# IMDB Movie Reviews Classifier: Comparison of BERT vs Naïve Bayes

## Introduction:

Sentiment Analysis is the method or process of understanding the sentiment or emotions from text. The text can be social media comments, movie reviews, blogs etc. As a field of research, sentiment analysis deals with understanding people’s opinion towards different subjects, products, events or organizations [1].

There are three main types of approaches in sentiment analysis [1]

1. **knowledge-based techniques**: These methods classify text based on presence of specific, unambiguous key words such as happy, sad and bored. Some of these approaches also assign words a “probably affinity” based on context and emotions. These techniques perform poorly in examples where text contains mixed emotions, negations, or sarcasm.
2. **Statistical methods**: These methods try to identify statistical relationships and patterns by looking at a large number of examples, and predict the sentiment based on that understanding. A number of techniques such as bag of words, latent analysis, mutual information, etc. can be categorized as traditional approaches. Deep learning is a new approach of processing entire sequence of text to extract contextual information in form of word embeddings, which can be later used for classification.
3. **Hybrid approaches** leverage both machine learning and elements from knowledge based techniques such as ontologies and semantic networks in order to detect semantics that are expressed in a subtle or ambiguous manner, e.g., through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.

Sentiment analysis is useful in a variety of applications and scenarios. For a company or brand, it can be critical to understand the “pulse of their consumer”, what the customer likes and what isn’t working well in the market. Governments can use sentiment analysis to keep track of public option to key policies and events, and anticipate large-scale issues, security threats, and problems ahead of time. In recommendation systems used widely in e-commerce and other applications, tracking consumer sentiment is useful in recommending the best products which can have a significant impact on brand loyalty and profitability.

A variety of traditional machine learning techniques such as naïve bayes, support vector machines, random forest etc. have been reported in literature [2]. The typical approach with these models is to convert text into bag of words and calculate numerical features based on term frequency (such as TF-IDF: ratio of term frequency to inverse document frequency). A classification model is then trained based on these numerical inputs [2].

More recently, deep learning techniques such as Recurrent Neutral networks and Transformer models have been applied to the problem and reported improved accuracy [3]. These models make use of entire sentences as inputs, hence can take into account not only frequency of occurrence but also relationship between the words in the text and improve the classification accuracy [3]. Recurrent Neural Networks such as LSTMs have been shown be very effective in correctly classifying a wide variety of natural language tasks. However, a major bottleneck is that their training time increase exponentially with the size of the dataset, and they are not amenable for parallel processing using GPUs. On the other hand, Transformer models which are based on the attention mechanism to identify word relationships, are well suited for parallelization, and hence faster to train using GPUs.

BERT (Bidirectional Encoder Representation of Transformers) is a large neural network language model (300 million parameters) based on the transformer architecture [5]. It is pre-trained on a large corpus of unlabeled data, and it well suited for a wide-variety of classification tasks by fine-tuning an additional output layer [5]. This process known as transfer learning provides word embeddings based on the context in its training sentences, which can be used downstream to train models for specific classification tasks [6].

## Problem Statement

In this project, given a large number of movie reviews, the goal is to build a machine learning system which can accurately classify the review as either being positive or negative sentiment.

## Dataset

The IMDB movie reviews dataset consists of 25,000 highly polarizing movie reviews [7]. Each review is tagged as “positive” or “negative”. It is well suited for text classification using supervised learning approaches. The raw data is equally divided in *train* and *test* folders with further division into pos (positive) and neg (negative) reviews. Each review within these folders is available in a text file. For this project, in order to reduce the GPU training time for the BERT classifier, we only consider 1000 reviews from the dataset, with equal number of positive and negative labels.

## Solution Statement:

We build a sentiment classifier using a pre-trained BERT model and compare the performance in terms of accuracy and ROC-AUC metrics with a Naïve Bayes classifier for the IMDB movie reviews dataset [7]. We make use of the AWS Sagemaker service for data processing, exploration, model training and evaluation.

## Baseline Model

The performance of the BERT classifier is compared against a naïve Bayes classifier.

In supervised machine learning, Naive Bayes are a set of algorithms based on the Bayes theorem, and assume each feature in the model is independent given the dependent variable [8]. The conditional independence helps simplify the relationship between dependent (x) and independent (y) variables. There are various flavors of this approach based on assumption about the prior distribution P(x|y).

In spite of strong assumptions about conditional independence, Naive Bayes algorithms show good performance on document classification, spam filtering etc. The main advantages and disadvantages of naïve Bayes methods are

* because of the simplicity, they are extremely fast compared to other methods such as random forest, neural networks etc.
* their parameters can be estimated with small amount of training data.
* they dont suffer from curse of dimensionality, since each feature is assumed independent.
* a major disadvantage is that Naive Bayes methods are not good estimators, their probability predictions are not reliable.

