Application of Machine Learning to Classify News Headlines

Pavani Guttula, Reham Mohamed Aburas, Sonali Srijan

pguttula@purdue.edu,raburas@purdue.edu, ssrijan@purdue.edu

Massive amounts of news data are added to the web every day, which increases the necessity to analyze and extract valuable information from this data. Categorizing news data is a crucial task for providing more organized and easy access to news articles. However, this task is still not in the mainstream. Moreover, a long history of news data can be analyzed to detect the positive and negative news, and make predictions about the future. In this project, we plan to leverage machine learning techniques to analyze news data and investigate their applicability to tackle these problems.

1 Project Summary

1.1 Introduction

Curiosity of humans to learn and share information about things happening world-wide paved way for providing news through multiple channels like broadcasting, electronic communication, newspapers etc. Examining the events that unfolded over a specific period of time, one can see all the important episodes

that impacted their life- be it economy, natural disasters, opportunities or crimes. In general, people tend to read news/daily updates about topics that interest them or relevant to them. Hence people would prefer having an option to choose the categories of news one would want to follow. We want to provide a solution to address these two predilections of people. However, every light has its shadow. Exploiting the human desire to stay updated, a lot of fake news gets circulated for various nefarious purposes. We also plan to address this issue. We chose a dataset that provides birds eye view of the happenings of Europe for the past two decades. This news dataset from Kaggle (`https://www.kaggle.com/therohk/ireland-historical-news`), has 1.43 million rows of data giving us an ample opportunity to explore various dimensions of news as it is consumed across the Europe continent. The dataset has information about headline of a news, the day the headline got published and what category the news belongs to. In this project, we performed NLP on the source data, trained deep learning models like CNN, LSTM and ensemble machine learning models like Random Forests to predict which category a certain headline belongs. We also performed sentiment analysis on the news dataset to predict if a given headline is a positive news or not. We are also interested in understanding the trends of news in various categories and how they evolved over time.

1.2 MOTIVATION

With increasingly large volumes of news available, it is difficult for users to access categories of their interest if articles are unlabeled. Categorization of news into sub-sections allows easier navigation to readers, saves time and effort, and also allows news websites to recommend relevant articles or advertisements [1]. Multi-label classification is thus, a relevant problem. Also, having a model trained to classify headlines would help tag uncategorized data to a relevant category.

Analyzing the nature of news articles and the underlying emotional tone behind a string of words offers an insight into public opinion related to the topic in question. It is extremely useful in social media monitoring and can predict a number of things like shifts in the stock market, probable results of elections, etc, with shifting mental attitudes [2, 3]. Using the training dataset, we would predict the positive or negative connotations of news articles. Having this information

would also allow us to understand how a certain category of news has evolved over time. We can answer questions like are we seeing a positive shift of events in a certain category?

2 Literature Survey

Classification of text has been a complex problem and there has been significant amount of work done in this domain. Deep learning models, Recurrent Neural Networks for example, have shown good potential. Basenet et al. [4] for example, demonstrated better performance of Long Short Term Memory in document classification in comparison to other machine learning algorithms. The 20 newsgroups data set was used for the experiment with around 20,000 documents. They used TFIDF for feature extraction and used maximum features as 5000 for the documents. Similarly, Kim [5] showed that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results for single-sentence classification. Both CNN and LSTM have been combined in one model in [6] which uses a C-LSTM model for text classification. The idea is to capture both local features of phrases as well as global and temporal semantics of a sentence. They tested their model on sentiment classification and question classification tasks using Stanford Sentiment Treebank (SST) benchmark [7] and TREC benchmark [8] respectively. They compared with several approaches including CNN based models and bidirectional LSTM based models, and showed that their model has the best accuracy.

Sentiment analysis for online news have been also a hot research topic recently. In [9], sentiment analysis was classified based on the structure of the data into three categories. We focus on unstructured sentiment which is based on free text. In [10], sentiment analysis for financial market based on news is provided using classical SVM classifier with parameters optimised through particle swarm optimisation (PSO). The authors show that this approach performs better than deep learning models based on the financial news dataset. A hybrid approach for sentiment analysis on news comments have been proposed in [11] which involves using sentiment lexicon for polarity detection, then using the results from the lexicon based method to train machine learning algorithms, such as

KNN and SVM. We would try classical ML approaches as well as deep learning on our tagged data to compare their performance.

3 Project Methodology

3.1 Dataset Description

3.1.1 Original Dataset

Dataset we obtained from kaggle has 1.42 million news headlines published by The Irish Times based in Ireland. Daily headlines of the newspaper starting from 1996-01-01 to 2018-12-31 gives a long term birds eye view of the happenings within Europe. This kaggle news dataset has it's news headlines divided into 156 categories. Category labels include News, Sports, Business etc. Treemap in the figure 1 shows the distribution of the 1.42 million headlines into 156 categories.

