COMP 551 - Miniproject #3 Report

Classification of Textual Data

Maxim Boucher — 260902501
Allison Mazurek — 260895254
Youssef Samaan — 261032708

***Abstract*—The aim of this project is to perform sentiment analysis on text data taken from the IMDB Movie Review Dataset [5]. This report compares the performance of Naive Bayes, implemented from scratch, BERT (Bidirectional Encoder Representations from Transformers) [3] with pertained weights using SimpleTransformers and TensorFlow, as well as other classification and transformer models including Logistic Regression, SVM, NBSVM, XLNet [8] and ALBERT (A Lite BERT) [4] models, all imported from packages and libraries. The results show that Naive Bayes performed well with a test accuracy of 84.012% while training much faster than the transformer models. All of the non-transformer models were trained and evaluated much faster in comparison to the transformer models. The Logistic Regression, Support Vector Machine (SVM), and Naive Bayesian Support Vector Machine (NBSVM) models received test accuracies of 86.864%, 84.836%, and 89.083%, respectively. The transformer models BERT, ALBERT (A Lite BERT), and XLNet received test accuracies of 90.386%, 90.472%, and 91.844%, respectively. BERT performed much better than Naive Bayes. However, it took considerably longer to compute compared to the non-transformer models. ALBERT and XLNet were also much slower than the non-transformer models, but they both ran a little faster than BERT. XLNet's training time was not as high as BERT and ALBERT but was still markedly slower than the non-transformer models. Finally, we tested an MLP model to compare feed-forward neural networks to the bi-directional transformers, which got an accuracy of 49.540%.**

I. Introduction

This project focuses on comparing traditional text analysis methods to classify reviews from the IMDB Movie Review Dataset as having either positive or negative sentiment. The main focus of this report is to study the performance of our simple Naive Bayes model implemented from scratch with BERT, a more sophisticated transformer model implemented using the SimpleTransformers library. We have also decided to compare their performance against other models of their ilk, and as such we have tested sentiment analysis on two other transformer models that succeeded BERT: ALBERT (A Lite BERT) and XLNet. While ALBERT was selected to compare its effectiveness against its parent model, XLNet was chosen to measure how much more accurate later transformer models have gotten. Similarly, in order to draw comparison to our Naive Bayesian model we have tested other simpler models, including Linear Regression, Support Vector Machines (SVM), and Naive Bayesian Support Vector Machines (NBSVM) on the same dataset. This dataset consists of *25,000 highly polar movie reviews for training, and 25,000 for testing* [5]. The supervised IMDB-dataset was imported using TensorFlow due to its large size and subsequently loaded into different data structures as required for each model (NumPy arrays and Pandas Dataframes).

Naive Bayes performed considerably well and in a very short time, resulting in a test accuracy of 84.012%. The Logistic Regression, Support Vector Machine (SVM), and Naive Bayesian Support Vector Machine (NBSVM) models were trained in roughly the same amount of time and received test accuracies of 86.864%, 84.836%, and 89.083%, respectively. Meanwhile, the transformer models - BERT, ALBERT, and XLNet - received test accuracies of 90.368%, 90.472%, and 91.844%, respectively. The test and train accuracy of each model is listed in Table II. Lastly, we tested an MLP model made from scratch (adapted from MP2 code) to compare feed-forward neural networks to the bi-directional transformers, which returned a test accuracy of 49.540%. All models were trained on data that was pre-processed the same way.

II. Related Work

NBSVM is a variation of both Naive Bayes and Support Vector Machines designed and described in a paper by Wang and Manning [7]. NBSVM addresses their frustrations with other researchers' failure to consider the high variance in performance dependent on the features and datasets present in NB and SVM models. NBSVM skillfully solves these issues by interpolating both models.

BERT [3] is the first deeply bidirectional, unsupervised language representation [2] model. However,

it contains an immense amount of parameters (110M and 340M for BERT-base and BERT-large, respectively) and is thus very computationally expensive. BERT has been expanded upon to reduce the number of parameters without reducing the resulting accuracy, as well as incorporating bi-directionality. While training, BERT applies masks on roughly 15% of each data sequence and predicts on these masked tokens. This means that while BERT does quite well in practice and was cutting-edge for its time, this masking causes an independence assumption that holds it back from other, more advanced transformer-based models like XLNet.

