Machine Learning Report

Diogo Pereira - 31422012 - dfpp1e19@soton.ac.uk
Alexander Newton - 31449247 - ahrn1e19@soton.ac.uk
Aaron Wing - 28724887 - aw2g19@soton.ac.uk
Michael Lawrence - 31344941- mklg19@soton.ac.uk

I. Introduction

One of the prominent research areas of Natural Language Processing (NLP) is the ability to classify text. Successful examples of text classification include topic labelling, spam detection, and intent detection. Sentiment analysis is another example of a developing research area within text classification that labels text with a sentiment score. This project uses the Amazon Book reviews dataset to explore how to receive the most performant models to accurately identify the sentiment of book reviews.

Firstly, section II covers the feature extraction methods for text data. Section III shows how the problem of dataset imbalance was managed and sections IV and V describe how the machine and deep learning techniques are implemented. Section VI and VII give the results for the 3 class and 5 class problem, respectively, followed by section VIII which evaluates our final results. Finally, section IX concludes the report with explanation of future work.

II. Features

As previously explained in the data exploration report, the Amazon book review dataset [1] was chosen to train models with the aim to predict the start rating of the review. After preprocessing the dataset, the review samples needed to be transformed into a set of features that could be used as an input for machine learning models.

A. *Bag-of-words Models*

For classical machine learning, features can be created using a bag-of-words (BOW) model. The BOW model represents a corpus of text as an unordered set of words ignoring their position relative to each other and only counting their frequency. We can then define a feature as the frequency that a word appears.

The traditional BOW model can be improved by applying a weighting to each of the tokens. This is known as Term Frequency-Inverse Document Frequency (TF-IDF) which considers words of high frequency to have less importance. The weight of each token is calculated by $w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$ where $tf_{i,j}$ is the frequency of term i in document j, df_i is the frequency of i in all the documents and N is the total number of overall documents. This has the effect of increasing token importance if it appears frequently in a single document, but the final importance is also decreased if it appears frequently among all documents.

B. *Vector Embeddings*

The disadvantage of BOW models is that they lack the ability to represent the semantics of a words. For example, they do not take into account word similarities or contextual position. The current best approach is to use vector semantics where a word is represented as a vector in some high dimensional space. Words projected in this higher dimensional space are known as vector embeddings. Subsequently words that are similar to each other appear close together in the vector space and linear operations can be applied to the vectors to produce words with a combined meaning. Vector embeddings are the standard way of giving input to deep learning models however they can also be used in place of BOW models for classical machine learning. In the data exploration report it was discovered that the GloVe [2] and FastText [3] embeddings give a high coverage over the dataset vocabulary and subsequently are used to convert review tokens to their dense vector representation.

III. Managing Dataset Imbalance

In the data exploration report it was highlighted that the dataset was heavily skewed. 59% of the reviews have a 5 star rating with only 2% of the reviews being rated 1 star. The imbalanced dataset was run through a Naïve Bayes model to give an idea of the effect that class imbalance has on classification. The accuracy score of the model was 0.60, however, the classification report showed that the f1-score for classes with star rating 1-4 was zero or very to zero. On the other hand, the f1-score for the 5 star review class was 0.75. This large discrepancy between the f1-scores indicate that the accuracy score was mostly dependent on the correct classification of the 5 star reviews. One could essentially create a toy model that only predicted 5 star for each review and the model would obtain 59% accuracy. Therefore the imbalanced dataset would not give valid indication of the performance of a model and hence re-sampling strategies were explored.

A. *Resampling*

Resampling is a method of correcting imbalanced datasets by either removing samples from the majority classes (under-sampling) or adding more samples to the majority classes (over-sampling).

To initially balance the dataset, the classes were down-sampled to the size of the smallest class. In this case the smallest class was the one star reviews with 20,000 samples. Therefore a subset of the dataset was generated with 100,000 reviews with each class having 20,000 reviews. Naïve Bayes trained on the down-sampled dataset and achieved a poor accuracy of 0.207 with an f1-score of 0.208 with the small number of samples per class contributing to the low classification performance.

To increase the number of reviews in the dataset while keeping the classes balanced, under-sampling was combined with over-sampling. Over-sampling using the Synthetic Minority Oversampling Technique (SMOTE) algorithm [4] was applied to the dataset to generate more synthetic samples for the underrepresented classes. The algorithm works by picking random points from the minority class and adding the the k-nearest neighbors points to the dataset. Using the imbalanced-learn python library [5], SMOTE was applied to the TF-IDF encoded samples of minority classes. Due to the size of the dataset and the high dimensionality of the TF-IDF vectors, SMOTE took a large amount of processing power and long time in order to generate new synthetic samples. Unfortunately the added samples gave minimal improvement to the accuracy of models. It may be the case that the SMOTE algorithm is not complex enough to create new synthetic natural language sentences that give any benefit to a classifier model.

