Automated Newspaper Prediction from News Articles using ML and DL Based Approaches

Ahnaf Mozib Samin, Ayça Avcı, Shantanu Nath
(s4996763), (s4505972), (s4998405)
a.m.samin@student.rug.nl, a.avci@student.rug.nl, s.n.nath@student.rug.nl

Abstract

With the advancement of technology, text classification is now being broadly applied to solve many problems in businesses, government, and research and thus has drawn the attention of the research community. However, it is still considered a challenging task due to the unclear meaning of texts and ambiguity in detecting suitable labels. This paper explores Long Short-Term Memory (LSTM) network and Transformer-based architectures such as Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-Training (GPT) for newspaper name prediction task using a text corpus with 32,226 articles about climate change. We achieve 74.6% accuracy using LSTM with a popular pretrained word embedding model "GloVe" and 85.8% accuracy using RoBERTa, a variant of BERT. In addition, we also emphasize experimenting with the classic machine learning algorithms that yield competitive results in our experiments. We obtain 81.7% accuracy with Support Vector Machine (SVM) algorithm. These results demonstrate the effectiveness of BERT compared to the other algorithms in predicting newspapers from mostly similar types of articles about climate change.

1 Introduction

Text classification utilizes the methods in Natural Language Processing (NLP), text analysis, and Computational Linguistics (CL) for automatically classifying the texts into either binary or multiple classes. This technique has been proven to be very useful in business since it can provide insights into public opinions and classify large numbers of documents using automated systems. The reviews extracted from different websites are used to monitor the current trends, likes, and dislikes which can be used for improvement in the quality of products/services. Nowadays marketing approaches have also been shaped by public opinions. The benefit of text classification is not only confined in business but also has widely been used in different sectors including the Government agencies for decision making and managing documents.

A massive amount of documents can be found online which are not structured well. There have been many attempts to classify news articles based on topics of the articles. In 2009, D. B. Bracewell et al. propose a keyword extraction algorithm for category classification and topic discovery of Japanese and English news articles (Bracewell et al., 2009). P. Chen et al. develop the Recurrent Attention on Memory (RAM) model combining multiple attentions with a Recurrent Neural Network and achieve state-of-the-art accuracy on four data sets, namely SemEval 2014 (two data sets), Twitter data set, and Chinese news comment data set (Chen et al., 2017) in 2017. B. Jang et al. compare the news articles and tweet classification results when training with Convolutional Neural Network (CNN) and word2vec word embedding model in 2019. (Jang et al., 2019). More recently in 2020, large scale news classification using BERT with the pipelines from Spark NLP has been implemented and it has been shown that this approach decreases the training time significantly (Setyo Nugroho et al., 2021). Although there have been many experiments on news article classification based on topics of the articles, little efforts have been made to predict the names of the newspapers from articles, especially when

Classification type	Classes	Number of samples						Total samples (per class)	
		Train		Dev		Test			
		Freq.	(%)	Freq.	(%)	Freq.	(%)	Freq.	(%)
Multi-class	The New York Times	4,000	18.4	828	25.4	848	26.0	5,676	17.4
	The Australian	4,000	18.4	633	19.4	594	18.2	5,227	16.0
	The Washington Post	3,500	16.1	533	16.3	554	17.0	4,587	14.1
	Sydney Morning Herald	3,500	16.1	525	16.1	503	15.4	4,528	13.9
	The Age	3,000	13.7	411	12.6	411	12.6	3,822	11.7
	The Times of India	2,500	11.5	262	8.0	274	8.4	3,036	9.4
	Mail & Guardian	500	2.3	32	1.0	37	1.13	569	1.7
	The Hindu	500	2.3	21	0.6	24	0.73	545	1.7
	The Times	250	1.2	17	0.5	17	0.5	284	0.9

Table 1: Data set divided into different classes for multi-class text classification

topic of the articles in the dataset are quite similar. This is a more challenging task since the inherent writing styles of each of the newspapers have to be captured by the classifier models.

The contribution of this paper is three-fold. At first, we explore LSTM and several transformer-based pretrained architectures including BERT, RoBERTa and GPT etc. to train on our data set. In experimental evaluation, we find that the RoBERTa model yields the best accuracy on our test set, getting 85.8%.

Secondly, we train several machine learning algorithms such as the Naive Bayes, Decision Tree, Random Forest, KNN, Support Vector Machine (SVM), and an ensemble learning approach combining multiple algorithms to compare with the deep learning approaches. We also observe the effect of bag-of-words and TF-IDF input features when training with several machine learning classifiers.

Lastly, we conduct experiments on a climate news based data set that contains news articles from nine different newspapers from four different countries namely United States, Australia, South Africa and India. We measure the cosine similarity of each pair of the nine newspapers and analyze the most unique newspaper based on the result and how it has an impact on the classification result.