MultinomialNB in scikit-learn [8] implements the naïve Bayes algorithm for multinomially distributed data, and is one of the two classic naïve Bayes variants used in text classification. Typically, the text data is converted to numerical features as word counts, n-grams, or TF-IDF (term frequency – inverse document frequency) representation. The parameters are estimated by a smooth version of maximum likelihood, which is equivalent to counting relative frequency of each class. Alpha is a smoothing parameter which accounts for features not present in the learned samples, and prevents zero probabilities in the predictions.

## Evaluation Metrics:

The baseline naïve bayes model and the BERT classifier are compared based on their accuracy and AUC-ROC metrics.

* **Accuracy** is defined as the ratio of number of labels (positive and negative) correctly classified by the model to the total number of labels in the test/evaluation set.
* **AUC-ROC** is the area under the receiver operating characteristic (ROC). ROC is the plot of true positive rate to the false positive rate of the model at various threshold levels of the probability score. The area under this curve is a good indicator of model performance under a variety of scenarios where different weightage is given to false positive vs false negatives.

## Project Details:

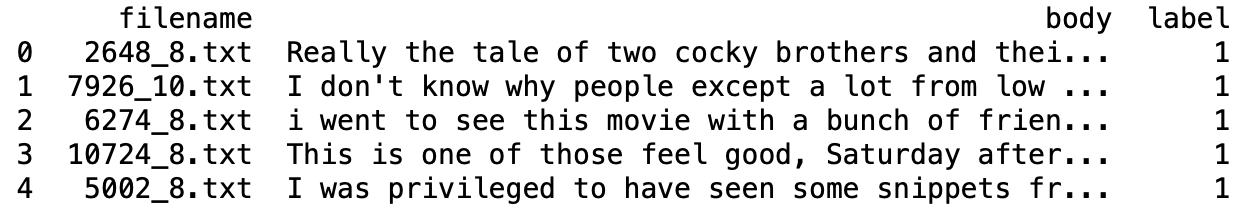
The project is implemented in following steps

1. **Create workspace on AWS Sagemaker.**

The entire project is executed within AWS Sagemaker service. A new notebook instance was created with a custom lifecycle configuration to install all the required dependencies (libraries such as pytorch, transformers etc.) every time the instance is launched.

1. **Download and process input dataset.**

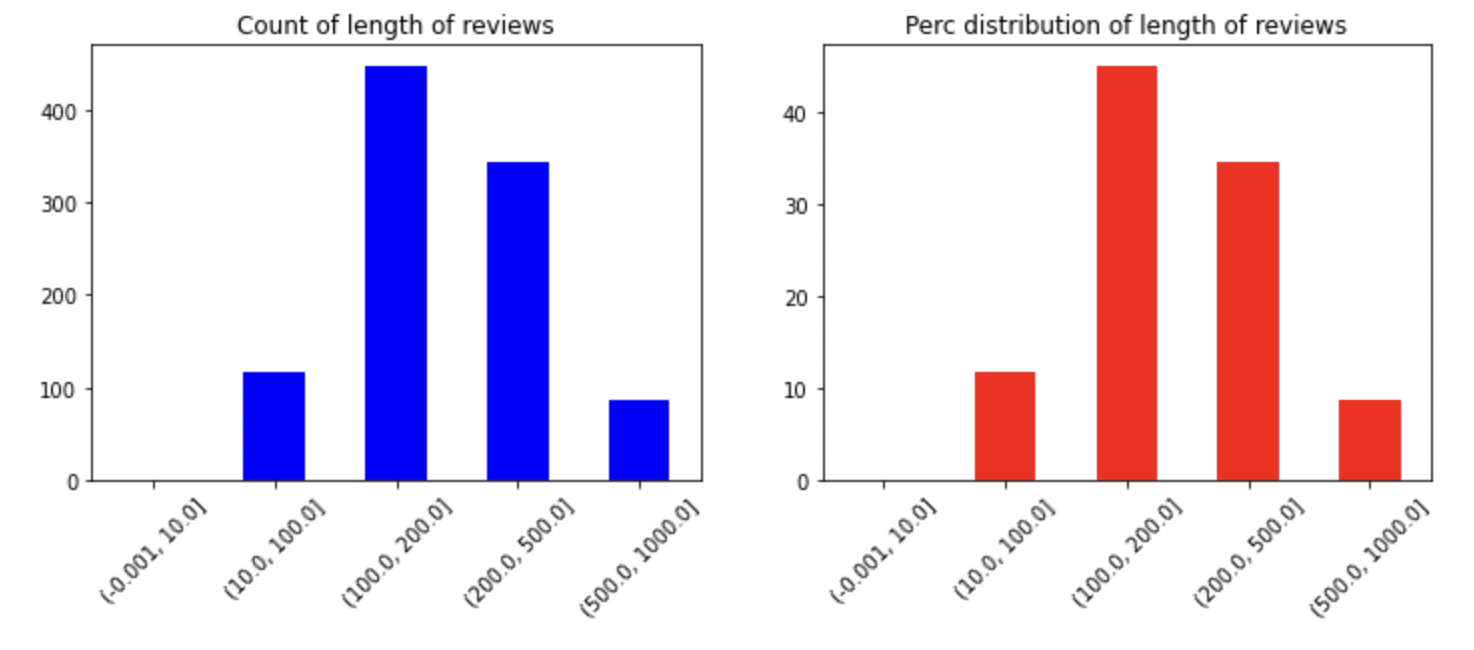
The data processing is done in *preprocess.py* within the *IMDBDataProcessor* class. First the dataset is downloaded and decompressed in the *inputs* folder. The text files are parsed and concatenated to a pandas dataframe for each of train and test sets. In order to expedite the training process, especially for the BERT classifier, we select a sample of 500 positive and negative reviews. The dataframe has 3 columns, filename: indicating the filename of the review, body: text of the review and label: binary 0 (negative) and 1 (positive). A sample of the dataframe is shown in Fig 1. Below