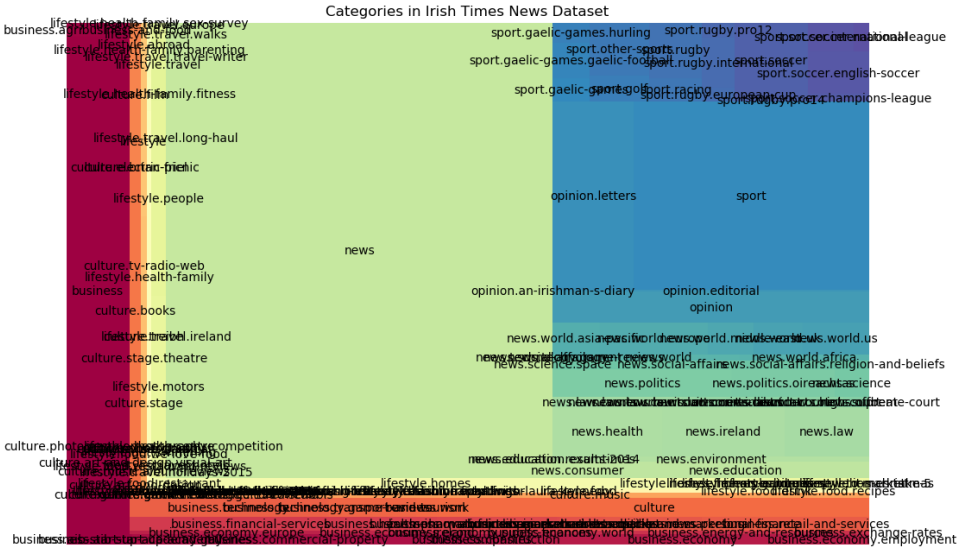


Figure 1: Distribution of Headlines in the dataset

As we can clearly see, most of the data is divided into "news" category making it very skewed. One of our major challenges in this project was to reduce the

number of categories and make the data more balanced.

We followed three approaches to reduce the number of categories, in order to achieve better accuracy. We define three datasets for the three approaches as follows:

1. Dataset A: For this dataset, first we filter all categories with less than 1000 rows. Then, we use the roots of the labels hierarchy and merge all the labels under the root. For example, all the labels news.education, news.law, etc, are all merged into news label. By doing this, we get **6 categories** in total. Since the data is unbalanced, we downsample the data so that each category has 55,000 rows.

2. Dataset B: We applied this approach to get more fine-grained categories. We found that among the 156 categories, only few of them have large number of rows and the rest have relatively very few rows. So, we filter all rows with less than 10,000 rows. The result is **19 categories**. Again, we downsample the data to 10,000 rows for each category to solve the unbalanced labels issue.

3. Dataset C: For this dataset, we manually merge related labels. Our goal was to make more use of the data samples and labels we have. This approach resulted in **21 categories**. For each category we downsample to 12,000 rows.

3.1.2 Dataset for sentiment Analysis

1. Dataset a
   To perform sentiment analysis, we initially manually labelled around 1100 headlines as positive, negative or neutral.Out of the 1100 rows of data there are 463 positive labels, 384 negative headlines and 264 Neutral headlines. Treemap in 2 shows the distribution of the three labels.

2. Dataset b
   Once we had initial set of results with Dataset a, we wanted to label more

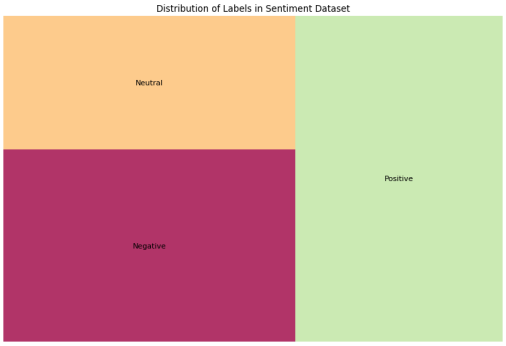


Figure 2: Distirbution of sentiments in the dataset a

data and see how if affects the performance of labels. We have 1544 labeled rows of data now. Comparison of Datasets a and b would be detailed in the results section. Out of the 1100 rows of data there are 601 positive labels, 525 negative headlines and 418 Neutral headlines. Composition of positive, negative and neutral labels in the extended dataset is visually represented in the treemap 3.

