Machine Learning

Exercise 3: Deep Learning on Text

Emile Johnston 12229987
Fani Sentinella-Jerbić 12206591
Daria Stefan 12229258

July 2023

Contents

1 Approach

1.1 Datasets

In our assignment we focused on two different classification tasks based on two different datasets; Rotten Tomatoes and AG News. Both datasets are highly popular, available through dataset loaders and often used for benchmarking.

Rotten Tomatoes. This dataset is based on movie reviews obtained from the Rotten Tomatoes platform. The task is to classify reviews as displaying positive or negative sentiment, making it a binary classification task. The dataset consists of 10,662 processed sentences in total, equally divided into positive and negative classes, making it balanced. It was first used by Bo Pang and Lillian Lee, in *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.*, Proceedings of the ACL, 2005.
Dataset link: `https://huggingface.co/datasets/rotten_tomatoes`

AG News. AG is a collection of more than a million news articles gathered by an academic news search engine ComeToMyHead from more than 2000 news sources. Xiang Zhang (xiang.zhang@nyu.edu) constructed a news topic classification dataset based on the AG collection. The topic classification was constructed with a multiclass setting with possible classes being "World", "Sports", "Business", "Sci/Tech". It was used as a text classification benchmark in Xiang Zhang, Junbo Zhao, Yann LeCun. *Character-level Convolutional Networks for Text Classification.* Advances in Neural Information Processing Systems 28 (NIPS 2015).

Dataset link: `https://huggingface.co/datasets/ag_news`
With one dataset representing binary and the other representing multiclass classification, as well as the different natures of the tasks being sentiment and topic categorization, we believe we chose diverse enough corpora to gain a good understanding of the performances of different models in the further steps.

1.2 Models

The assignment structure was set up in a way to compare traditional and deep methods so we opted for the following models.

Traditional. For feature extraction, we tried TF-IDF, Bag of Words, and N-grams. On top of each of these we used Decision Tree, Random Forest and XGBoost classifiers.

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical representation used in natural language processing and information retrieval to assess the importance of words in a document relative to a collection of documents. It computes term frequency, which is how frequently a word appears in a specific document, and inverse document frequency, which is the rarity of a word across the entire document collection.

For **Bag-of-words**, the algorithm counts the occurrences of each word in a document and represents it as a vector, where each element corresponds to the word's index and holds the count. This results in a high-dimensional, sparse vector for each document, representing the frequency of words in that document.

N-grams are combinations of N words that follow each other in the document. We used 1-grams and 2-grams.

Deep learning. For the deep learning approach we tried a recurrent neural network (RNN) and BERT architectures.

The **RNN** consists of an embedding layer, to convert discrete data input into continuous vector representations, a recurrent layer with 50 nodes, and a linear layer that produces the output.

For **BERT**, we used a pre-trained uncased base model from HuggingFace and fine-tuned it on our downstream classification tasks.

1.3 Evaluation

Evaluation for this assignment was performed by using a single split of the data, otherwise the deep learning models would take a lot of time and electricity which would be a waste which we don't believe is justified, as inspired by Bender, Emily M., et al. *On the dangers of stochastic parrots: Can language models be too big?* Proceedings of the 2021 ACM conference on fairness, accountability, and transparency. 2021. While we recognize this makes the model comparison less reliable, we consider it good enough for our level of NLP knowledge and experience.

As for the metrics, we used the standard metrics for classification: accuracy, precision, recall, F1-score for each class and aggregated. We also plotted the confusion matrix to inspect the exact errors each model makes. Another thing to consider was the runtime for training* and inference, also taking

into account feature extraction (for traditional methods) or encoding (for deep learning approaches). *Obviously, for the BERT model we didn't account for pretraining time.

2 Results

We present our results in tables, to be able to see the values precisely. We include macro average precision, recall and F1 score for completeness, but it must be noted that since the two data sets we worked on didn't suffer from class imbalance, these values are always very similar to accuracy, and it can be enough to look at the accuracy column to get an idea of performance.

The columns Fitting and Inference indicate the time in seconds required to fit the algorithm to the train set of data, and to evaluate it on test set, respectively. For traditional algorithms, this fitting time also includes the time needed for feature extraction, as well as the time needed for training the model. For neural networks, since they were used on the data without feature extraction, this fitting time corresponds to encoding and training time.

