Advanced Data Analysis And Machine Learning
BBC News Classification

November 25, 2024

Bright Wiredu Nuakoh

The BBC provides a wide range of news content, often spanning single or multiple pages, to cater to diverse interests. While some individuals may focus on sports coverage, others may be more interested in political developments. Effectively classifying BBC news content by specific categories—such as sports, politics, or business—can offer valuable insights. This targeted approach allows users and organizations to concentrate on the aspects most relevant to their objectives, ultimately aiding strategic decision-making and fostering business growth. Our goal is to classify BBC text documents, using a pretrained tranformer and compare its performance to Long Short-Term Memory networks (LSTMs) to capture the sequential dependencies between characters in our data.

Data Description And Visualization

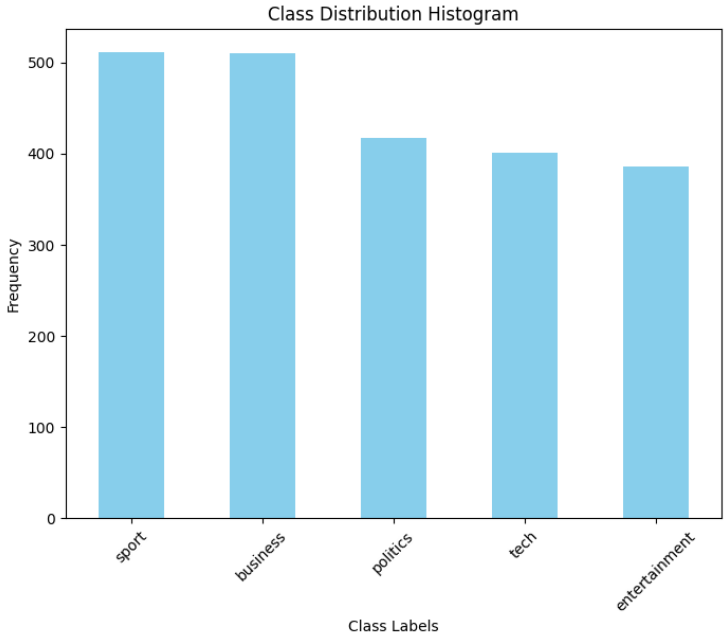

Figure 1: Distribution of classes in our data

The news dat is categorized into five sets knows as sport, tech, politics, business and

entertainment. This labels describing each text spans one column of the data and their respective texts forming the only feature making our data to have 2 columns with 2225 observations. The text in each observation is unprocessed and contains special characters, requiring data preprocessing. The histogram that displays the distribution of class labels within the dataset is shown in Figure 1

From Figure 1, sport and business categories have the highest frequency, each with over 500 occurrences, indicating they are the most represented. In contrast, politics, tech, and entertainment have slightly lower frequencies, ranging between 400 and 420. This distribution suggests a relatively balanced dataset, though sport and business are slightly more dominant compared to the other classes.

Data Preprocessing and Onboarding

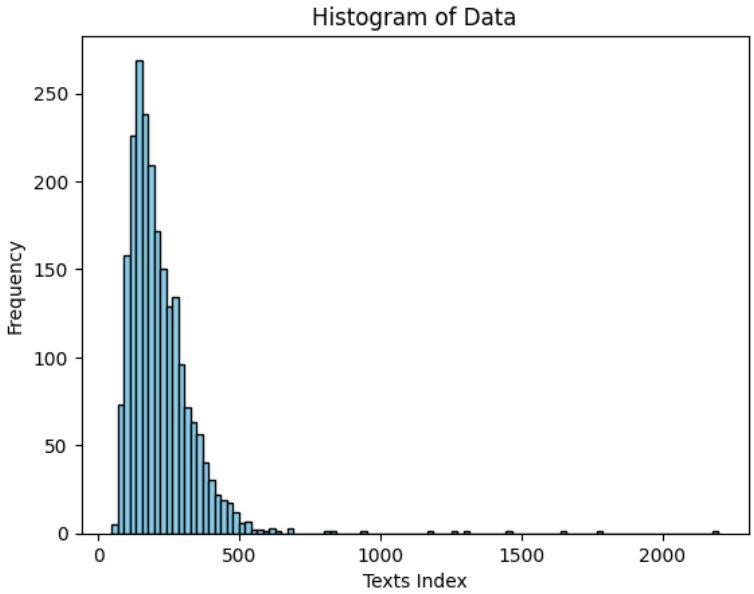

Figure 2: Histogram of lengths for Vacubularies in Processed Data

In the data preprocessing and onboarding process for a transformer model like BERT, raw textual data undergoes several essential steps to ensure readiness for training. The TextDataset class, designed using PyTorch, encapsulates these steps efficiently. During initialization, the raw text data and corresponding labels are tokenized using a BERT tokenizer, which converts text into input tokens and attention masks. Each input is padded or truncated to a specified maximum length to maintain uniformity. The dataset class leverages PyTorch's Dataset functionality to provide an iterable structure for model training. In the getitem method, each text instance is tokenized into input IDs and attention masks, ensuring that sequences are properly formatted for BERT's input requirements. Labels are converted into tensors for compatibility with PyTorch models. This

preprocessing pipeline ensures that our data is appropriately structured and ready for the transformer model to learn effectively, facilitating accurate sequence classification. From Figure 2 distribution is heavily skewed to the right, suggesting that the majority of texts are relatively short, with only a few exceeding 500 tokens. Very long texts (above 1000 tokens) are rare. This distribution suggests that setting an appropriate maximum sequence length, such as 100-500 tokens, would efficiently capture the majority of the data without excessive truncation or padding, optimizing the performance of models like BERT.