In Section 2, information about data set used in the model training and testing is provided. In Section 3, we explain experimental setup and evaluation methods that are used to assess model performance. In Section 4, we provide our results, and in Section 5, we discuss our findings. In Section 6, we make conclusions on the results that are presented in Section 4.

2 Data Set

For newspaper prediction task, we use a data set containing 32,226 articles about climate change published in 9 different newspapers of 4 regions including United States, Australia, South Africa and India, in the time period around the first 25 "Conferences of the Parties" (COP) meetings. The newspapers from the United States are - the New York Times (NY) and the Washington Post (WP). There are three newspapers namely the Australian (AUS), Sydney Morning Herald (SYD), and the Age (AGE) from Australia. From South Africa, the Times (TIMES) and Mail Guardian (MAIL) are chosen. Lastly, the Times of India (INDIA) and the Hindu (HINDU) contributes the articles from India. All the samples in the data set are randomly split into three sets - train, validation, and test sets with a 80:10:10 ratio. The number of articles from each newspaper, however, is not equal and so the data set is imbalanced. For example, the New York Times and the Australian have higher number of articles (8473 and 6120, respectively) whereas the Times and the Hindu have the minimum number of articles, 157 and 255, respectively. Therefore, we apply down-sampling for the classes having higher number of articles and up-sampling for those which contain less number of articles. Down-sampling

and up-sampling have been applied only on our train set since we want to preserve the natural distribution of classes in our dev and test sets. Down-sampling is just the repeating of samples with minority classes whereas Down-sampling is the way of randomly remove some samples from majority classes to make a balance on dataset. The final distribution of our whole data set into train (70%), dev (15%), and test (15%) sets and number of articles per class are shown in Table 1. We have 21750 samples in our train set and 3262 samples in both dev and test sets.

We apply several preprocessing steps on these three sets. First of all, we remove all the names of the newspapers from the articles. We find website addresses of the newspapers online version within the article and so we remove these website links. There are lots of words and letters from other languages which we also excluded from our data set. Then, we remove stop-words and punctuation marks. Moreover, we apply lemmatization in our data set. The stop-words removal and lemmatization are done for all the models except for the transformer based models including BERT. The average length of the articles is reduced after doing the preprocessing. We also add some custom features like number of geo-positional entities, number of organisations, and number of named entity to do experiments with a classic machine learning algorithm. For hyper-parameter tuning, we use our dev set and use our test set only for the final evaluations to keep the fairness of the experiments.

3 Methods

To run the experiments, we use 80% of provided data set as training set. We tune our model parameters using the validation (val) set to select our best-performing models. The main purpose of not including the test set in parameter tuning is to get an unbiased estimate of model performance with the best parameters using the test set. We evaluate our best models' performance on the test set. We perform the multi-class classification task for newspaper prediction using

1. **A baseline classic model using Bag-of-Words and TF-IDF:** Naive Bayes, SVM, KNN, Random Forest, Decision Tree and Ensemble.

2. **A classic model with optimized feature set:** custom features, n-grams.

3. **An optimized LSTM model with pretrained static embeddings:** GloVe

4. **A fine-tuned pretrained language models:** BERT, DistilBERT, RoBERTa, XLNet and GPT.

We experiment using different feature vectors for the classification task performed by our baseline models. Two of the feature vectors are generated:

1. **Bag of Words (Count) Vectorizer:** Converts text data into a vector of token counts, the number of times a word appears in the document.

2. **TF-IDF (Term Frequency - Inverse Document Frequency) Vectorizer:** Converts text into a vector of overall document weightage of a word, weights the word counts by a measure of how often they appear in the document.

We use the Optuna[1] (Akiba et al., 2019) to determine the best parameters for each classification algorithm, and run the experiments using these parameters.

First, we train a Naive Bayes model to perform newspaper classification task. Naive Bayes relies on strong independence assumption between features, and assumes that the presence of a particular feature in a class is not related with the presence of any other feature. It calculates the posterior probabilities for each class for every item in the test set, and assigns each data item to a specific class c^* which has higher probability compared to other classes. Equation 1 represents the Naive Bayes formula.

$$c^* = argmax_{c \in C} \frac{P(X|c)P(c)}{P(X)} \quad (1)$$

Second, we train Decision Tree and Random Forest models. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from training data. The decision rules are generally in form of if-else statements. Root node is from where the decision tree starts. It represents the entire data set, which

[1] https://github.com/optuna/optuna

further gets divided into two or more homogeneous sets. Internal nodes of the tree represent the features of a data set, branches represent the decision rules, and each leaf node represents the outcome. Random Forest is one of the ensemble learning method for the classification tasks that operates by constructing a multiple of decision trees at training time. It fits a number of decision tree classifiers on various sub-samples of the data set, and takes average of the decision trees' results to improve the predictive accuracy and control over-fitting.

