AG News Classification

Sai Akhilesh Ande
Dept. of Mechanical and Industrial Engineering
College of Engineering, Northeastern University
Boston, MA
ande.s@northeastern.edu

***Abstract*—Text classification is one of the fundamental problems in the domain of Natural Language Processing. It has wide applications like spam detection, sentiment analysis, recommendation systems, etc. The goal of this project is to address one such problem called News Classification with classical Machine-learning and Deep-learning based approaches using the dataset of AG_NEWS [1] [2]. This work involves baseline approaches like Multinomial Naive Bayes, Logistic Regession and deep-learning based approaches likes LSTM and Bi-LSTM.**

***Index Terms*—AG News, Naive Bayes, Logistic Regression, LSTM, Bi-Directional LSTM.**

I. Introduction

Text Classification includes a set of problems that involves labeling text sequences with predefined labels. Some common problems in this area are Sentiment Analysis, Text Reviews, Recommendation Systems, etc. News Classification is one such common problem in this area. It involves categorizing news articles into predefined labels called topics. This might seem like a simple classification problem but given the complex semantics of natural languages and the difficulty of extracting semantic features, it is quite often challenging to achieve good classification results. It involves many pre-processing steps to convert unstructured data to structured information.

News information was not readily and easily available on the Internet until the beginning of the last decade. But with the exponential growth of netizens and online news media, the number of news events generating in a unit amount of time has grown very rapidly since then. This resulted in an increase in the need for developing algorithms for news classification as it is a tedious task to do it manually. With such algorithms, users can navigate and access news of their topic of interest in real-time without diving into each random news article. A major problem of news classification is that an article might fall into more than one category or a completely new category. Although humans could do this task more accurately it is almost impossible to manually tag thousands of news articles in real time.

In this project, I have explored various machine-learning and deep-learning-based classification approaches with the goal of developing a model that can best classify a given news article into one of the four predefined news categories - World, Sports, Business and Science & Technology. For this problem, I have used AG_NEWS [1] [2] dataset.

Identify applicable funding agency here. If none, delete this.

II. Background

With the advanced in Deep Learning in parallel with computing machines, approaches like Recurrent Neural Networks, Long Short Term Memory and the latest Transformer based methods have been a huge success. One such Transformer based model called XLNet [4] has so been the best model on this dataset with an accuracy of 95.55%. The next based model is BERT-ITPT-FiT [5] heavily inspired and fine-tuned from the BERT model with an accuracy score of 95.2%.

III. Approach

A. Dataset

The AG_NEWS dataset [1] is a sub-dataset of AG's corpus of news articles constructed by assembling titles and description fields of articles from the 4 largest classes ("World", "Sports", "Business", "Sci/Tech") of AG's Corpus [2]. It contains 30,000 training and 1,900 test samples per class. Each training sample contains a label, news title, and description. For all samples, the title and description are concatenated.

Then the test dataset is further split into validation and testing sets with stratified random sampling to ensure the classes are balanced in all of the training, validation, and test sets.

B. Pre-processing

- In Exploratory Data Analysis(EDA), word-clouds are plotted for each label in the train set to get an overview of the most frequent words. Many HTML tags like ”p”, ”href”, etc. are identified. These words are removed.
- Currency and monetary symbols are converted into respective word tokens. E.g. $ to ”dollar”, % to ”percentage”, etc.
- Contractions like ”won't”, ”I'm”, etc. are expanded to ”will not”, ”I am” etc.
- The common stop-words and irregular words like ”href”, ”p”, etc. which are identified in the word clouds are removed.
- Punctuation, numbers are removed and converted into tokens with ” ” as a delimiter.
- Finally Lemmatization is done using ‘WordNetLemmatizer()‘
- The same set of pre-processing steps are applied to the Validation and Test sets.

C. *Vector Embeddings*

- For the machine-learning based models like Multinomial Naive Bayes and Logistic Regression, vector embeddings of size 2000 are generated using tf-idf model with a vocabulary size of 2000.
- For the deep-learning based models like LSTM and Bi-LSTM, vector embeddings of size 200 are generated with a vocabulary size of 10,000 and 50,000. Each sample is padded with zeros to transform into a fixed embedding size of 200.