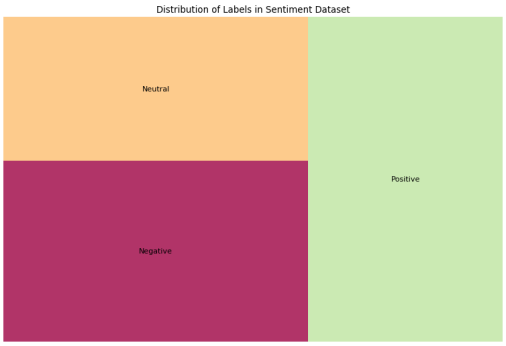


Figure 3: Distirbution of sentiments in dataset b

3.2 PREPROCESSING

Our news dataset is composed of English text headlines of news and category the headline belongs to. We would use the headlines as our features to perform different machine learning tasks as we show in the rest of this section. Therefore, we would start by applying some preprocessing steps to filter the text data and make it more suitable for prediction tasks.

The main preprocessing tasks applied to natural language text is lemmatization, stopwords removal and tokenization. We used nltk library's WordNetLemmatizer function to implement Lemmatization. Lemmatization aims at returning words to their dictionary form known as lemma by removing inflectional endings of words. This would help achieving a normalized form for each word, so that different inflectional forms of the same word are grouped and mapped to this normalized form. Stopwords are words that do not affect the semantics of text and therefore are highly likely to appear in any statement. For example, the articles, prepositions, verb to be, are all examples of words that are common in all English text. Therefore, filtering these words would reduce noise and improve the accuracy of ML tasks. Gensim[12] library provides methods to remove stopwords. Finally, tokenization is applied to chop each sentence into tokens or words which are then used to extract features.

In order to apply machine learning techniques on natural language text, we need to map the text into feature vectors. Recently, most of NLP research have been using word embeddings to represent text, where each word or token is mapped into a numerical vector that represents the semantics of this word. Therefore, words that have similar meaning should have closer word vectors. There are several algorithms for word embedding representations. Word2Vec[13], GLoVe[14],tf-idf[15] are some of the most popular techniques. Each headline text could be represented as an aggregation or concatenation of its word vectors. We talk more about how we performed experiments using few of these popular embeddings in the later sections.

3.2.1 STRATIFIED SAMPLING

Since we have a rather unbalanced dataset in terms of category divisions, we opted for stratified sampling[16] to ensure each of our datasets training, testing and validation has all the subcategories included. During stratification each headline is divided into homogeneous subgroups called stratum before sampling. Then simple random sampling is applied within each stratum so that sampling of the dataset is more balanced. In our project, we first perform stratified sampling on the entire dataset to divide it into 60 % of training data comprising of all the categories and we again perform stratified sampling on the remaining 40 % of the dataset to divide in into validation and test datasets each comprising 20 % of original dataset.

3.3 NEWS HEADLINE MULTILABEL CLASSIFICATION

In this task, we want to classify the news dataset into categories based on the news headline text. We primarily use deep learning for this task as discussed in the literature review. To apply deep learning, first we need to vectorize the text data, so that each headline sequence can be represented as a feature vector. For this purpose, we use word embeddings technique which maps each word to a feature vector that represents the semantics of this word. We have two approaches for word embeddings: the first one is to train our own word embeddings using the vocabulary in our dataset. The second is to use a pretrained word embeddings based on a large dictionary of words, that would include most of the news dataset vocabulary.

For the pretrained word embeddings, we use GLoVe [14] which uses unsupervised learning algorithms to map each word to a word vector space. We use Common Crawl pretrained dictionary, which contains 6B tokens. Each word is mapped to a 100D vector.

For deep learning, we use python Keras library with Tensorflow backend. We start each deep learning model with an Embedding layer. This layer takes as input the integer sequence from the tokenizer and outputs a vector that represents the embeddings features vector for this sequence. This layer is trained as part of

the deep learning model to learn the embeddings.

To use the pretrained embeddings, we use GLoVe dictionary as the weights of the Embedding layer and mark this layer as untrainable.

On top of this layer, we design three deep learning approaches based on literature. One of the is based on 1-dimensional CNNs and the second is based on bidirectional LSTMs. The third approach is a combination of 1D CNN and bidirectional LSTM. For comparison purposes, we also train regular ML classifiers as Random Forests.

3.3.1 1-DIMENSIONAL CNN MODEL

This model consists of three CNN 1D layers. The CNN layer is composed of 1D filters applied on the word embeddings to extract higher level features. The layers are formed of 128, 64 and 64 filters respectively. Each filter is of size 5 and uses ReLU activation function. Since CNNs are sensitive to features locations, we add MaxPooling layers after the last CNN layer, which down samples the feature maps of the CNN. This makes the network less sensitive to features position changes.

Finally, we add a Dense layer with Softmax classifier that outputs a probability for each class label. The label with highest probability is selected as the correct label. A visualization for the model can be found in Figure 11.