B. Obtaining more data

The optimal solution to solving the class imbalances is to obtain more data for the smaller classes. It was discovered that the creator of the Amazon Book review dataset had recently revised and updated the dataset [6]. The new dataset includes 27,164,983 book reviews with amazon review data spanning from May 1996 - Oct 2018. In the new dataset, the one star review class was the smallest with 827,009 reviews. Therefore the dataset was down-sampled to have 100,000 reviews in each class with 500,000 reviews in total. Even though the new dataset is smaller than the original dataset, the classes now have an equal number of reviews. Applying the baseline Naïve Bayes to the new dataset gave an accuracy score 0.506 which is an significant improvement over the previous dataset. For the rest of the report the new dataset will be used to train the models.

IV. Machine Learning Models

A range of supervised ML models from the Scikit-learn [7] python library were trained to see which models that gave the best classification score. These models include: Support Vector Machines (SVMs), Decision Tree, Random Forest, Logistic Regression, Naïve Bayes and Adaboost.

A. Methodology for Classical Machine Learning Models

For training the baseline ML models, the reviews were first pre-processed and then transformed into a sparse vector format through a TF-IDF vectoriser. The subsequent vectors are shuffled and split into training and testing with a 80:20 split. These models were initially tested with default hyper-parameters in order to give baseline scores.

To further fine-tune the models, hyper-parameter search was conducted via randomised grid search with 3 fold cross validation. Due to the large size of the dataset it can take models several minutes to train and 3 fold cross validation requires 3 models to be trained per set of chosen parameters. Therefore an exhaustive grid search could take hours or even days to run depending on the granularity of the parameters space to be searched. Randomised grid search overcomes this problem by sampling a fixed set of parameters values for a model from a distribution over possible values instead of searching the entire parameter space. By performing cross-validation, the models' parameters can be tuned over several validation sets to give scores of how well the model could perform on new data.

V. Deep Learning Models

Two different approaches were used for the deep learning models: fine-tuning pre-trained models and training neural networks from scratch. For fine-tuning, three state-of-the-art deep neural networks were chosen: Google's BERT [8], Carnegie Mellon University and Google's XLNET [9] and Facebook's RoBERTa [10]. All these models are based on the Transformers architecture [11] which is comprised on an encoder-decoder architecture with the addition of attention. Attention decides which parts of the input sequence are important allowing the model to focus only on the relevant parts of the sequence. What also differentiates these models is that they are **Autoencoders (AE) language models**, which are defined by trying to reconstruct corrupted input data by replacing random words from the input sequence into a **[MASK]** and aiming to predict the original word. The advantage of such models is that they can see the context of a sentence in the forward and backward direction.

A. Methodology for Deep Learning Models

For fine-tuning, all the reviews text needs to be converted into a vector format that can be processed by neural networks. Each of the reviews are parsed by a tokeniser which splits text into tokens and adds a classification token to the beginning of each sentence. The tokeniser assigns each token with a corresponding ID (token embeddings) and then proceeds to pads or truncate sentences to the same length. The maximum sentence size was set to 256 tokens (words) given that the majority of reviews do not exceed this size limit. Finally the tokeniser adds attention masks to avoid performing attention on the padding tokens.

For BERT, the base architecture is used with 12 layers, hidden size of 768, and 12 attention heads trained on lower case English text. The network was fine-tuned with a starting learning rate of $2 * 10^{-5}$, which linearly decreased each step. RoBERTa is a variation of BERT that uses an improved training procedure and is pre-trained on a larger corpus of data. Due to this the starting learning rate was set to be $5 * 10^{-5}$ because RoBERTa's weights are already highly optimised. XLNET also has an architecture similar to BERT, however, it handles **[MASK]** tokens in a different way. Fine tuning involved using a starting learning rate of $5 * 10^{-5}$ with linear decrease per step. All the models were ran through 2 epochs on a RTX 2070 with a batch size of 8.

Recurrent Neural Networks (RNNs) using LSTM (Long Short Term Memory) units were trained from scratch. RNNs have loops that allow information from previous steps to persist in the network, while the LSTM units avoid long-term dependency problems. LSTM units are composed by gates (input, output and forget) which pass information through a sigmoid activation functions to control how much of the information should be let through. The cell carries the information, that can be updated when passing through the gates. At the beginning of the RNN there is an embedding layer (using GloVe embeddings), followed by LSTM layers. The output of the LSTM layer is passed to a fully connected layer containing 5 or 3 output units (depending on the class problem in hands) with a softmax activation function. After some hyper-parameter tuning, the embedding dimension was set to 100 and the number of LSTM layers was set to 3 with 100 hidden units each. This was trained for 10 epochs, with a batch size of 64 and starting learning rate set to 10^{-3} decreasing by a factor of 0.1 every 3 epochs.