ALBERT [4], submitted to the ICLR 2020 conference and released by the creators of BERT a year after the latter, incorporated parameter reduction techniques, including factorized embedding parameterization to increase the hidden size of the model without increasing its parameter count, and cross-layer parameter sharing to prevent a substantial increase in the number of parameters as the depth of the network grows. ALBERT-base only involves 12M parameters, reducing the number of BERT parameters by 70% while maintaining BERT's effectiveness.

XLNet [8] is a generalized auto-regressive model where the next token is dependent on all previous tokens. In other words, like BERT, XLNet is a bidirectional unsupervised language model, but it uses *Permutation Language Modeling* in its pre-training, which allows it to use subsets of a word's neighbors in its predictions so that the prediction is not seen only through the words used to predict it. It also does not change the order of the words while doing so; if it did, this technique would simply degenerate into a bag-of-words model. By contrast, while BERT also captures bi-directional context, as previously mentioned its training applies masks on roughly 15% of each data sequence and as such it makes a significant independence assumption. XLNet does not make such an assumption, giving it an advantage over BERT and its variants.

III. Dataset Description and Preprocessing

The IMDB Movie Review Dataset is used throughout the project and is divided into *25,000 highly polar movie reviews for training, and 25,000 for testing* [5]. The supervised IMDB-dataset was imported using TensorFlow due to its large size and was converted to NumPy arrays for simplicity when creating the Naive Bayes model from scratch. For certain other models, the input data was loaded into data structures as required for each model (NumPy arrays and Pandas Dataframes). The raw data was in the form of an array of movie reviews in the form of textual sentences (converted to lowercase) and was labeled as either positive sentiment (1) or negative sentiment (0). Additionally, we fixed all quotation marks, removed HTML tags, backslashes and hexadecimal characters. Also, we used the *vectorized* using the scikit-learn function CountVectorizer() in different configurations depending on the model, in order to convert the raw sentence data into a numerical Bag of Word (BOW) representation. BOW counts the number of times a word is repeated in the dataset and is represented as a matrix of token counts. The transformer models all used the same tokenization techniques specific to transformer models and provided by the simpletransformers library.

IV. Models and Implementation

A. Naive Bayes

The Naive Bayes algorithm used the Bag-Of-Words technique, which was done with sklearn's CountVectorizer function to give us the BOW matrix. With the parameters we set for this function, it would set a 1 in the returned array if the word was present and 0 otherwise. The algorithm counted the number of times a word was present in all the positive and negative sentences and then divided it by the total number of training examples. Laplace smoothing was then added to ensure no word would ever have a probability of 0, We chose α = β = 1 to ensure that the probability of a word would not equal 0 for both the positive or negative reviews if it did not exist for either one of them. In order to predict a sentence's sentiment, each word that was present in the sentence was multiplied by the probability in which it appeared for both positive and negative instances (to get positive and negative predictions). The output was then multiplied by the prior obtained from the training set. The greater positive or negative probability was chosen as the final prediction. Remarkably, Naive Bayes gave accurate results for the test dataset, given that features are considered to be conditionally independent. Section V-A described the results returned for the Naive Bayes in detail.

B. BERT

The Bidirectional Encoder Representations from Transformers, better known as "BERT", was introduced in 2018 and was built upon the existing unidirectional language models by incorporating attention and self-attention [6] mechanisms. This adds bi-directionality and allows a word to simultaneously gather context from both the left and right direction of a sentence [3]. The context of a word is learned through a pre-training objective using a *masked language model* (MLM) which inputs an incomplete sentence to the model and learns the optimal

weights in order for the output sentence to be the same as the original input. The self-attention mechanism encodes each input as a function of all of the other inputs. Due to the pre-training, if multiple nouns are present in a sentence, self-attention will encode the word *it* to be strongly dependent on the noun it is referring to and less dependent on other nouns. For example, *the animal didn't cross the street because it was too tired* — the word *it* will be strongly dependant on the noun *animal* rather than *street*.

The SimpleTransformers library was used to test the BERT model. It was trained on all the training data for one epoch with a learning rate of 1e-5 and a maximum sequence length of 64. After training and recording the accuracy results, we saved the model and then used the transformers library in order to quickly and painlessly output attention. These results can be found below.