3.3.2 BIDIRECTIONAL LSTM MODEL

LSTMs are commonly used in text classification where it takes into consideration the sequence of words in the sentence. A Bidirectional LSTM gives output for each node based on the past and future states. So for our model, the output for each word is trained based on the preceding and succeeding words in the sentence.

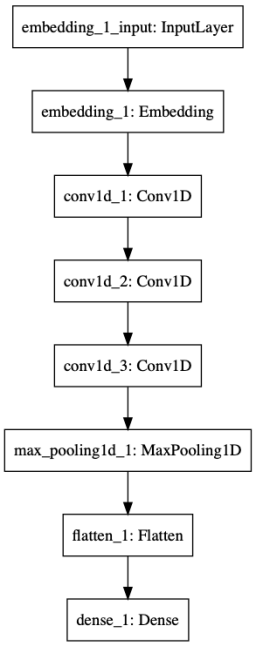


Figure 4: 1D CNN model

We use 3 bidirectional LSTM layers, each with a hidden size of 100. A dropout layer of size 0.3 is used to decrease overfitting on the training data. Finally, a dense layer with softmax classifier is used similar to the CNN model. A visualization for the model can be found in Figure 5.

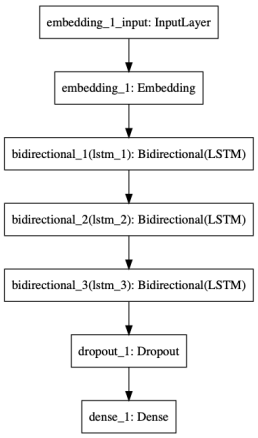


Figure 5: LSTM model

3.3.3 1-D CNN + BIDIRECTIONAL LSTM

The CNN are commonly used with images for extracting spacial features. Bidirectional LSTMs are used for extracting temporal features, where it considers the history and future of features. One way to extract both types of features is to combine CNN and LSTM layers in one model. As shown in literature, this combination was used to improve deep learning accuracy for NLP tasks.

We build a model using one 1D CNN layer with 128 filters of size 5 and three bidirectional LSTM layers each with hidden layer of size 100. Again, we use a dense layer with softmax classifier for the classification task. The model details is shown in Figure 6.
For training each model, we use the categorical crossentropy as the loss function. Categorical crossentropy compares the distribution of the model output predictions with the true distribution. We represent the labels of the dataset by one-hot

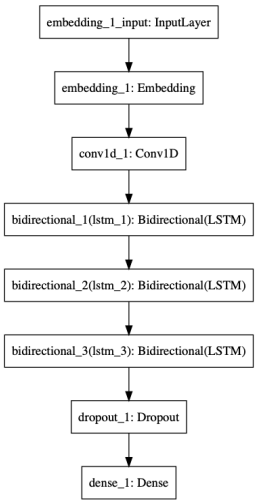


Figure 6: 1D CNN + LSTM model

encoding scheme. The closer the labels to the model's output, the lower the loss function. The categorical cross entropy is defined as:

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N} y_{ij} * log(\hat{y}_{ij}) \tag{1}$$

Where y_{ij} is the value of label i in the onehot encoded vector of the data point j. And $\hat{y}_{ij}$ is the corresponding predicted value.

We use the Adam optimizer which has recently shown advantage over commonly used optimization algorithms, such as Adagrade and RMSprop.

3.3.4 Random Forests

We also implemented a Random Forests model consisting of a spectrum of decision trees. To apply Random Forests, we used one TF-IDF embedding after the necessary preprocessing steps. The splitting attribute for branching was decided by maximum Gini gain. Gini index is defined as:

$$Gini = 1 - \sum_{i=0}^{C} p_i^2 \tag{2}$$

Gini gain measures decrease in gini index after split.

3.4 Sentiment Analysis

Since our source data only gives us an information on the publish date of news, headline and the category the news belongs to, to perform sentiment analysis we wanted to manually classify a headline to positive or negative or neutral. In our proposal, we mentioned that we would manually label atleast 1000 headlines under category "news". However, after taking a closer look at the dataset we realized that although "news" category headlines contribute to 40% of the dataset, the headlines are highly skewed towards negative sentiment. Hence, we decided to label headlines under "business" category which appeared to be more balanced. Figure 7 shows the most prominent words in the headlines related to business.

Figure 7: Prominent words in Business category

3.4.1 SAMPLING AND SPLITTING OF DATASET

To make the dataset more balanced, out of 1100 rows of dataset a, we picked 350 rows of each category and for dataset b we picked 450 rows of each dataset. With an 60-20-20 train-validation-test split with sampling on both Datasets a and Datasets b, we ran multiple machine learning and Deep learning models to perform sentiment analysis.
To train our ML models to perform sentiment analysis we used both TF-IDF and word2vec embeddings to vectorize headlines. We talk more about how models performed with each of the embeddings in detail in results section.
Embedding: TF-IDF
For Machine Learning models, we used TF-IDF[15] embedding to convert the tokenized words to vector form. TF-IDF reflects how important a word is to a document in a corpus.
Embedding: Word2Vec
For Machine Learning models we also used Word2vec skip gram embedding that takes as its input a large corpus of text and produces a vector space with context, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.
We will discuss the performance of multiple models with various embeddings in the results section.