VI. 5-class problem Results

This section compares the results of machine learning and deep learning models for predicting the 1 to 5 star ratings. A different range of metrics are used: accuracy, macro f1-score, and Matthews Correlation Coefficient (MCC). Accuracy only takes into account the number of correct predictions so it only focuses on the true positives and true negatives whereas f1-score is concerned with the balance of the model focusing mainly in the false positives and false negatives. The macro f1 score was used as it averages the f1-score of each class instead of the micro f1 score as the classes are balanced. Finally, MCC focuses on the correlation between the true and predicted values and a good balance between all of these three measures is what classifies a good classifier. The data was split training, testing, validation with a 80:10:10 ratio.

A. Classical Machine Learning Results on the 5-class problem

Model	Test Accuracy	Macro F1 Score
Naïve Bayes	0.505	0.504
Random Forest	0.493	0.482
Decision Tree	0.379	0.378
AdaBoost	0.421	0.404
Logistic Regression	0.533	0.530
SVM	0.482	0.481

TABLE I: Machine Learning Results for the 5 class problem

Class	precision	recall	f1-score	support
1	0.590	0.640	0.614	33092
2	0.463	0.404	0.432	32996
3	0.463	0.425	0.443	32736
4	0.478	0.538	0.506	32965
5	0.680	0.680	0.680	33002
Accuracy				0.538

TABLE II: Classification Report for the 5 class Logistic Regression

Table I shows the results for the accuracy and F1 scores for each of the models. The best models include Logistic Regression and Naïve Bayes. Interestingly the worst performing models (AdaBoost and Decision Trees) both make use of decision trees for classification. Overall the general performance of all the models is very low with the best models only classifying half the reviews correctly. Table II shows the classification report for Logistic Regression. The model achieves an overall accuracy of 0.538 however the f1 scores show that the model manages to classify 1 and 5 star reviews with the most accuracy. Classes 2, 3 and 4 have f1 scores less then 0.5 indicating that the models are struggling to predict these star rating.

B. Deep Learning Results on the 5-class problem

Model	Train Accuracy	Test Accuracy	Macro F1-Score	MCC
BERT	0.604	0.58	0.57	0.470
RoBERTa	0.674	0.640	0.640	0.548
XLNET	0.643	0.610	0.610	0.516
RNN	0.625	0.580	0.580	0.476

TABLE III: Deep Learning Results for the 5 class problem

As it can be seen in Table III all the deep learning models performed better than the machine learning models. The training accuracy was always slightly higher than the test accuracy as expected given that the model was optimized with the training data. However, the model under-fitted given that both accuracy scores are low. The best models were RoBERTa and XLNET given that they are the most recently released with updated architectures. Interestingly, even though the RNN was much shallower than the other models, it achieved a good performance when compared to the other. Similarly to the previous subsection, the deep learning models classified the 1 star and 5 star reviews with the highest classification accuracy as indicated by the f1-scores. Subsequently the MCC score is quite low as not all classes are predicted well.

VII. 3-class problem Results

Based on the scores for the 5 class classification problem, it was decided to decrease the complexity of the classification problem by reducing the number of classes to predict from five to three. The data exploration report which showed that one/two star reviews and three/five star reviews shared common bi-grams and therefore makes it difficult to classify the reviews with ratings of 2, 3 and 4. Therefore the classification problem was pivoted to predict sentiment of positive, neutral and negative by removing 2 star and 4 star reviews. As the dataset had equal reviews in each class, this reduced the dataset size to 300,000 reviews.

A. Classical Machine Learning Results on 3-class problem

Table IV shows results for machine learning models trained on the three class dataset with the baseline columns showing the scores of the models before hyper-parameter tuning. Overall, the models achieve a significantly higher accuracy then the five class dataset with the Decision Tree achieving the lowest score of 0.648 and the Logistic regression achieving the highest score again with an accuracy of 0.796.

Hyper-parameter tuning was then applied to each of the models as described in section IV-A in order to find parameters that give a close to optimal classification score for the dataset. Each model was ran again with 3 fold cross validation with the best hyper-parameters to obtain a final score. The mean accuracy and F1 scores are reported in Table IV with the corresponding standard deviations.