V. Results

A. Task 1: Overall Model Comparison

Tables 1 and 2 demonstrate the training and test accuracies of all tested models, including the custom Naive Bayes model and the BERT model. We will describe the results of these two here, while detailed descriptions of the other models can be found in Section V-C.

After the Naive Bayes model completed its notably short training time (less than a minute), it returned a train accuracy of 85.268% and a test accuracy of 84.012%. Additionally, computing the confusion matrix for the test prediction revealed that the 84.012% accuracy was divided into 10306 True Positives (TPs), 1803 False Positives (FPs), 10697 True Negatives (TNs) and 2194 False Negatives (FNs). The higher amount of FNs over FPs, though slight, reveals that this model is more likely to classify a sentence containing both positively and negatively correlated words as being negative instead of positive, leading to a higher FN rate.

Meanwhile, the BERT model ran considerably longer (over an hour to train and evaluate) and returned an accuracy of 91.896% on the train set and of 90.388% on the test set. Its confusion matrix values on the test set were 11745 TPs, 1655 FPs, 10845 TNs and 755 FNs.

Model	Train Accuracy	Test Accuracy
Naive Bayes	85.268%	84.012%
BERT	91.896%	90.368%
ALBERT	92.080%	90.472%
XLNet	92.876%	91.844%
LogReg SAG	89.039%	86.860%
LogReg SAGA	89.024%	86.864%
SVM	100.000%	84.836%
NBSVM	94.140%	89.083%

TABLE I: Accuracy for all tested models, using both train and test data.

Model	TP	FP	TN	FN
Naive Bayes	10306	1803	10697	2194
BERT	11745	1655	10845	755
ALBERT	11745	1627	10873	755
XLNet	11847	1486	11014	553
LogReg SAG	10782	1718	10933	1567
LogReg SAGA	10781	1719	10935	1565
SVM	10722	1778	10487	2013
NBSVM	11163	1337	11112	1388

TABLE II: Confusion matrix values for all models on test data.

B. Task 2: Attention Matrix Values

After training the BERT model using the 'simpletransformers' library, we saved the model and loaded it into a 'transformers' model in order to use its attention output feature. A visual representation of the attention matrices are shown as seaborn heatmaps in Figures 1 and 2.

In both cases, we can see that the first attention layer has lower attention values throughout. However, the final layers have higher values and reveal connections between words. More specifically, we observe that the activation of the attention layer was larger on the diagonal. This shows that the BERT model indeed portrays that each word is correlated to the ones on either side of it. The model was able to learn the connection between words, which is expected since it is a bidirectional model.

From studying these figures, we observed a more active attention layer for the correctly classified examples shown in Figure 1. This indicates that the model is successfully making connections between relevant information in the input sequence in order to make accurate predictions. The model was able to correctly associate pronouns with nouns, verbs, and adjectives. One example of this is the high attention between "Damon" and "was", which reveals that BERT is correctly assigning relationships between words.

On the other hand, the graph of the incorrectly classified sentence shown in Figure 2 shows the opposite. In that case, the model is not correctly connecting relevant information in the input sequence, which leads to an inaccurate prediction. Once again, taking an example, we see some promising results, like the relatively higher attention between "problems" and "terrible". However,

the lack of attention to ”still” and ”like” is likely what prevents the model from correctly predicting the positive sentiment.