D. *Loss Function*

Cross Entropy Loss is used for multi-class classification.

$$L_{crossentropy}(y,p) = -(y\log(p) + (1-y)\log(1-p))$$

E. *Evaluation Metrics*

I choose to use Accuracy as a performance measure because, the classes are balanced in Train, Validation, and Test sets.

$$Accuracy(y,\hat{y}) = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

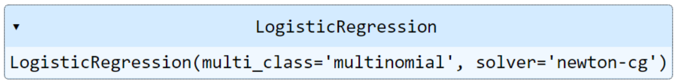
IV. Models

A. *Multinomial Naive Bayes*

As a baseline model, the Multinomial Naive Bayes model with a Laplace smoothing of 1.0 is trained on the Train set with embeddings size 2000.

B. *Logistic Regression*

Then, a Logistic Regression model with L2 regularization and 'newton-cg' solver is trained on the Train set with embeddings of size 2000.



```
▾                     LogisticRegression
LogisticRegression(multi_class='multinomial', solver='newton-cg')
```

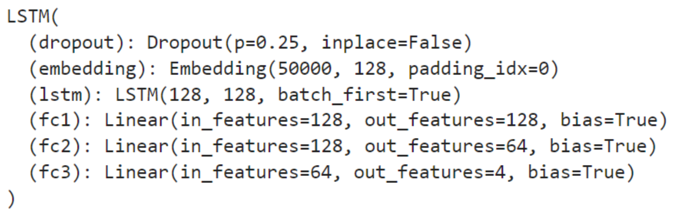
Fig. 1. Logistic Regression

C. *LSTM*

The LSTM architecture consists of one Embedding layer with an input size of 50,000 (vocabulary size) and a hidden dimension of 128. It is followed by an LSTM layer and 3 fully-connected dense layers. The first two fully-connected layers have an activation function of ReLU followed by a drop-out of 0.25.

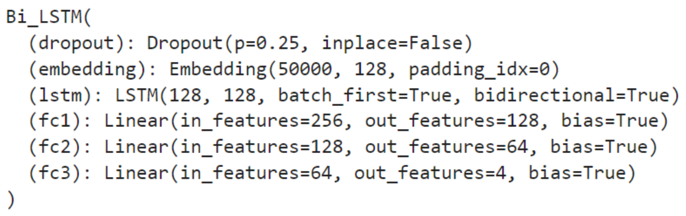
D. *Bi-LSTM*

The Bi-LSTM's architecture consists of one Embedding layer with an input size of 50,000 (vocabulary size) and a hidden dimension of 128. It is followed by two bi-directional LSTM layers and 3 fully-connected dense layers. The first two fully-connected layers have an activation function of ReLU followed by a drop-out of 0.25.



```
LSTM(
  (dropout): Dropout(p=0.25, inplace=False)
  (embedding): Embedding(50000, 128, padding_idx=0)
  (lstm): LSTM(128, 128, batch_first=True)
  (fc1): Linear(in_features=128, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=4, bias=True)
)
```

Fig. 2. LSTM



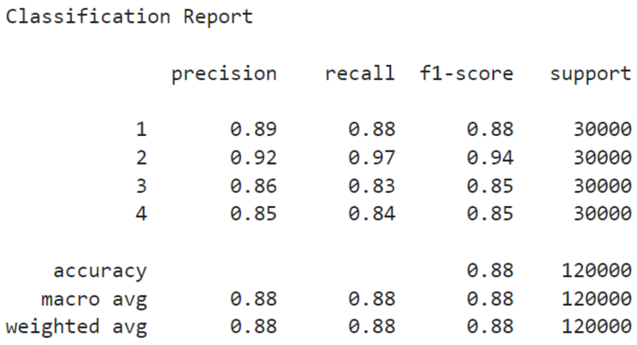
```
Bi_LSTM(
  (dropout): Dropout(p=0.25, inplace=False)
  (embedding): Embedding(50000, 128, padding_idx=0)
  (lstm): LSTM(128, 128, batch_first=True, bidirectional=True)
  (fc1): Linear(in_features=256, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=4, bias=True)
)
```

Fig. 3. Bi-LSTM

V. Results

A. *Mulitnomial Naive Bayes*

The Naive Bayes model gave an average Training Accuracy of 88% and an average Testing Accuracy of 87%.