***Figure 1. Sample data from the train dataframe***

1. **Data Exploration.**

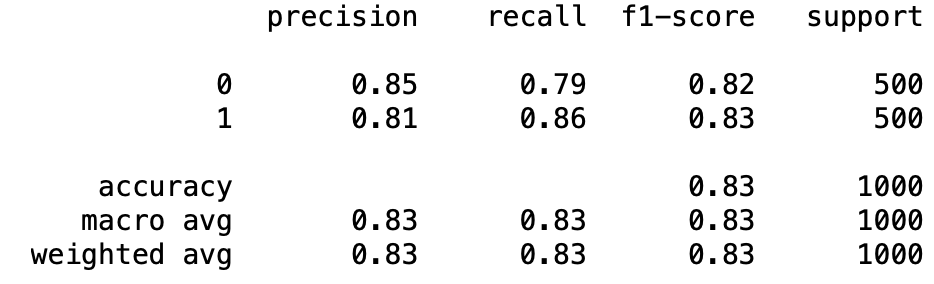
### Next some data exploration is conducted to understand finer details about the dataset. It is a balanced dataset since we have equal number of positive and negative reviews by design. We plot the distribution of length of the reviews in Fig 2. The length of the reviews follow an approximate normal distribution. About 90% of the emails have between 100-500 words. 10% of the messages have (10, 100) words or more than 500 words. The average value of the distribution is close to 200, which is used in the subsequent section for the padding length during the tokenization process.



**Figure 2: (a) Count and (b) Percentage distribution of the length of reviews**

1. **Baseline Model**

We use Naïve bayes classifier (multinomialDB) as our baseline classifier. We use the CountVectorizer to convert text to count of tokens with n-grams of length 1-3. Hence we have count features of individual words, bi-grams and tri-grams in the corpus. The classification is done with default settings for the multinomialDB classifier in scikit-learn. The model is trained on the train set and performance reported on the test set. The classification report as shown in Fig. 3 show **accuracy = 0.83.** The precision and recall are 0.81 and 0.86, which suggests a pretty good well-rounded model.



**Figure 3: Confusion matrix for the baseline model**

1. **Tokenizer for BERT**

In this section we use the DistilBertTokenizer from transformers [9] library for converting the text sentences into tokens based on the corpus on which BERT has been trained on. The pad length is set at 200 based on the text length distribution analysis. The tokenizer outputs sequence of tokens and a list of attention masks. The attention vector is a list of 0 and 1, which provide reference to which words/tokens the BERT model has to pay attention. The token sequence, attention masks and the target labels are converted to pytorch tensors and saved as pickled objects for later use. The saved tensors are uploaded to S3 bucket for use during the training and testing phase