Model	Baseline Acc	Baseline F1	Accuracy	F1 Score
Naïve Bayes	0.735	0.734	0.783 $\pm$ 0.00299	0.784 $\pm$ 0.00289
RF	0.716	0.708	0.730 $\pm$ 0.00395	0.720 $\pm$ 0.00364
DT	0.629	0.632	0.648 $\pm$ 0.00105	0.648 $\pm$ 0.00040
AdaBoost	0.665	0.661	0.691 $\pm$ 0.00383	0.689 $\pm$ 0.00326
LR	0.796	0.795	0.800 $\pm$ 0.00465	0.799 $\pm$ 0.00463
SVM	0.747	0.749	0.7581 $\pm$ 0.00234	0.7594 $\pm$ 0.00181

TABLE IV: Machine Learning Results for the 3 class problem

The final results show that hyper-parameter tuning added a few percentage points to the accuracy scores of all the models. Similarly to the baseline models, logistic regression gave the best scores with 0.8 and 0.790 for accuracy and F1 score respectively. The variation of the scores for each of the ML models is very small with scores only varying very slightly over different validation sets. This may be due to the equal class sizes which prevents any bias to a specific class.

B. Deep Learning Results on the 3-class problem

Again for the 3-class problem, the deep learning models managed to improve on the classical machine learning. Once again, the train accuracy was slightly higher than the test accuracy but this time the models didn't under-fit. RoBERTa achieved the best accuracy of 87% and there was a bigger balance between the f1-scores of each class for every model, which is also reflected in a better MCC score.

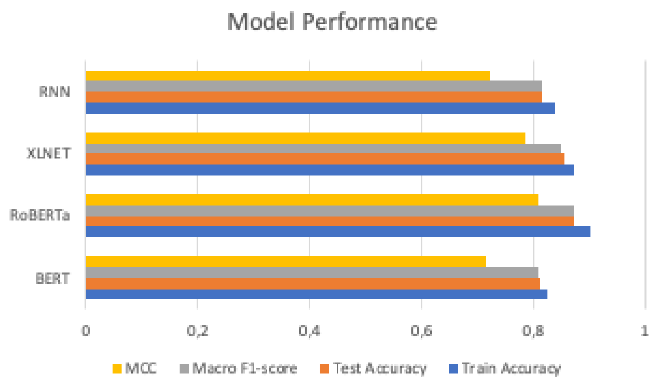

Fig. 1: Performance of Deep Learning models on 3 class problem

VIII. Discussion

There is a clear difference seen in the results between the 5 and 3 class datasets. The significantly lower score for 5 class indicates that differentiating between five different classes is a complex task given that even for a human it would be difficult to distinguish between some reviews. Even some of the deep learning models achieve a modest 64% accuracy. On the other hand, the 3-class task is more plausible where most models had a good performance.

All the deep learning models outperformed the machine learning ones showing that deep neural networks are better at modeling the underlying sentiment of a review. Such can be explained by the fact the deep learning models used automatic feature extraction and can have advanced network architectures using transformers and attention mechanisms. Evidently this has proven useful in building more accurate features, however, this all comes at a cost of computation time and resources. Fine tuning one of the deep models took over 14 hours on average with a RTX 2070 while an equivalent machine learning model (such as Logistic Regression) only takes around 5 minutes. It is even possible that the deep models could achieve higher performances when trained with more review samples. If trained on a dataset with more then 300,000 reviews, the accuracy could have surpassed 90%, however, more time and resources would be needed to fine tune the hyper-parameters and weights which wasn't feasible with our limited resources.

The best scores for the machine learning models rival some of the deep neural networks. Logistic regression for example scores 80% on the 3 class problem where BERT scores only marginally higher accuracy of 81.5%. This is a case of Occam's razor when given 2 models with the same outcomes, one should choose the simpler model. In this case, significantly simpler models can perform equivalently to very complex models.

IX. Conclusion

In this project, our goal was to try to apply sentiment analysis to Kindle reviews. We started by applying models to a 5 class problem, however the results fell short of expectations given that the more neutral reviews were indistinguishable from each other. This led us to switch to a 3 class problem where all models achieved a significantly higher classification score, with the deep learning models performing better than the machine learning models at the cost of computational power and resources. Future work could include exploring the research area of text generation where given a star rating and a product name, the models would produce a fake review. Such a project could be accomplished using fine-tuning once again on more complex models like GPT-2 [12] or by implementing models ranging from simple Markov models to more complex Conditional Variational Autoencoders.

References

[1] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with One-Class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, (Republic and Canton of Geneva, CHE), pp. 507–517, International World Wide Web Conferences Steering Committee, Apr. 2016.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. http://www.aclweb.org/anthology/D14-1162.

[3] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[4] A. Dal Pozzolo, O. Caelen, and G. Bontempi, "Comparison of Balancing Techniques for Unbalanced Datasets," *Machine Learning Group Universite Libre de Bruxelles Belgium*, vol. 16, no. 1, pp. 732–735, 2010.

[5] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[6] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 188–197, Association for Computational Linguistics, Nov. 2019. https://www.aclweb.org/anthology/D19-1018.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. http://arxiv.org/abs/1810.04805.

[9] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. http://arxiv.org/abs/1906.08237.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. http://arxiv.org/abs/1907.11692.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. http://arxiv.org/abs/1706.03762.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.