3.4.2 LOGISTIC REGRESSION

For Logistic regression algorithm, considering we have a multi-label classification at hand we used sklearn's LR model with "lbfgs" solver and maximum iterations set to 1000. We used L2 regularization to penalize loss function.

3.4.3 SUPPORT VECTOR MACHINE

Since we have a relatively small dataset, to implement SVM we used sklearn's SVC model with one vs one decision classifier.SVM model with linear kernel gave us a much better performance compared to rbf we were using earlier.

3.4.4 RANDOM FORESTS

Wee implemented this emsemble model with a spectrum of dofferent number of decision trees. The splitting attribute for branching is decided by maximum Gini gain.
We also implemented Deep Learning models 1-Dimensional CNN and LSTM using Glove,Multi-Layer Perceptron using word2vec,TF-IDF embeddings. Embedding: Glove
For Deep Learning models we use a pretrained word embedding GLoVe [14], which uses unsupervised learning algorithms to map each word to a word vector space.

3.4.5 1-DIMENSIONAL CNN MODEL

Using GLoVe[14] we build an embedding matrix that will map each unique word in our corpus to a 100D vector. On top of the embedding layer formed using embedding matrix, we have two 1 Dimensional CNN layers. The CNN layer is composed of 1D filters applied on the word embeddings to extract higher level features. Each layer is formed of 128 filters of size 5 and uses ReLU activation function. Since CNNs are sensitive to features locations, we add MaxPooling layers between the CNN layers, which down samples the feature maps of the CNN. This makes the network less sensitive to features position changes. Finally, we add a Dense layer with Softmax classifier that outputs a probabilityfor each class label. The label with highest probability is selected as the final sentiment for the headline. 1-D CNN model visual representation is shown in Figure 11.

3.4.6 MULTILAYER PERCEPTRON CLASSIFIER

A multilayer perceptron (MLP) is a simple feedforward neural network with multiple layers of perceptrons. With a hidden layer size of 128, we used l2 regularization to penalize the loss function and stochastic gradient descent to minimize the loss function. We used multiple activation functions like relu,tanh for comparision purposes.

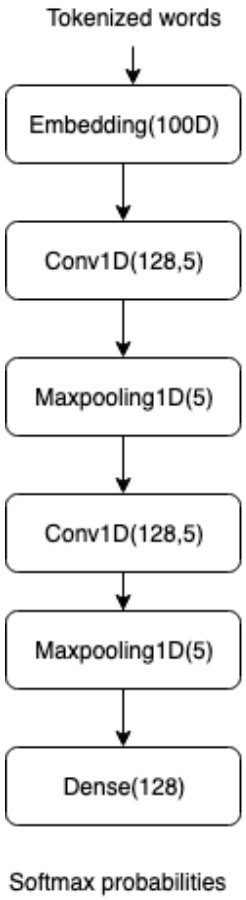


Figure 8: 1D CNN model

3.4.7 BIDIRECTIONAL LSTM

We used a simple two layer bidirectional LSTM model each with a hidden layer of size 100 to predict sentiments. Embedding matrix of 100D vectors for each unique word in corpus is fed to two layers of LSTM. Output of the second layer of LSTM is fed to a softmax classifier to predict the final label.

4 EXPERIMENTAL EVALUATIONS AND OBSERVATIONS

4.1 MULTILABEL CLASSIFICATION

We split the data into training, validation and test sets with 60-20-20 split. We evaluate our models using the three datasets as discussed in Section 3.1.1. We tune each model using the validation set and train for 20 epochs. We use the GLoVe 6B embedding with 100D as the default embedding. Table 1 shows the results for each model.

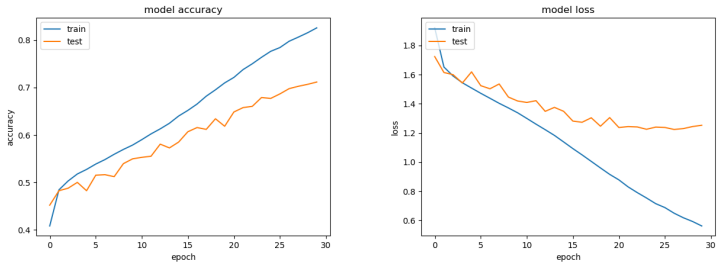
The table shows that for the deep learning models CNN+LSTM gives the best accuracy. Surprisingly, the random forest model outperforms the deep learning techniques. The best accuracy can be achieved with 6 categories (RF) is 77%, while with 19 categories (RF) we can still achieve a good accuracy up to 75%.