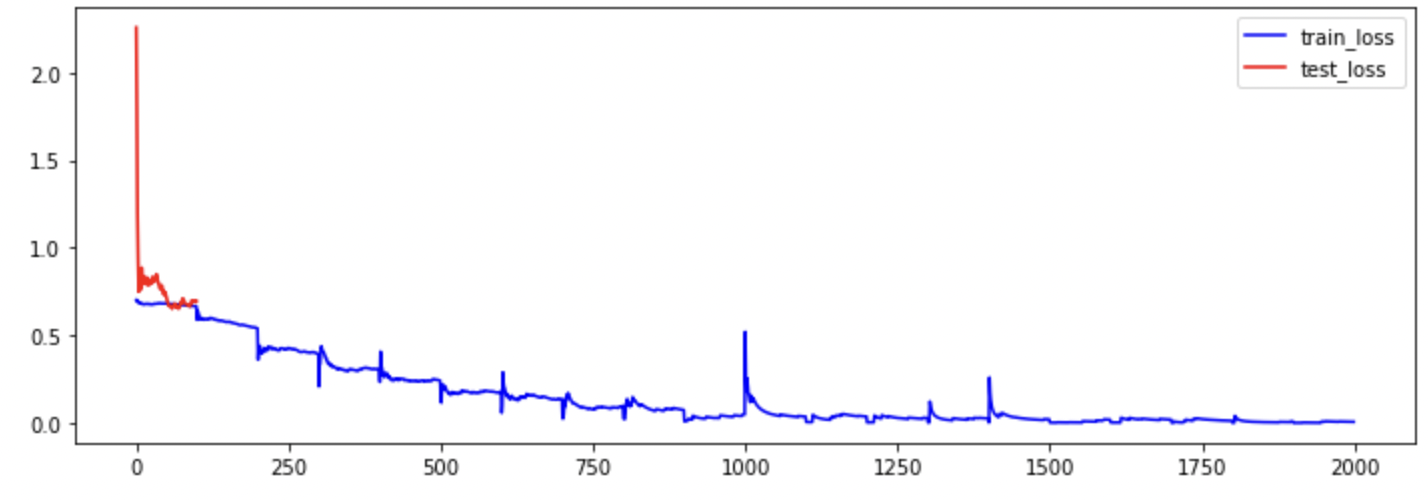
1. **Training and Evaluation of the BERT model**

The model architecture is pretty simple and consists of the pre-trained BERT model from *pytorch\_pretrained\_bert* module followed by linear layer of size (768,1) [10]. The model training is conducted using the Adam optimization function with fixed learning rate of 3e-6, and using the BCEWithLogitsLoss loss function from pytorch library. A dataloader object is created by loading the token sequences and attention mask tensors from S3. The model is then trained for 20 epochs with a batch size of 10. All the necessary methods for training are placed in train.py file. The entire training is conducted on a GPU instance by calling the Sagemaker Estimator object.

After the model has been trained for 20 epochs, the performance is evaluated by predicting outcome on the test set. The losses and predicted outcomes (probabilities) are uploaded to the S3 bucket specified in the estimator configuration.

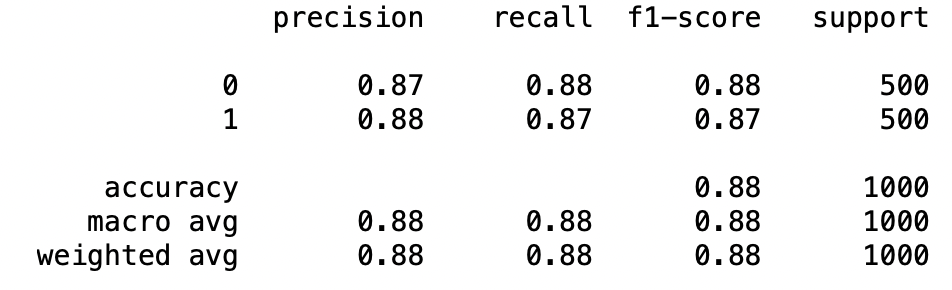
1. **Evaluation of model performance**

The losses from the training and test phase, as well as predicted probabilities from the test phase were downloaded from the S3 location. Figure 4 shows the variation of loss vs iterations over the different epochs during training and for the test phase. Overall, the loss decays rapidly from 0.5 to less than 0.05 over 20 epochs. The fluctuations are caused because of the small batch size (10) which result in stochastic behavior.