Third, we train K-nearest Neighbors (KNN) classifier. It is often referred to as a distant-based algorithm since it calculates the distance of a data point from all points, and then filters out the ones that are closest. To be able to classify the data points correctly, K value needs to be determined. K value is a parameter that refers to the number of nearest neighbours to include in the majority of the voting process. If the K value is too small, overfitting occurs. The model will capture the noise in the data, and will perform poorly for the test set. If the K is too large, underfitting occurs. In this case, the model would be unable to correctly learn on the training data. We decide K as 118 using Optuna since it gives the best accuracy on validation set.

Then, we train Support Vector Machine (SVM) classifier. SVM algorithm relies on creating a line or a hyperplane which separates the data into two classes. We experiment using Support Vector Classifier (SVC) from scikit-learn SVM package. We use different kernels namely "linear" and "Radial Basis Function" (rbf), and different C and gamma values as a parameter to observe the change in the validation set accuracy using Optuna. A kernel is a function that takes low dimensional input space, transforms it to a higher dimensional space. It converts not separable problem to a separable problem. It is especially used for non-linear separation problems. C value controls the trade-off between smooth decision boundary and classifying training points correctly. A large value of C means we will get more training points correctly. Gamma parameter is only used when the kernel is "rbf". It defines how far the influence of a single training example reaches. If the gamma value is high, it means that every point has a close reach. If it is too high, and decision boundary will be dependent on the points that are very close to the hyperplane. This will result in ignoring the points that are very far from the decision boundary since closer points get more weight. Conversely, if the gamma value is low, every point has a far reach, thus far away points get effective weight, decision boundary will be more linear.

We also use custom features that are generated by us, including the count of sentences, count of organizations, and count of the gpe for each text data. We combine them with the other feature vectors to experiment with whether the accuracy changes significantly.

We also experiment with the ensemble model that combines Naive Bayes, Random Forest and SVM classifier. After model training, we test our data with a test set, and print accuracy, precision, recall, and F1-score values of each model mentioned above. We compare these results in Section 5.

After the classic models that are mentioned above, we perform the newspaper classification task using Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) model using GloVe word embeddings. Word embeddings are a learned representation for text where words that have the same meaning have a similar vector representation. LSTM is a special type of artificial recurrent neural network (RNN) architecture. RNNs differ from feedforward neural networks since they have feedback connections that allow information to be passed from one step of the network to the next one. However, RNNs have problems learning long-term dependencies. When the gap between the relevant information and the point where it is needed to becomes very large, RNNs become unable to connect such information. For instance, we have a very long text that we are trying to guess what the last word should be depending on the context of the text. If the last word is relevant to the words that are beginning of the text, RNN most likely will fail to guess the word correctly. LSTMs solve this problem since they are capable of learning long-term dependencies. They use the context of the text/document when making predictions.

They work well on various problems such as speech recognition, handwritten recognition, time series prediction, and grammar learning, and are widely used. Bidirectional LSTM is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. BiLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence).

For this experiment, we use the Keras deep learning API. First, we tune our model parameters. We change learning rate, batch size, epochs to get an LSTM model that does reasonably well in terms of accuracy. Then, we experiment with the LSTM layer in different settings for our model:

1. **Adding extra Bidirectional LSTM layers between Embedding layer and Dense layer.**
2. **Adding Batch Normalization layer:** It solves the problem of interval covariate shift and vanishing gradient problem.
3. **Experiment with dropout on Bidirectional LSTM layers.**

We have some specific tasks like word embedding, language modeling where we can train by using plain text. Therefore, we can use a word-embedding or language model which is trained on a different data set and can be used in other specific tasks like classification, question answering, and other NLP tasks. Pretrained word embeddings like Word2Vec, GloVe, etc. are trained on a huge corpus and know a lot about words where our custom-built word embeddings based on a relatively small data set did not have enough knowledge than pretrained word embeddings. Therefore using pretrained word embeddings is better than training from scratch. Using a huge corpus to learn extra features, and use them on other specific tasks is called Transfer Learning. The idea, transfer the knowledge from one word embeddings to another word embeddings, is also the same for the pretrained models. Here, we are mainly focused on Sequential Transfer Learning from different types of Transfer learning approaches. We used a pretrained Language Model (LM) for multi-class text classification. Because LM is not only a representation of words, but also a representation of sentences, paragraphs, and documents. We mainly focused on Transformers (Vaswani et al., 2017) architecture which only depends on the self-attention mechanism without using any Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) in encoder and decoder. We experienced several Transformers architectures with different auto-encoding pretrained LMs. The Auto-encoding model tries to reconstruct a sentence without using any mask and represents a sentence bidirectionally. One of the auto-encoding models is Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) which uses Masked Language Model (MLM) and predicting next sentences on a corpus. Contextualized Word Vectors (CoVe) (McCann et al., 2018), Embeddings from Language Models (ELMo) (Peters et al., 2018) replace embedding layers and keep the task-specific models whereas BERT replaces the entire embedding layers and model. MobileBERT (Sun et al., 2020) is another Transformers model that also follows BERT architecture with better compatibility with low-resource devices. Then, we experimented with DistilBERT (Sanh et al., 2020), which is a distilled version of BERT base. As the number of parameters is less than the BERT base, it is smaller and runs faster than the BERT base model. Afterward, we experimented with a Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019) that changes some hyperparameters including larger mini-batches and learning rates and removes the next sentence prediction objective. We also experimented with Auto-Aggressive pretrained models like XLNet, GPT to learn bidirectional contexts. The auto-aggressive model is trained in such a way that it can predict the next word in a sentence by using the mask. After getting the best pretrained models, we try different model parameters. First, we changed the length of tokenized sentence by changing the max_length value. Then, we try different optimizers with different learning rates and batch sizes.