Model Name / Accuracy(%)	**DS-A (6 cats)**	**DS-B (19 cats)**	**DS-C (20 cats)**
1D CNN	66	49	47
Bidir-LSTM	73	61	55
1D CNN+LSTM	74	71	49
Random Forests	76	74	65

Table 1: Multilabel Classification results for various algorithms

To reduce overfitting the training data, we show the accuracy and loss of the model with increasing the number of epochs in Figure 9. In this figure, we use DS-B with 1D CNN+LSTM model. The loss almost stabilizes after 20 epochs

for the validation data while it still decreases for the training data which shows that after that the model would overfit the training and will not perform well on validation data.



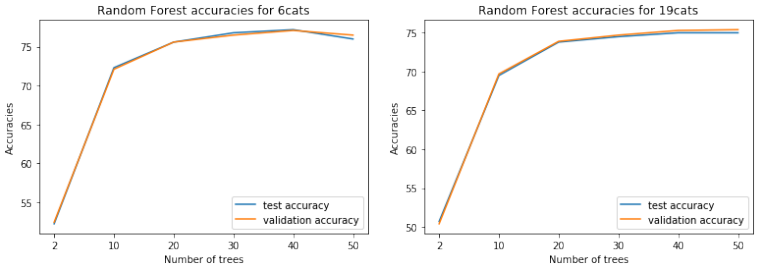
(a) Accuracy vs Epochs (b) Loss vs Epochs

Figure 9: Accuracy/Loss vs Epochs for 1D-CNN+LSTM models

We experimented with different number of trees in the random forest model, as shown in fig 10. Interestingly, for all the 3 datasets (6 categories, 19 categories and 20 categories datasets), we found that the optimum number of trees was 40. The maximum number of features to consider when looking for the best split was set equal to $\sqrt{numberoffeatures}$.
For 1D CNN+LSTM model on the test set using the three datasets, The results of the test data is very close to the validation showing 70% accuracy for 19 categories dataset.

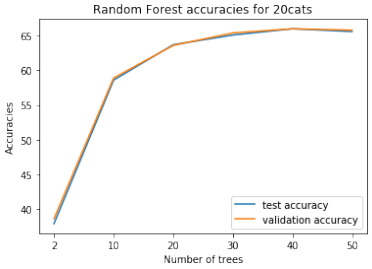
4.2 Sentiment Analysis

With parameter tuning,initially with dataset a using word2vec, tf-idf embeddings we achieved the accuracies on the validation and test datasets as stated in Table 2. We made multiple experiments to achieve the current results. Initially we went for the word2vec embedding approach and the prediction accuracy was around 39%. We realized it is as good as a random experiment considering that we have 3 labels. Hence, we tried multiple other word embeddings and finally decide to use tf-idf embedding. We could clearly see an improvement in results when we ran algorithms on the dataset which is symmetric when compares to the dataset which was slightly more skewed towards positive values.



(a) Accuracy vs number of trees for 6-category dataset

(b) Accuracy vs number of trees for 19-category dataset



(c) Accuracy vs number of trees for 20-category dataset

Figure 10: Accuracy vs Number of trees in RF model

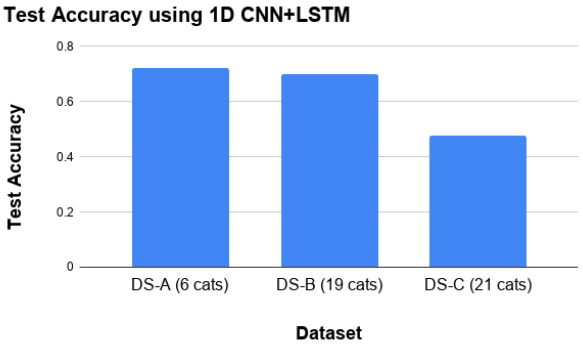


Figure 11: Test set accuracy on 1D CNN+LSTM

Model	**Validation**		**Test**	
	Word2Vec	TF-IDF	Word2Vec	TF-IDF
MLP Classifier(Relu)	47	52	40	51.3
MLP Classifier(Tanh)	41	46	45	42
Logistic Regression	38	46	46	55
Support Vector Machine	38	51	44	51
Random Forests	36	44	46	44