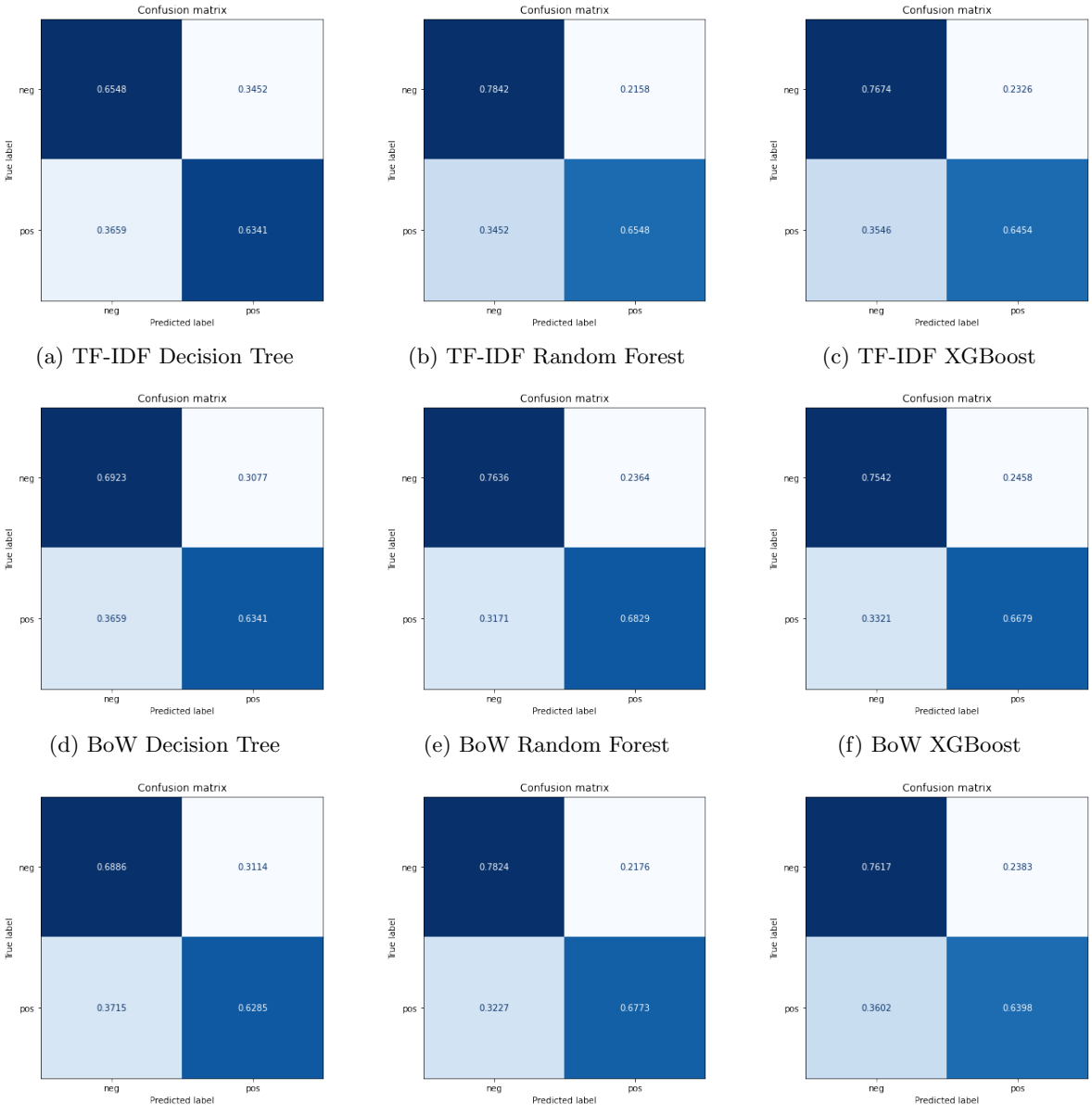
2.1 Rotten Tomatoes

Model	**Precision**	**Recall**	**F1 score**	**Accuracy**	**Fitting[s]**	**Inference[s]**
TF-IDF + Decision Tree	0.6445	0.6445	0.6444	0.6445	2.0427	0.0170
TF-IDF + Random Forest	0.7233	0.7195	0.7183	0.7195	1.4105	0.0588
TF-IDF + XGBoost	0.7095	0.7064	0.7053	0.7064	0.9525	0.0280
BoW + Decision Tree	0.6638	0.6632	0.6629	0.6632	1.6556	**0.0123**
BoW + Random Forest	0.7247	0.7233	0.7228	0.7233	1.4507	0.0673
BoW + XGBoost	0.7127	0.7111	0.7105	0.7111	**0.5444**	0.0233
N-grams + Decision Tree	0.6591	0.6585	0.6582	0.6585	5.6537	0.0306
N-grams + Random Forest	0.7324	0.7298	0.7291	0.7298	5.9113	0.0821
N-grams + XGBoost	0.7038	0.7008	0.6996	0.7008	1.7309	0.0670
RNN	0.5851	0.5844	0.5836	0.5844	11.2089	0.1607
BERT	**0.8434**	**0.8433**	**0.8433**	**0.8433**	969.9965	8.3246

2.2 AG News

Model	**Precision**	**Recall**	**F1 score**	**Accuracy**	**Fitting[s]**	**Inference[s]**
TF-IDF + Decision Tree	0.8114	0.8121	0.8117	0.8121	134.1890	0.2950