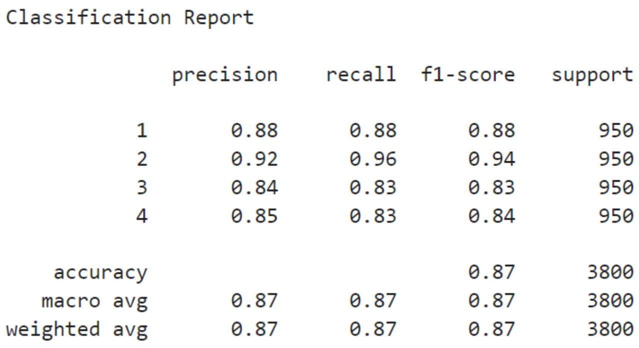
```
Classification Report

              precision    recall  f1-score   support

           1       0.89      0.88      0.88     30000
           2       0.92      0.97      0.94     30000
           3       0.86      0.83      0.85     30000
           4       0.85      0.84      0.85     30000

    accuracy                           0.88    120000
   macro avg       0.88      0.88      0.88    120000
weighted avg       0.88      0.88      0.88    120000
```

Fig. 4. Naive Bayes - Train set



```
Classification Report

              precision    recall  f1-score   support

           1       0.88      0.88      0.88       950
           2       0.92      0.96      0.94       950
           3       0.84      0.83      0.83       950
           4       0.85      0.83      0.84       950

    accuracy                           0.87      3800
   macro avg       0.87      0.87      0.87      3800
weighted avg       0.87      0.87      0.87      3800
```

Fig. 5. Naive Bayes - Test set

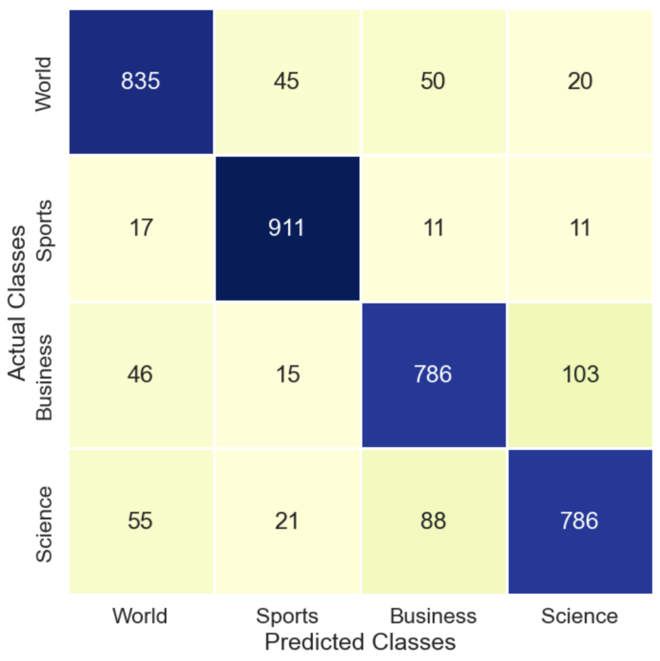
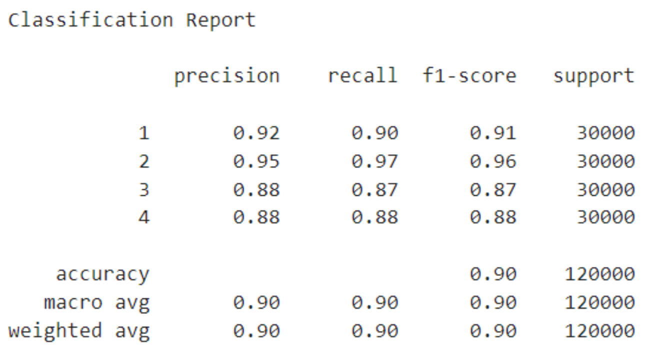