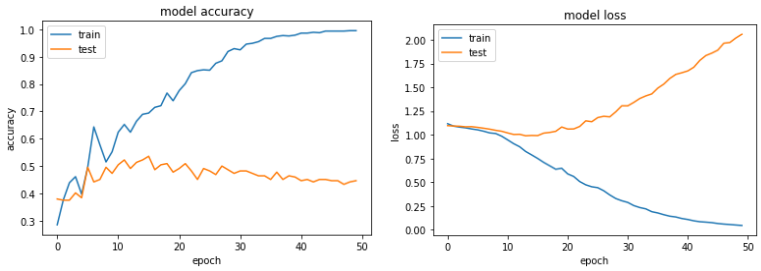
Table 2: Sentiment Analysis results with Word2Vec & TF-IDF embedding for Dataset a

Model	**Validation**		**Test**	
	Word2Vec	TF-IDF	Word2Vec	TF-IDF
MLP Classifier(Relu)	52	45	45	62
MLP Classifier(Tanh)	50	48	46	68
Logistic Regression	52	56	41	50
Support Vector Machine	52	57	42	54
Random Forests	46	54	42	54

Table 3: Sentiment Analysis results with Word2Vec & TF-IDF embedding for Dataset b

However, we wanted to observe how the model performances would be impacted with an increase in the training data. With an additional 500 rows of labelling, we saw an improvement in results. We believe more training data would have improved our accuracies by a marginal amount. Model performances on the new dataset with word2vec,tf-idf embeddings are shown in 3.
For CNN and LSTM we achieved best accuracies with Glove word embedding. In 4, we present results for both CNN and bidirectional LSTMs for Datasets a and b. We can see that the test data accuracy is better for Dataset b which has more supervised data. Also, with an increase in epochs, we observed an improvement in CNN predictions. We also observed that Bidirectional LSTM performs better than undirectional LSTMs since the model gets more context of headlines with bidirectional LSTM.

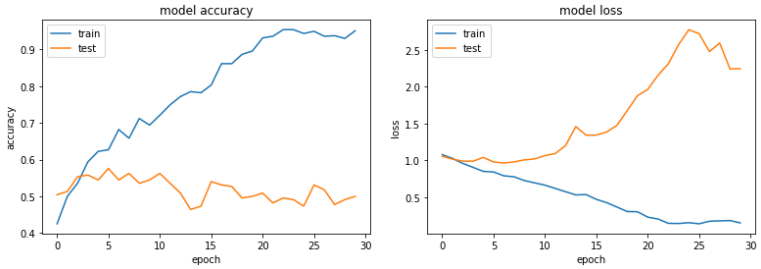
Figure 12 (a) shows how training accuracy keeps increasing with an increase in the number of epochs. We can clearly see how validation accuracy increases till the model starts overfits the training data. Figure 12 (b) shows the how validation loss increases after a point due to overfitting of training data. Training loss keeps decreasing as our cnn model fits with training data more and more.



(a) Accuracy vs Epochs (b) Loss vs Epochs

Figure 12: Accuracy/Loss vs Epochs for 1D-CNN models

13 shows similar patterns in accuracy and loss with increase in epochs as explained above.



(a) Accuracy vs Epochs (b) Loss vs Epochs

Figure 13: Accuracy/Loss vs Epochs for LSTM models

Model	Validation		Test	
	Dataset a	Dataset b	Dataset a	Dataset b
1-Dimensional CNN	52	52	47	62
Bidirectional LSTM	61	56	47	54

Table 4: Sentiment Analysis results with Glove embedding for Datasets a and b

4.3 Sentiment Analysis: Time Series

To predict the trajectory of sentiment for business news, we used the trained MLP classifier to make sentiment predictions for the unlabelled business data. In doing that, we observed the following trend (y axis representing the proportion of positive, negative and neutral news) illustrated in Fig 14:

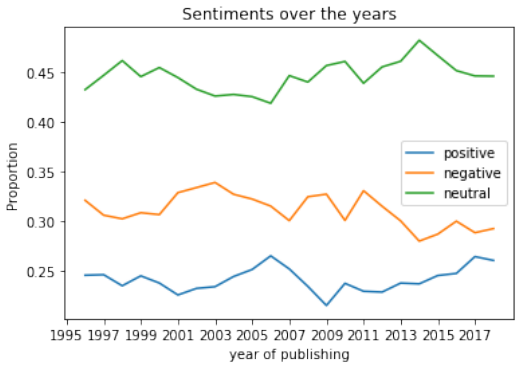


Figure 14: Time series analysis of Sentiment

There's a couple of notable things that we can observe here. Firstly, in the year 2009, we can see that the proportion of positive business news was the lowest for the entire spectrum of time-frame. This clearly captures the trend, as 2009 was the year of a global economic recession, where the world markets were severely affected by a financial meltdown.
Secondly, we can see that the proportion of positively labelled business news has continued to rise since 2009.