4 Results

In this section, we are going to show the outcomes of our experiments. All the results can be repli-

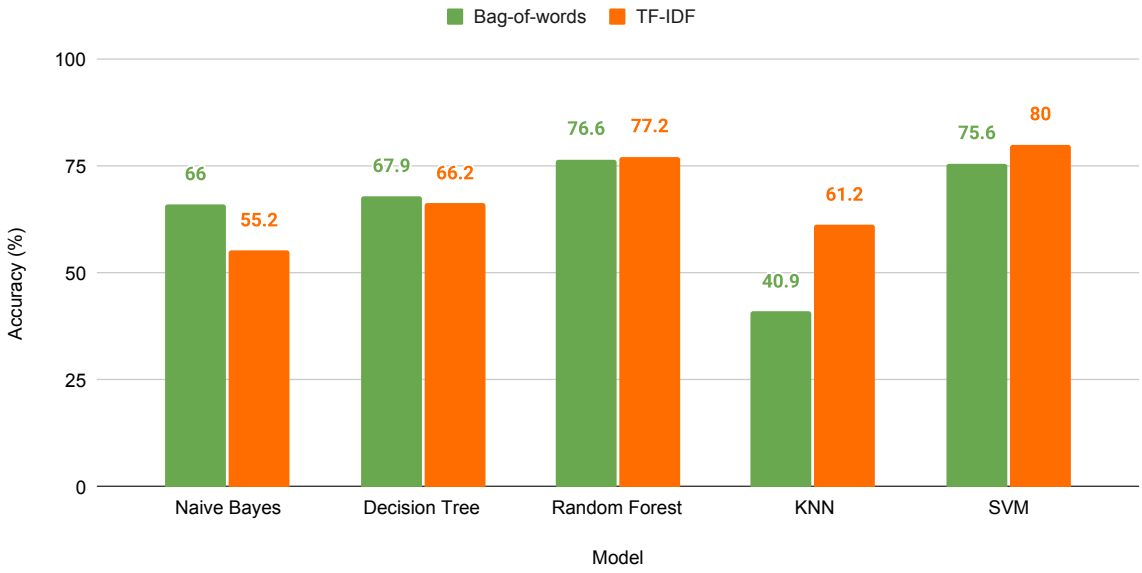

Figure 1: Effect of training machine learning models with either bag-of-words or TF-IDF input feature. Accuracies are reported on the dev set.

Architecture	**Model**	**Tokenizer**	**Accuracy (%)**	
			dev	**test**
BERT	BERT-based-uncased	AutoTokenizer	83.6	84.2
DistilBERT	Distilbert-base-uncased	DistilBertTokenizer	82.4	82.4
RoBERTa	Roberta-base	RobertaTokenizer	85.8	85.8
BERT	Mobilebert-uncased	MobileBertTokenizer	79.2	80.1
XLNet	xlnet-base-cased	XLNetTokenizer	83.6	84.1
GPT	openai-gpt	OpenAIGPTTokenizer	82.1	82.6

Table 2: Several transformer based models' performance comparison on both dev and test sets.

Model	**F1**	**Accuracy (%)**
Naive Bayes	0.44	65.8
Decision Tree	0.59	68.3
Random Forest	0.57	78.1
KNN	0.49	64.0
SVM	0.73	81.7
Naive Bayes+Random Forest+SVM	0.66	80.8
LSTM	0.69	74.6
Bidirectional LSTM	0.69	73.1
BERT	0.85	85.8

Table 3: Comparison between our best-performing classical machine learning models and deep learning models for the newspaper prediction task. Results are reported on test set.

cated by using the LFD-final-project[2] repository on Github.