Fig. 6. Naive Bayes - Confusion Matrix

B. Logistic Regression

The Logistic Regression model gave an average Training Accuracy of 90% and an average Testing Accuracy of 89%.



```
Classification Report

              precision    recall  f1-score   support

           1       0.92      0.90      0.91     30000
           2       0.95      0.97      0.96     30000
           3       0.88      0.87      0.87     30000
           4       0.88      0.88      0.88     30000

    accuracy                           0.90    120000
   macro avg       0.90      0.90      0.90    120000
weighted avg       0.90      0.90      0.90    120000
```

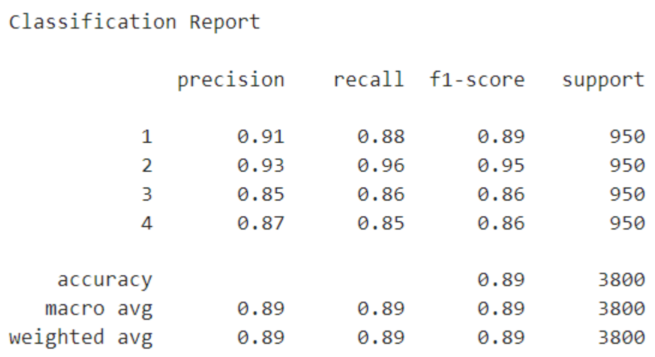
Fig. 7. Logistic Regression - Train set

C. LSTM

Optimizers like Stochastic Gradient Descent, Adam, and RMSProp were used with ReduceLROnPlateau Learning-rate scheduler. The best performance of 91.44% Test Accuracy is obtained with Adam and Nesterov momentum.

D. Bi-LSTM

Adam and RMSProp optimizers were used with ReduceLROnPlateau Learning-rate scheduler. The best performance of 92.34% Test Accuracy is obtained with Adam and Nesterov momentum.



```
Classification Report

              precision    recall  f1-score   support

           1       0.91      0.88      0.89       950
           2       0.93      0.96      0.95       950
           3       0.85      0.86      0.86       950
           4       0.87      0.85      0.86       950

    accuracy                           0.89      3800
   macro avg       0.89      0.89      0.89      3800
weighted avg       0.89      0.89      0.89      3800
```