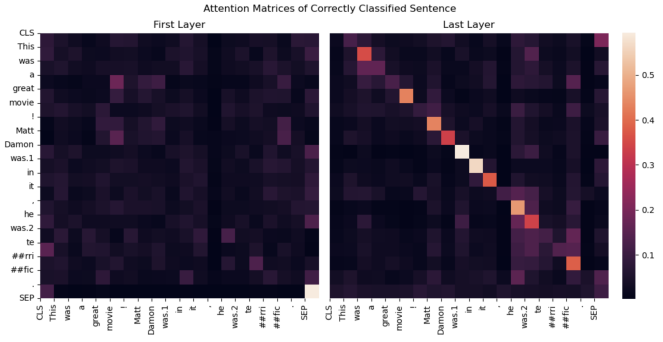

Fig. 1: Attention Matrices of Correctly Classified Sentences

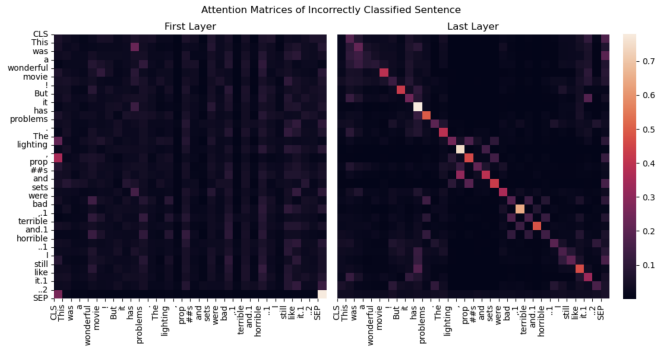

Fig. 2: Attention Matrices of Incorrectly Classified Sentences

C. Extra Model Implementations

Other models tested in this project are Logistic Regression, Support Vector Machine (SVM), Naive Bayesian Support Vector Machine (NBSVM), XLNet, ALBERT and MLP.

Logistic Regression and SVM were chosen due to their popularity and simplicity. Both models were tested using the Sklearn library.

We tested two variations of the Logistic Regression model: one using the Stochastic Average Gradient descent solver (SAG) and another using an augmented version of SAG known as SAGA [1], which adds L1 regularization and attempts to reduce training time. SAG is remarkably effective at training logistic regression models on large datasets because it uses random samples of previous gradient values when it trains the model instead of stepping through all values. As such, SAG (and its variant SAGA) were ideal choices on the large IMDb dataset. Logistic Regression with the SAG solver yielded a training accuracy of 89.039% and a test accuracy of 86.860%. SAGA ran slightly faster than SAG but yielded a slightly lower result of 89.024% and 86.864%. We also recorded the confusion matrix values on the test set, which had 10782 True Positives (TP), 1718 False Positives (FP), 10935 True Negatives (TN), and 1567 False Negatives (FN) for the SAG solver. The SAGA solver had 10781 TPs, 1719 FPs, 10935 TNs, and 1567 FNs. Thus, we can see that though there are more FPs than there are FNs, the difference is very slight, and the model is not much more likely to assign a Negative over a Positive.

SVM was tested in order to compare it with its variant, the NBSVM model, which is described in Wang and Manning’s paper. Specifically, we tested Sklearn’s linear support vector classifier with a regularization strength of 0.5. This model astonishingly returned a test accuracy of 100% but a lower test accuracy of 84.836%, indicating that this model is susceptible to over-fitting even though the test accuracy is comparable to the previous models. SVM also had 10722 TPs, 1778 FPs, 10487 FNs and 2013 FNs on the test set. Thus SVM behaves inversely to Logistic Regression; it has slightly more FNs than FPs, though, as before, the difference between the two isn’t incredibly large.

Unlike SVM, NBSVM is implemented using the Keras library. The data was also pre-processed using CountVectorizer, but in order to run NBSVM, it was also necessary to convert the matrix returned by CountVectorizer into sequences of Word-ID pairs. However, the matrix-form was still used to create the embedding matrices required for NBSVM, along with a custom function to calculate the Naive-Bayesian log-count ratios. This NBSVM model returned a training accuracy of 94.140% and a test accuracy of 89.083%, an improvement upon SVM, which reduces overfitting, as expected. This model had 11163 TPs, 1337 FPs, 11112 TNs, and 1388 FNs in its predictions on the test set. When compared to Logistic Regression and SVM, NBSVM seems to have even less bias since the difference in the amount of FPs and FNs is only 51 instead of 235.

The ALBERT model was implemented using the same libraries and data-loading as BERT. After a training time slightly longer than BERT’s, it returned a training accuracy of 92.080% and a test accuracy of 90.472% - only slight improvements over BERT. Its confusion matrix values were also similar: 11745 FPs, 1627 FPs, 10873 TNs, and 755 FNs.