[2] https://github.com/aycavci/LFD-final-project

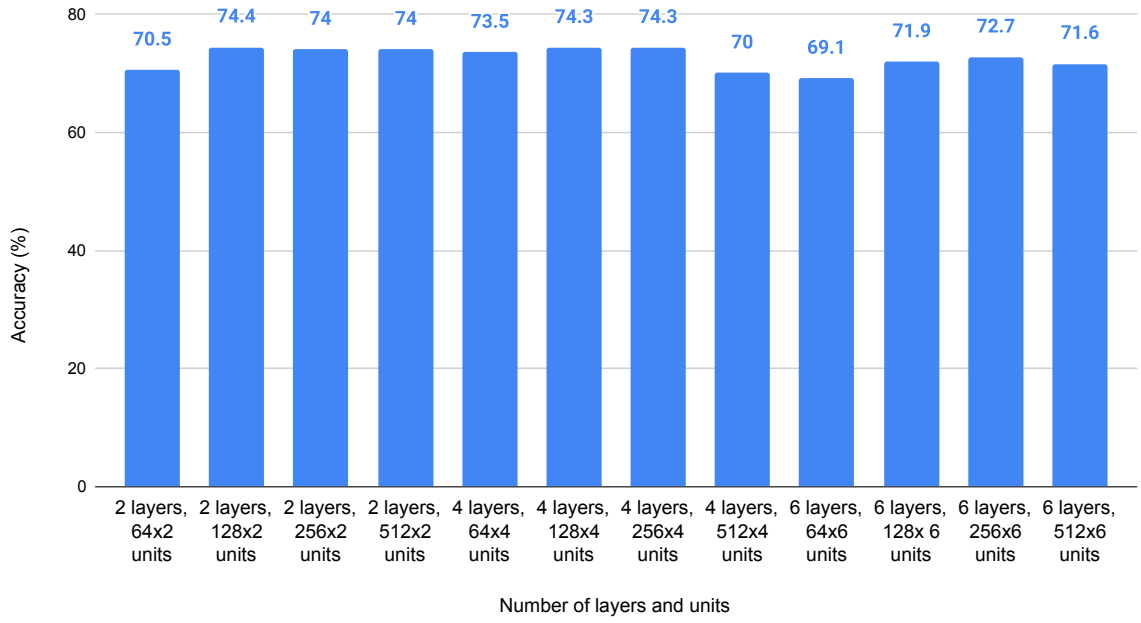

Figure 2: Effect of number of LSTM layers and hidden units on accuracy using dev set. Here, N layers, MxN units denotes that there are N layers and each layer has M units.

Approach	**Accuracy (%)**
Baseline SVM	80.8
SVM + Custom Features	74.2
SVM + bi-gram	76.1
SVM + tri-gram	66.6
SVM + tetra-gram	59.1
SVM + uni-gram + bi-gram	80.6
SVM + uni-gram + bi-gram + tri-gram	79.6

Table 4: Comparison between the baseline SVM and SVM with either custom features or different combinations of N-gram features. Results on dev set are reported.

We use Naive Bayes with bag-of-words input features as our baseline model. In Figure 1, the accuracies are presented calculated using several machine learning algorithms for both bag-of-words and TF-IDF input features on the dev set. Each model has been optimized with the best hyper-parameters found in our experiment. We get 66% accuracy using our baseline Naive Bayes model with bag-of-words features and 55.2% accuracy using TF-IDF features. Using decision tree, 67.9% and 66.2% accuracies have been achieved with bag-of-words and TF-IDF features, respectively. Random Forest performs quite similarly using both of these features, getting 76.6% accuracy with bag-of-words and 77.2% accuracy using TF-IDF. We see noticeable difference in accuracy when training models with KNN. Using bag-of-words, we obtain 40.9% accuracy wheres the accuracy increases to 61.2% with TF-IDF. SVM algorithm yields 75.6% and 80% accuracy using bag-of-words and TF-IDF features, respectively.

Next, we experiment with LSTM using a pre-trained word-embedding model "GloVe". To get a satisfactory result with LSTM, we first set different learning rates and batch sizes. We also find out a baseline architecture with 2 layers and 64 units