TF-IDF + Random Forest	0.8978	0.8982	0.8977	0.8982	243.9085	0.5450
TF-IDF + XGBoost	0.8915	0.8916	0.8914	0.8916	82.6199	0.3570
BoW + Decision Tree	0.8291	0.8296	0.8293	0.8296	117.8931	**0.2433**
BoW + Random Forest	0.8973	0.8976	0.8972	0.8976	274.2800	0.5149
BoW + XGBoost	0.8931	0.8933	0.8931	0.8933	**24.5286**	0.3013
N-grams + Decision Tree	0.8009	0.8013	0.8010	0.8013	856.1183	0.5549
N-grams + Random Forest	0.8984	0.8988	0.8983	0.8988	1253.9895	1.5937
N-grams + XGBoost	0.8883	0.8884	0.8881	0.8884	176.1867	1.1786
RNN	0.8667	0.8667	0.8667	0.8667	250.4943	0.6605
BERT	**0.9480**	**0.9478**	**0.9478**	**0.9480**	12352.1544	64.9745

Figure 1: Rotten Tomatoes Confusion Matrices for Traditional Methods



(a) TF-IDF Decision Tree (b) TF-IDF Random Forest (c) TF-IDF XGBoost

(d) BoW Decision Tree (e) BoW Random Forest (f) BoW XGBoost

(g) N-gram Decision Tree (h) N-gram Random Forest (i) XGBoost

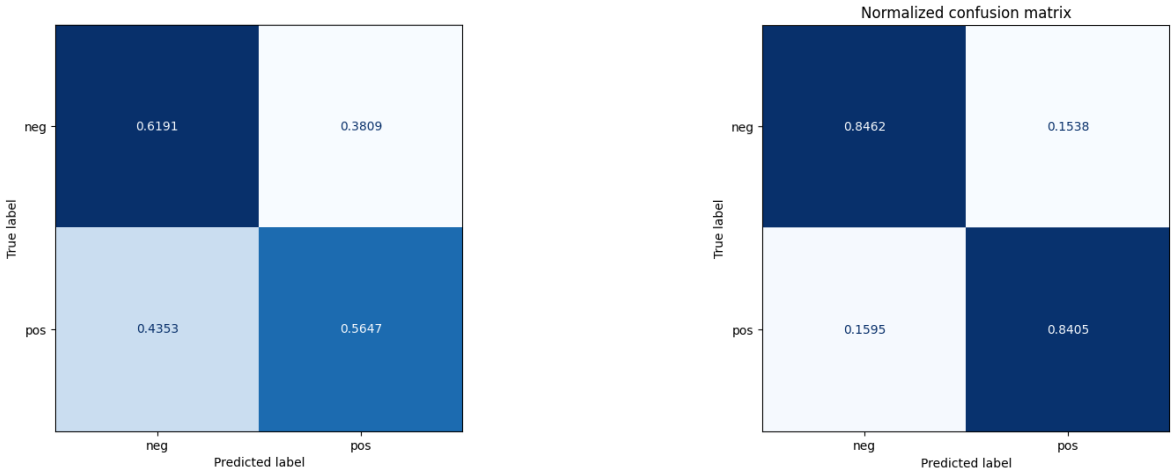
Figure 2: AG News Confusion Matrices for Traditional Methods