Model Definition And Architecture

Our objective is to develop a deep learning architecture for BBC news text classification using transformer models. Additionally, we will implement an LSTM network to compare its prediction accuracy with that of the transformer-based model. We will also analyze the impact of various hyperparameters on the performance of both models to understand their influence on overall accuracy.

Model Architecture And Training

The BERT model architecture comprises several key components designed for sequence classification. The Embeddings Layer converts input tokens into high-dimensional vectors while encoding contextual information. It includes word embeddings that map tokens from a 30,522-size vocabulary into 768-dimensional vectors, position embeddings that encode token order within sequences up to 512 tokens long, and token type embeddings that differentiate between sentence segments (up to two). Layer normalization and dropout (with a 0.1 rate) stabilize training and prevent overfitting. The Encoder Layer is the model's core, consisting of 12 identical transformer layers (BertLayers). Each layer features a self-attention mechanism utilizing scaled dot-product attention with query, key, and value matrices (each of 768 dimensions) to capture contextual relationships. Dropout (0.1) and layer normalization are applied for stability. The intermediate layer expands the hidden state dimension from 768 to 3072 using a linear transformation, followed by a GELU activation function. The output layer reduces the representation back to 768 dimensions with additional layer normalization and dropout. The Pooler Layer processes the CLS token from the final encoder layer, applying a linear transformation followed by a Tanh activation to produce a fixed-size sequence representation for downstream tasks. Finally, the Classification Head includes a dropout layer (0.1) and a linear layer that maps the 768-dimensional input to 5 output classes, enabling multi-class classification.

From Figure 4, the training shows rapid improvement in accuracy, with the training accuracy reaching nearly 100% by epoch 5, while validation accuracy peaks around 98% at epoch 3 and slightly declines to 96%, suggesting potential overfitting influencing our decision to use only 3 epochs for futher analysis. The training loss decreases sharply