Class	Model														
	NB			RF			SVM			LSTM			RoBERTa		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
AUS	0.52	0.80	0.63	1.00	0.24	0.39	0.86	0.84	0.85	0.67	0.76	0.71	0.94	0.90	**0.92**
SYD	0.52	0.52	0.52	0.57	0.72	0.64	0.61	0.68	0.65	0.52	0.60	0.56	0.72	0.75	**0.74**
AGE	0.59	0.26	0.36	0.70	0.34	0.46	0.62	0.60	0.61	0.54	0.38	0.45	0.64	0.67	**0.65**
INDIA	0.91	0.85	0.88	0.98	0.98	0.98	0.97	0.97	0.97	1.00	0.99	**0.99**	0.99	0.97	0.98
HINDU	1.00	0.04	0.08	1.00	0.08	0.15	1.00	0.46	0.63	0.73	0.79	0.76	0.96	1.00	**0.98**
TIMES	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.24	0.38	0.43	0.35	0.39	0.87	0.77	**0.81**
MAIL	0.00	0.00	0.00	1.00	0.24	0.39	0.84	0.57	0.68	0.63	0.59	0.61	0.83	0.78	**0.81**
WP	0.87	0.51	0.64	0.82	0.88	0.85	0.87	0.89	0.88	0.83	0.87	0.85	0.91	0.88	**0.89**
NY	0.73	0.93	0.82	0.87	0.91	0.89	0.92	0.92	0.92	0.91	0.86	0.88	0.93	0.94	**0.93**

Table 5: Precision, recall and F1-score calculated using Naive Bayes (NB), Random Forest (RF), SVM, LSTM and RoBERTa models per class. Here, AUS, SYD, AGE, INDIA, HINDU, TIMES, MAIL, WP, NY refers to the Australian, the Sydney Morning Herald, the Age, the Times of India, the Hindu, the Times, Mail & Guardian, the Washington Post, the New York Times, respectively. Results are reported on the test set

in each layer in addition to a batch normalization layer in between the 2 hidden layers. Based on this architecture, we conduct an experiment changing the number of hidden layers from 2 to 4 and 6, consecutively. For 2, 4, and 6 LSTM layers, we also see the impact of different numbers of units by exploring 64, 128, 256, and 512 units in each layer. The results are reported on our dev set (See Figure 2). Although we cannot observe substantial differences between most of the models, we choose the model with 2 LSTM layers and 128 units per layer as our best-performing model since it yields slightly better accuracy, getting 74.4%.

Table 2 presents the results of our experiments with different transformer based architectures including BERT, RoBERTa (A Robustly Optimized BERT Pretraining Approach), DistillBERT, XLNet and GPT. We try out different architectures of transformers and observe that the RoBERTa-base model with RobertaTokenizer performs the best, getting 85.8% accuracy on both dev and test sets. On the other hand, the Mobilebert-uncased model gets 79.2% and 80.1% accuracy on dev and test sets, respectively, which is the lowest among other models. The rest of the models including BERT-base, DistilBERT, GPT, and XLNet perform quite well, getting more than 82% accuracy on both dev and test sets.

Now, we report the best results performed by several machine learning and deep learning approaches that have been utilized in our experiments with newspaper classification task. As for machine learning algorithms, we exploit Naive Bayes, Decision Tree, Random Forest, KNN, SVM, and an ensemble learning approach combining Naive Bayes, Random Forest, and SVM. In each cases, either bag-of-words or TF-IDF feature has been utilized which is selected on the basis of the performance as described in Figure 1. In addition to that, we also report the results of our best-performing LSTM model and transformer-based RoBERTa model. Table 3 presents the performances of these models on the test set. Each of these models has been evaluated multiple times using 5 different random seeds and then the average value has been taken. Naive Bayes, Decision Tree, Random Forest, KNN and SVM-based models achieve 65.8%, 68.3%, 78.1%, 64% and 81.7% accuracies, respectively on the test set. Our ensemble learning based model combining Naive Bayes, Random Forest, and SVM, however, performs slightly worse than the individual SVM model, getting 80.8% accuracy. In terms of F1-score, we can also see that SVM gets the hightest score, getting 73% while Naive Bayes obtains only 44% score. LSTM and Bidirectional LSTM can not outperform the traditional machine learning algorithms. LSTM scores 74.6% accuracy whereas Bidirectional LSTM performs slightly worse than LSTM, getting 73.1%. At last, our best-performing pretrained

RoBERTa model achieves 85.8% accuracy and an F1-score of 85%, outperforming the rest of the models.

Since, we find that SVM achieves the highest accuracy among other classical machine learning algorithms, we further experiment with this algorithm implementing custom features and N-gram features with different values of N (See Table 4. As seen from the table, adding custom features significantly degrades the accuracy. For N-gram features, bi-gram performs much better than tri-gram and tetra-gram, getting 76.1% accuracy. The combination of uni-gram and bi-gram features reaches close to our baseline SVM model, obtaining 80.6% accuracy.

Finally, we calculate the precision, recall, and F1-score per class using Naive Bayes, Random Forest, SVM, LSTM and RoBERTa on the test set to observe how well different classes are classified (See Table 5). From this Table, we can clearly see that RoBERTa model performs exceedingly well in predicting all classes compared to the rest of the models.