****

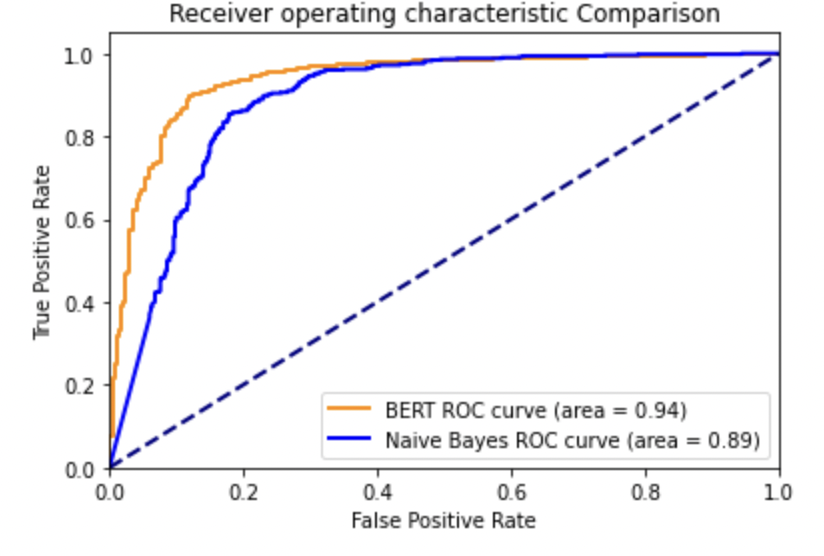
**Figure 4. Variation of loss with iteration for training and test phases**

The confusion matrix based on performance on the test set is shown in Fig 5. The BERT model without any hyper-parameter tuning achieves an accuracy of 0.88 (as compared to 0.83 for the naïve bayes classifier). The precision and recall numbers are also higher compared to the baseline model, suggesting there are fewer false positives and false negatives with the BERT classifier.

****

**Figure 5. Confusion matrix for performance of BERT model on the test set.**

A more comprehensive picture of the model performance is gained by comparing the ROC curves for the two models in Figure 6. The BERT classifier demonstrates better performance at lower FPR (false positive rate) thresholds as compared to the naïve bayes classifier. The AUC score which measures the area under the ROC curve is 0.94 for the BERT model vs the baseline model.

****

**Figure 6. ROC curves and AUC scores for the BERT and naïve Bayes classifier**

## Conclusion

In this project, we compared the performance of a pre-trained BERT model with a Naïve Bayes classifier in classifying a movie review as “positive” or “negative” in the IMDB movie reviews dataset. The BERT classifier provided an accuracy of 0.89 compared to 0.83 for the naïve bayes classifier. The area under the ROC curve was 0.94 vs 0.89 for the BERT vs naïve bayes classifier.

**References**

[1] <https://en>.wikipedia.org/wiki/Sentiment\_analysis

[2] Olga Kolchyna, [Tharsis T. P. Souza](https://arxiv.org/search/cs?searchtype=author&query=Souza%2C+T+T+P), [Philip Treleaven](https://arxiv.org/search/cs?searchtype=author&query=Treleaven%2C+P), [Tomaso Aste](https://arxiv.org/search/cs?searchtype=author&query=Aste%2C+T), *Twitter Sentiment Analysis: Lexicon Method, Machine Learning Method and Their Combination*, [arXiv:1507.00955](https://arxiv.org/abs/1507.00955)

[3] Walaa Medhat, Ahmed Hassan, Hoda Korashy, *Sentiment analysis algorithms and applications: A survey,* Ain Shams Engineering Journal, Volume 5, Issue 4, 2014,

Pages 1093-1113, ISSN 2090-4479, <https://doi>.org/10.1016/j.asej.2014.04.011.

[4] Zhang, L,  Wang, S,  Liu, B.  *Deep learning for sentiment analysis****:****A survey*. WIREs Data Mining Knowl Discov.2018; 8:e1253. <https://doi>.org/10.1002/widm.1253

[5] [Jacob Devlin](https://arxiv.org/search/cs?searchtype=author&query=Devlin%2C+J), [Ming-Wei Chang](https://arxiv.org/search/cs?searchtype=author&query=Chang%2C+M), [Kenton Lee](https://arxiv.org/search/cs?searchtype=author&query=Lee%2C+K), [Kristina Toutanova](https://arxiv.org/search/cs?searchtype=author&query=Toutanova%2C+K) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018, arXiv:1810.04805

[6] <https://www>.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/

[7] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

[8] <https://scikit-learn.org/stable/modules/naive_bayes.html>

[9] [Victor Sanh](https://arxiv.org/search/cs?searchtype=author&query=Sanh%2C+V), [Lysandre Debut](https://arxiv.org/search/cs?searchtype=author&query=Debut%2C+L), [Julien Chaumond](https://arxiv.org/search/cs?searchtype=author&query=Chaumond%2C+J), [Thomas Wolf](https://arxiv.org/search/cs?searchtype=author&query=Wolf%2C+T) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2020 [arXiv:1910.01108](https://arxiv.org/abs/1910.01108)

[10] https://www.kaggle.com/atulanandjha/bert-testing-on-imdb-dataset-extensive-tutorial