(a) TF-IDF Decision Tree (b) TF-IDF Random Forest (c) TF-IDF XGBoost

(d) BoW Decision Tree (e) BoW Random Forest (f) BoW XGBoost

(g) N-gram Decision Tree (h) N-gram Random Forest (i) N-gram XGBoost

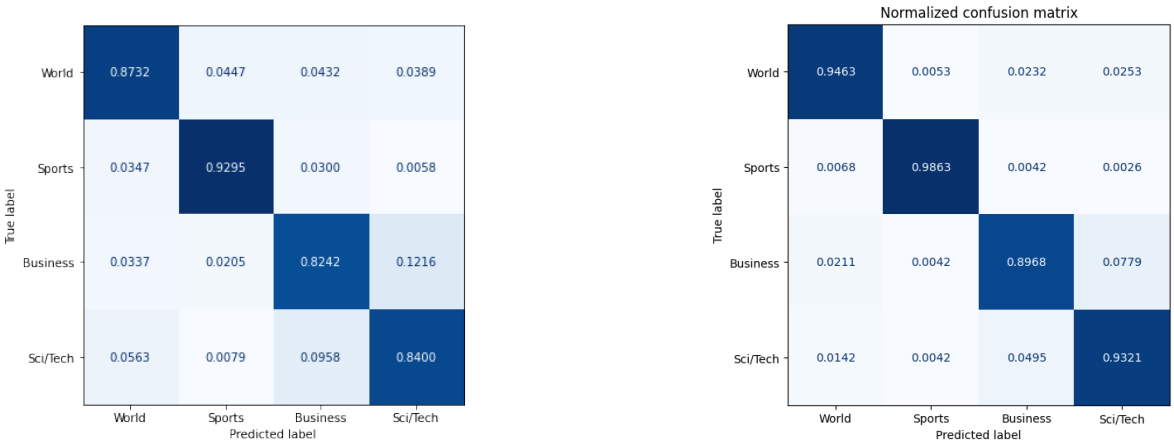
Figure 3: Rotten Tomatoes Confusion Matrices for Deep Learning Methods



(a) RNN

(b) BERT

Figure 4: AG News Confusion Matrices for Deep Learning Methods



(a) RNN

(b) BERT

3 Discussion & Conclusions

There is quite a lot of variation in our results, across models and data sets alike. The RNN performed worse than the more traditional machine learning algorithms for the smaller data set (Rotten Tomatoes), and similarly for the bigger data set (AG News). This is probably due to the nature of recurrent neural networks, which need a lot of data to work well: the Rotten Tomatoes data set may simply not be big enough to allow such a model to show its strengths.

On another note, the BERT model achieved much higher performance than all other algorithms, for both data sets. However, we can consider it natural, as BERT has been pre-trained on English Wikipedia and BookCorpus (a dataset consisting of 11,038 unpublished books). One could say it is actually unfair to even compare it with other algorithms since they are learning only on given data whereas BERT has been trained using masked language modeling and next sentence prediction before it has even been presented with the given training data.

For the rotten tomatoes data set, the traditional algorithms seem to be all biased towards negative. This is also the case for the RNN, however BERT is very well balanced. For the AG News data set, both RNN and BERT have more trouble with some categories than others. From most problematic to best classified, in this order: business, sci/tech, world, sport. This is also almost always the case for the algorithms, which suggests that the differences in accuracy are probably intrinsic to the diversity that exists in the text data, rather than specific to a particular algorithm. For example the Sports category may have more easily distinctive words like 'football' or 'basketball', while Business and Tech may have many key words in common, like 'innovation', 'performance', etc.

Although this may be of lesser importance, because what matters is choosing the best model, we can notice that the difference in performance between the various 'traditional' algorithms is similar to the difference in performance between those and BERT. In other words, the effect on performance of going from a poor 'traditional' model to a better one is similar to going from a better one to neural networks.

In terms of run time, although there is a lot of variation among all algorithms, BERT stands out as needing much longer than all others for both data sets, for both fine-tuning and inference. This can be seen as a trade-off: we can have a fast algorithm or a high-performing one, but not both at the same time. If there are time constraints, especially if we were working on much larger data sets for example, it might be better to go for one of the traditional algorithms even if we know BERT will probably perform better.