5 Discussion

In Figure 1, we demonstrate that how the accuracy changes for the classsic models namely Naive Bayes, Decision Tree, Random Forest, KNN and SVM using Bag-of-Words and TF-IDF interchangeably. We do an empirical study and find out that for Naive Bayes and Decision Tree classifiers, Bag-of-Words (66% and 67.9% for Naive Bayes and Decision Tree respectively) provides higher accuracy compared to TF-IDF (55.2% and 66.2% for Naive Bayes and Decision Tree respectively). In contrary, for Random Forest, KNN and SVM, Bag-of-Words (76.6%, 40.9%, and 75.6% for Random Forest, KNN and SVM respectively) outperforms TF-IDF (77.2%, 61.2%, and 80.9% for Random Forest, KNN and SVM respectively) giving the higher accuracy.

In Figure 2, the accuracy tends to decrease with layers. We have found that the highest accuracy, 74.4%, is on 2 layers and 128 units per layer whereas the lowest accuracy, 69.1%, is on 6 layers 64 units per layer on dev set. The overall performance of LSTM was not better than SVM because of vanishing gradient problem. SVM train each row at a time whereas LSTM has to learn sequentially to predict the next word on a sequence. Therefore, it encounters vanishing gradient problem for being longer sequence and unbalanced data. We could see a significant difference if we could have a balanced and relatively small article for each newspaper.

Figure 3 demonstrates the cosine similarity measured for each pair of the newspapers. These are measured by randomly taking 100 samples from each newspaper and measuring the cosine similarity between every possible pair of newspapers using a pretrained word2vec model. The above-mentioned experiment has been done twice and then the average value of the two experiments has been taken to strengthen our findings. We can see from the bar graph that the Times of India (INDIA) obtain less than 84% similarity six out of eight times (since there are eight newspapers to compare with). This newspaper might have some distinguishable characteristics that makes it less similar with other newspapers. From Table 5, we observe that the prediction accuracy of the Times of India is higher than the other classes. We argue that since the Times of India has comparatively lower cosine similarity, it becomes easier for our classifiers to predict this newspaper. This experiment also shows that all of the articles get more than 80% similarity which implies that our classification task is challenging.

In Table 3, we can see that SVM did better than any other traditional machine learning algorithm. Extreme generalization power with large feature space capability made SVM is one of the best algorithm for multi-label text classification. Also SVM did better when data size is relatively large with high amount of sparsity on features.(Chauhan et al., 2019) By using *Kernal* trick, it performs a non-linear classification using high dimensional feature space. Also, Ensemble of SVM, DT and Naive-Bayes shows similar performance on our test data.

In table 4, we also try to experiment with different custom features and combination of n-gram features on SVM as it is our best traditional machine learning algorithm. We added Geo positional entity count, Named Entity Count, sentence count

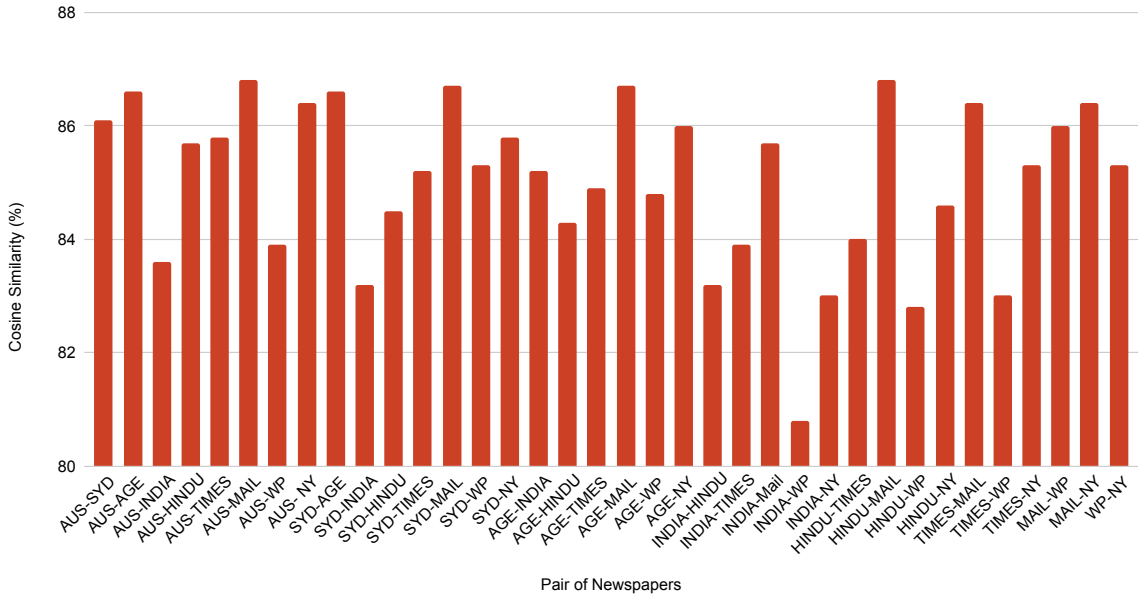

Figure 3: Cosine similarity measured for each pair of newspapers in percentages. Here, AUS, SYD, AGE, INDIA, HINDU, TIMES, MAIL, WP, NY refers to the Australian, the Sydney Morning Herald, the Age, the Times of India, the Hindu, the Times, Mail & Guardian, the Washington Post, the New York Times, respectively.