Fig. 8. Logistic Regression - Test set

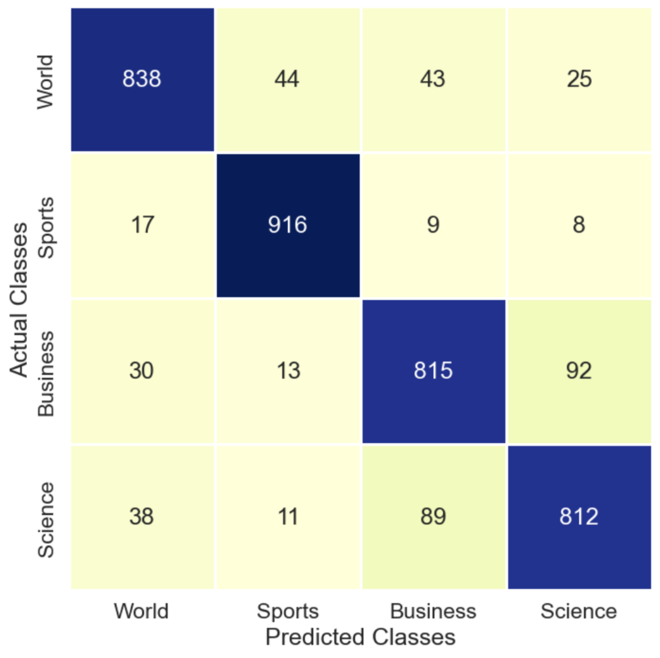


Fig. 9. Logistic Regression - Confusion Matrix

VI. Conclusion

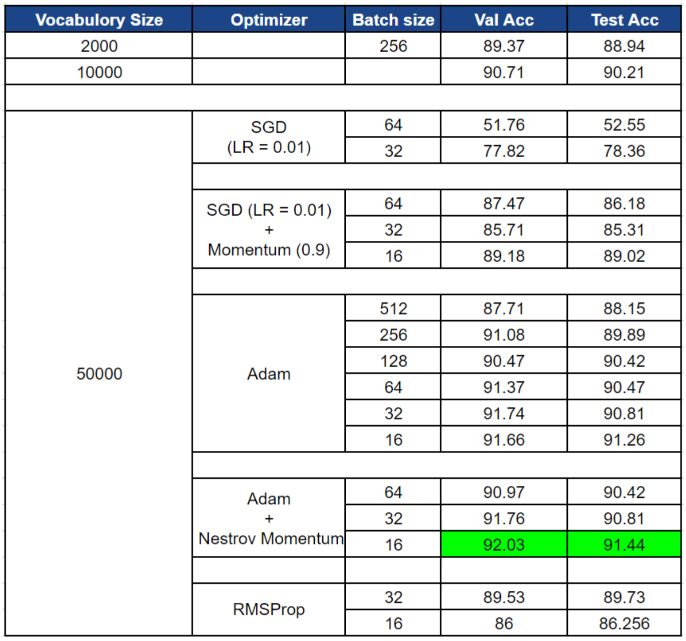
Deep Learning based approaches performed much better than classical machine learning approaches like Multinomial Naive Bayes and Logistic Regression even without the data cleaning pre-processing steps. Adam with Nesterov momentum optimizer gave the best performance of 92.34% Test Accuracy.

The next steps could be

- To use Word-2-Vec, Glove, FastText for embeddings.
- To apply Transfer Learning with the latest transformer-based models like Bert.

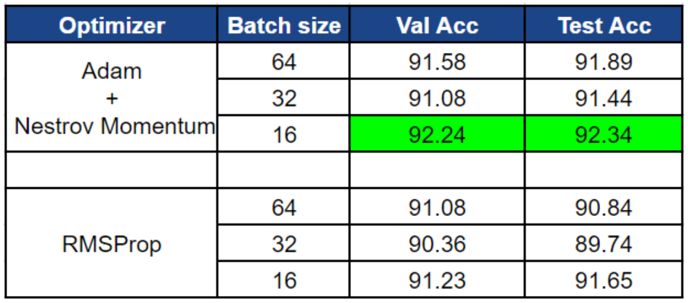
References

[1] https://pytorch.org/text/stable/datasets.html#ag-news
[2] http://groups.di.unipi.it/ gulli/AG_corpus_of_news_articles.html
[3] Character-level Convolutional Networks for Text Classification. Xiang Zhang, Junbo Zhao, Yann LeCun. arXiv:1509.01626. 4 Apr 2016.



Vocabulory Size	Optimizer	Batch size	Val Acc	Test Acc
2000		256	89.37	88.94
10000			90.71	90.21
50000	SGD (LR = 0.01)	64	51.76	52.55
		32	77.82	78.36
	SGD (LR = 0.01) + Momentum (0.9)	64	87.47	86.18
		32	85.71	85.31
		16	89.18	89.02
	Adam	512	87.71	88.15
		256	91.08	89.89
		128	90.47	90.42
		64	91.37	90.47
		32	91.74	90.81
		16	91.66	91.26
	Adam + Nestrov Momentum	64	90.97	90.42
		32	91.76	90.81
		16	92.03	91.44
	RMSProp	32	89.53	89.73
		16	86	86.256

Fig. 10. LSTM results



Optimizer	Batch size	Val Acc	Test Acc
Adam + Nestrov Momentum	64	91.58	91.89
	32	91.08	91.44
	16	92.24	92.34
RMSProp	64	91.08	90.84
	32	90.36	89.74
	16	91.23	91.65

Fig. 11. Bi-LSTM results

[4] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le - XLNet: Generalized Autoregressive Pre-training for Language Understanding 2019

[5] Chi Sun, Xipeng Qiu, Yige Xu, Xuanjing Huang. How to Fine-Tune BERT for Text Classification? 14 May 2019