XLNet was implemented the same way as the other transformer models. After fitting the data to the model, it obtained a training accuracy of 92.876% as well as a test accuracy of 91.844%. The test set’s predictions were divided with 12046 TPs, 11173 TNs, 1327 FPs, and only 454 FNs. Since XLNet is a cutting-edge model which is much more advanced than those previously

mentioned in this section, it is doubtful that the relatively larger amount of FPs than FNs is due to a biased model. Instead, it likely reveals that in this dataset, it is much more common for negative reviews to have positively-correlated words than it is for positive reviews to have negatively-correlated ones.

Given that BERT is a type of neural network, we also decided to test a feed-forward MLP model. The MLP model was implemented using linear layers and activation layers on top of them (same as MP2). The CountVectorizer function was once again used to give us a BOW representation for the training and testing examples. However, here the CountVectorizer instead returned the number of times the word occurred in the sentence. The input layer had the same number of input nodes, as there are unique words in all of the sentences. The idea is that each node represents a specific word, and the starting activation of each node is the number of times the word was present in the sentence. This model ran for 13.33 epochs and was trained on 15000 examples. The test accuracy was 49.540% on 10000 examples of the testing data. This is lower than what we hoped for. It is likely that with more training and by using the entire training set then, this model can reach higher accuracies. Regardless, BERT performed much better with only a single epoch, which clearly demonstrates the power of bidirectional learning as opposed to the feed-forward nature of an MLP.

VI. Discussion and Conclusion

Starting with BERT and Naive Bayes, the former took well over an hour to train and then to evaluate on both the train and test set, while Naive Bayes completed the same task in less than a minute with a respectable accuracy of 84.012%, especially considering the short time it took to reach this accuracy. This indicates that Naive Bayes can quickly get results that are moderately accurate and so would be an ideal choice in situations where false predictions are not especially problematic. However, it is necessary to use a model like BERT or its contemporaries if one desires to attain the highest accuracy possible, even if it comes at the cost of training time. It is worth noting that these models can also be tuned further, while Naive Bayes cannot. As such, their potential is even higher than what is shown here. Furthermore, our test on the MLP model clearly demonstrated the importance of bi-directional learning.

One must also keep in mind that transformer models should always be pre-trained on an external corpus when attempting to run similar tasks since a failure to do so would result in a need for more data, a longer training time, and lower accuracies for the same setup as described in this report. Training on an external corpus for sentiment classification gives the model a 'head start' so to speak, by providing the model with a greater understanding of language patterns, grammar, and contextual representations. By being exposed to a diverse range of text data from the external corpus, the pre-trained model can develop a better understanding of the nuances and intricacies of language, including the sentiment-related language used in movie reviews. An even better result could be obtained if the training was executed on an external corpus containing movie and/or TV-show related terminology.

While Naive Bayes performed reasonably well, it was still beaten by all the other tested non-transformer models. These models only took a few minutes longer to run compared to Naive Bayes. SVM improved its accuracy over Naive Bayes by less than 1%, while Logistic regression was 3% more accurate. The real outlier here is NBSVM, which nearly reached an accuracy of 90% - which would have put it well in line with the transformer models.

From all the above results, we can conclude that though the traditional non-transformer machine learning methods boast very fast training time and can still yield reasonable accuracy, they are no match for the highly-accurate (though slow) transformer models which consistently yielded accuracies of 90% or more. The only non-transformer model that comes close is NBSVM, but that model is highly-specific to this task, and yet it still falls just short. Furthermore, one must keep in mind that the transformers' great performance is due to their pre-training on external corpora which gives them a greater understanding of linguistic nuances, which are lost in the simplistic approaches of the non-transformer models due to their conditional independence assumptions. Additionally, it is worth noting that the transformer models tested here did not have fine-tuned hyper-parameters; greater results could thus be reached through extensive testing.

VII. Statement of Contributions

- Maxim Boucher: Helped load data, implemented and ran testing on all models except for NB. Wrote the majority of and helped review the report.
- Allison Mazurek: Implemented data processing and attention-value graphing. Helped implement BERT and helped write/review the report.
- Youssef Samaan: Implemented the Naive Bayes + MLP models and BERT attention-extraction, helped implement BERT, and helped write/review the report.

REFERENCES

[1] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives. 2014.

[2] Jacob Devlin and Ming-Wei Chang. Open sourcing bert: State-of-the-art pre-training for natural language processing, Nov 2018.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[4] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019.

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[7] Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. pages 90–94, 07 2012.

[8] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.