, pp. 582–589.

[19] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.

[21] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

[22] M. Rei, "Semi-supervised multitask learning for sequence labeling," arXiv preprint arXiv:1704.07156, 2017.

[23] L. Liu, J. Shang, X. Ren, F. F. Xu, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[24] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint manytask model: Growing a neural network for multiple nlp tasks," arXiv preprint arXiv:1611.01587, 2016.

[25] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in Advances in Neural Information Processing Systems, 2017, pp. 6294–6305.

[26] S. Min, M. Seo, and H. Hajishirzi, "Question answering through transfer learning from large fine-grained supervision data," arXiv preprint arXiv:1702.02171, 2017.

[27] A. Severyn and A. Moschitti, "Unitn: Training deep convolutional neural network for twitter sentiment classification," in Proceedings of the 9th IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019 - 6 - paper N-20204.pdfinternational workshop on semantic evaluation (SemEval 2015), 2015,pp. 464–469.