5 Conclusion

5.1 Multilabel Classification

We obtained our dataset from kaggle and there were few existing kernels where 60 % accuracy was achieved on test data. When we started out this project, our initial aim was to improve on this accuracy. As we digged more into the dataset we realized how skewed the dataset is towards certain categories as shown in 1. The 60 % accuracy shown on kaggle kernels can be easily obtained if the model just pushes all the headlines into news and business category which is not the classification of headlines we want to achieve.
As stated above we merged the headline categories in multiple ways and down-sampled the necessary categories so that we have equal representation of headline categories. We were able to achieve not only better accuracy than kaggle kernels but also better classification.
One another major learning for us was how important sampling is to get better predictions. We were initally getting validation accuracy as low as 20 % and when we investigated the issue we realized numpy random sampling is not able to shuffle the dataset properly. Our training dataset had for examples categories 1,2,3 and we were trying to validate data with categroies 3 and 4. We implemneted stratified sampling inorder to solve this problem. This improved our validation accuracy by three times.

5.2 Sentiment Analysis

For sentiment analysis task, our main challenge was that there was no labeled data available. With the man power and the time constraints we had we were able to label only 1500 rows of training data. However, it was exciting for us to see how model performances improved with the additional data even if it was only few hundred rows. Also, parameter tuning improved our results for both Machine Learning and Deep Learning models. Few examples would be using linear SVM kernel instead of rgb increase our test and validation accuracy by nearly 10 percent, using tanh activation function instead of relu improved our test accuracy for MLP classifier. All three of us contributed to manually labeling data to ensure there is no bias in the sentiments. Also, in both Datasets a and

Team Memeber	Contribution
Reham	Preprocessing data using word embeddings, Building the deep learning models for part 1, Tuning and optimizing accuracy for part 1 with Pavani.
Sonali	Preprocessing of data, Building and optimizing ML models for part 1, Time series analysis for part 2
Pavani	Preprocessing data using nltk. Data Visualization along with Sonali, CNN experiments and performance tuning for category classification along with Reham, Sentiment analysis- Machine Learning and Deep Learning experiments and performance tuning.

Table 5: Contribution of tasks

b our data was skewed towards positive points and hence we extracted equal samples of data from each category to ensure there dataset is more balanced. We noticed that predictions got better when the data is more balanced.

6 CONTRIBUTIONS

Table 5 details the contributions of each individual team member. Apart from the below listed details, all of our team members put more or less equal efforts in documenting project proposal,mid term report,Final project report,preparing slides for Final project presentation and labeling data for sentiment analysis experiments.

REFERENCES

[1] M.-A. Mittermayer and G. F. Knolmayer, "Newscats: A news categorization and trading system," in *Sixth International Conference on Data Mining (ICDM'06)*, pp. 1002–1007, Ieee, 2006.

[2] J. Staiano and M. Guerini, "Depechemood: a lexicon for emotion analysis from crowd-annotated news," *arXiv preprint arXiv:1405.1605*, 2014.

[3] Y. Rao, J. Lei, L. Wenyin, Q. Li, and M. Chen, "Building emotional dictionary for sentiment analysis of online news," *World Wide Web*, vol. 17, no. 4, pp. 723–742, 2014.

[4] A. Basnet and A. K. Timalsina, "Improving nepali news recommendation using classification based on lstm recurrent neural networks," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pp. 138–142, IEEE, 2018.

[5] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[6] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[7] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

[8] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7, Association for Computational Linguistics, 2002.

[9] D. M. E.-D. M. Hussein, "A survey on sentiment analysis challenges," *Journal of King Saud University-Engineering Sciences*, vol. 30, no. 4, pp. 330–338, 2018.

[10] R. Chiong, Z. Fan, Z. Hu, M. T. Adam, B. Lutz, and D. Neumann, "A sentiment analysis-based machine learning approach for financial market prediction via news disclosures," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 278–279, ACM, 2018.

[11] A. Mukwazvure and K. Supreethi, "A hybrid approach to sentiment analysis of news comments," in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pp. 1–6, IEEE, 2015.

[12] R. urek and P. Sojka, "Software framework for topic modelling with large corpora," p. LREC publication, 2010.

[13] T. Mikolov, K. Chen, G. S. Corrado, and J. A. Dean, "Computing numeric representations of words in a high-dimensional space," May 19 2015. US Patent 9,037,464.

[14] 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)*, pp. 1532–1543, 2014.

[15] B. L. S. B. C. Beel, Joeran; Gipp, "Research-paper recommender systems: a literature survey," in *International Journal on Digital Libraries*, ISSN, 2016.

[16] E. R. D. Mohammad Shahrokh Esfahani, "Effect of separate sampling on classification accuracy," in *Bioinformatics, Volume 30*, p. 242–25, IEEE, 2014.