as our custom features which overall shows a poor performance. Also, we observe that using the combination of n-gram shows better result than using n-gram separately.

In Table 3, we also find that SVM outperforms LSTM and Bi LSTM. Though LSTM reduces the long term dependencies, it still have vanishing gradient problem for relatively large dataset. LSTM predicts future word based on previous words whereas Bi-LSTM predicts current word based on previous and previous words in a context. Therefore, imbalanced class and relatively long sentences of our dataset might be the reason LSTM is showing poor performance. A balanced dataset with more pre-processing steps may show better result.

In Table 2, we compare different BERT models/architectures along with the XLNet and GPT. Results show that RoBERTa outperforms with the 85.8% accuracy on both development (dev) and test set. The noticeable outcome from the Table 3 is pretrained language models performs better than traditional machine learning and sequential learning. As pretrained language model trained on a large corpus, it has better understanding of each words on a context.we used transformers based architecture to train our model. we found that Auto-encoding model shows relativley better performance than auto-aggressive model. Auto-encoding model tries to reconstruct a sentence without using any mask and represents a sentence bidirectionally whereas auto-aggressive model is trained in such a way that it can predict the next word in a sentence by using the mask. One of the auto-encoding models is Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) which uses Masked Language Model (MLM) and predicting next sentences on a corpus. We experimented with a distilled version of BERT base(DistilBERT) (Sanh et al., 2020) and a small version of Bert base MobileBERT (Sun et al., 2020). But we found our best model by using a Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019) model that changes some hyperparameters including larger mini-batches and learning rates and removes the next sentence prediction objective. Additionally, (Liu et al., 2019) claims that BERT is undertrained significantly, and its performance can be exceeded by other language models that are published after it.

In Table 5, results demonstrate that, RoBERTa is overall the best model. It outperforms on each newspaper classes. Naive Bayes and Random Forest performs poorly compared to others, especially on The Times, The Times of India, and, Mail and Guardian. Compared to Naive Bayes and Random Forest classifier, SVM, LSTM and RoBERTa performs good on imbalanced classes (The Times and Mail and Guardian). LSTM did well predicting The Times of India with an exception compared to other models.

6 Conclusion

In this paper, we implement various machine learning and deep learning based models for predicting newspapers from news articles about climate change. This study shows that auto-encoding pretrained model based on RoBERTa architecture outperforms traditional machine learning algorithms as well as auto-aggressive based Transformer models and LSTM. Using RoBERTa, we get 85.8% accuracy, outperming the rest of the models. We also find that SVM, LSTM and RoBERTa performs well while predicting the imbalanced classes, whereas, Naive Bayes and Random Forest get poor accuracy. Moreover, we find out the similarity of each pair of newspapers by calculating the cosine similarity with a pretrained word2vec model. In the future, we want to use political orientations of these newspapers as features to train our classifiers.

7 Contributions

Data preprocessing: Collectively Running experiments: Ahnaf Mozib Samin (machine learning models), Ayça Avcı (LSTM), Shantanu Nath (BERT and friends) Writing: Ahnaf Mozib Samin (Abstract, Introduction, Result), Ayça Avcı (Method, Discussion), Shantanu Nath (Data set, Discussion, Conclusion) Github: Collectively

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

David B Bracewell, Jiajun Yan, Fuji Ren, and Shingo Kuroiwa. 2009. Category classification and topic discovery of japanese and english news articles. *Electronic Notes in Theoretical Computer Science*, 225:51–65.

Vinod Chauhan, Kalpana Dahiya, and Anuj Sharma. 2019. Problem formulations and solvers in linear svm: a review. *Artificial Intelligence Review*, 52:803–855, 08.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 452–461.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80, 12.

Beakcheol Jang, Inhwan Kim, and Jong Wook Kim. 2019. Word2vec convolutional neural networks for classification of news articles and tweets. *PloS one*, 14(8):e0220976.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2018. Learned in translation: Contextualized word vectors.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Kuncahyo Setyo Nugroho, Anantha Yullian Sukmadewa, and Novanto Yudistira. 2021. Large-scale news classification using bert language model: Spark nlp approach. *arXiv e-prints*, pages arXiv–2107.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.