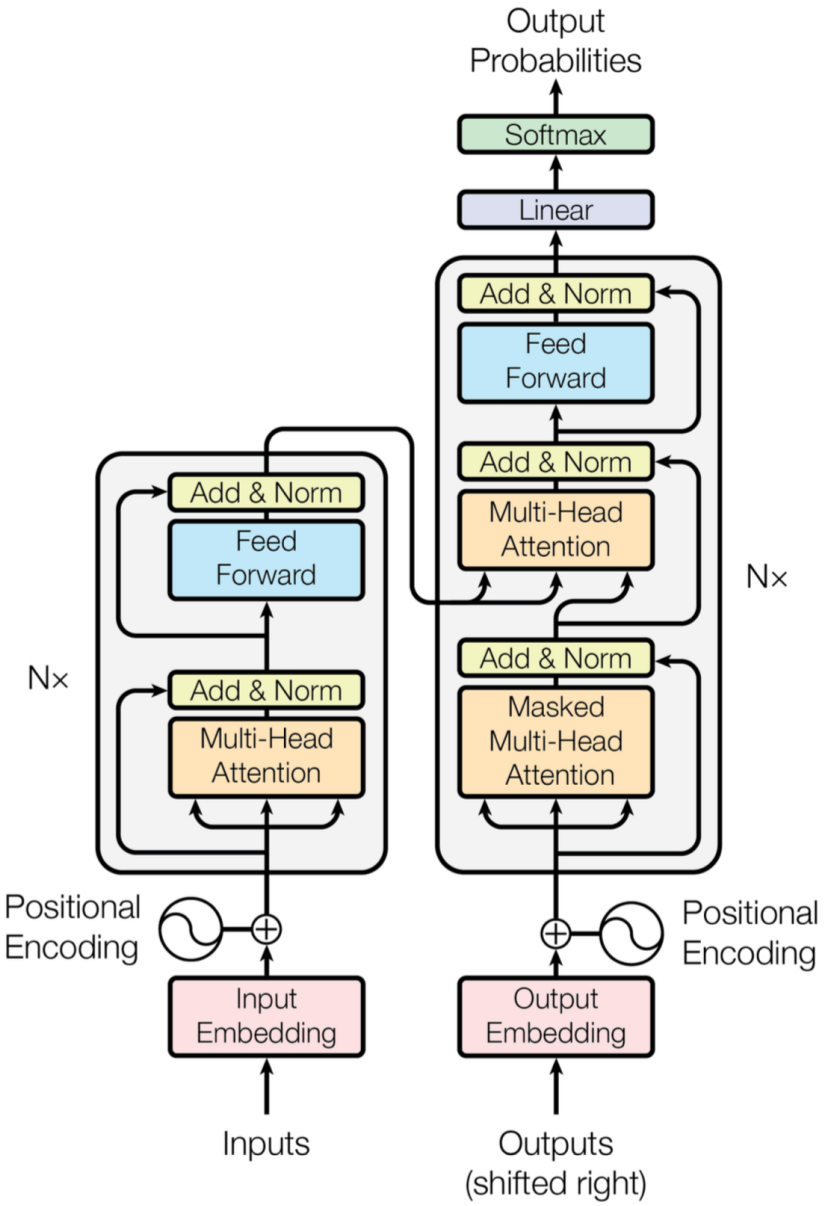

Figure 3: BET transformer , source: https://arxiv.org/pdf/1706.03762.pdf

from 25 to near zero, while the validation loss decreases more gradually but maintains a noticeable gap from the training loss. Overall, the model learns the training data well but may benefit from early stopping or regularization to improve generalization.

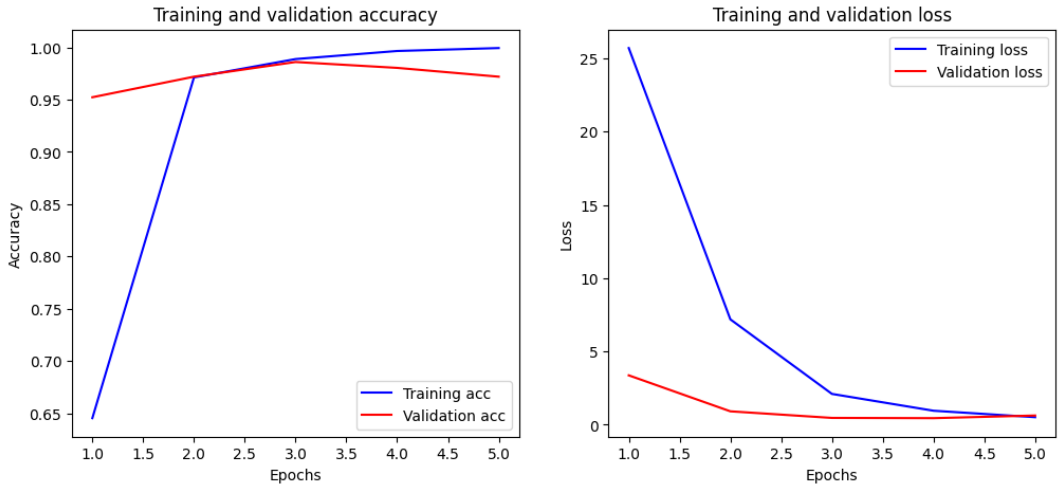

Figure 4: Training Progress of the BERT transformer

Experiment	BS	lr	AH	HL	Training accuracy	Test accuracy	Time (s)
1	32	3e-5	4	12	0.9895	0.9719	125.93
2	32	3e-5	4	24	0.9881	0.691	126.36
3	32	3e-5	8	12	0.9888	0.9607	125.93
4	32	3e-5	8	24	0.9916	0.9775	127.06
5	32	3e-1	4	12	1.980	0.2163	119.20
6	32	3e-1	4	24	0.2065	0.2444	118.90
7	32	3e-1	8	12	0.2065	0.21630	118.69
8	32	3e-1	8	24	0.2008	0.2167	116.71
9	64	3e-5	4	12	0.9888	0.9719	126.14
10	64	3e-5	4	24	0.9902	0.9607	126.36
11	64	3e-5	8	12	0.9881	0.9691	125.74
12	64	3e-5	8	24	0.9902	0.9719	125.59
13	64	3e-1	4	12	0.1924	0.2163	118.84
14	64	3e-1	4	24	0.1931	0.1770	117.28
15	64	3e-1	8	12	0.2149	0.1770	116.93
16	64	3e-1	8	24	0.2001	0.2163	118.41

Table 1: Training and Test Accuracy with Different Hyperparameters

Hyperparameter Analysis

The training experiments on the BERT model demonstrate the significant impact of hyperparameters on both accuracy and training time. From Table 1. when using a learning rate of 3e-5 and batch size of 32, models with 12 or 24 hidden layers achieved high accuracy, with slight variations due to attention heads. For instance, a model with

8 attention heads and 24 hidden layers (Combination 4) reached a training accuracy of 99.16% and validation accuracy of 97.75%, outperforming some configurations with 4 heads. However, increasing the learning rate to 0.3 led to poor performance across all configurations, as shown by validation accuracies consistently below 25%. This suggests that a high learning rate hampers model convergence. Training time remained consistent across models, averaging around 125 seconds, indicating that the model's complexity (in terms of attention heads or layers) had a negligible effect on duration but significantly influenced performance outcomes.

Models Evaluation And Performance

The BERT model demonstrates excellent classification performance, achieving a high overall accuracy of 98.43%. The F1-scores across all classes are above 96%, indicating a balanced performance with minimal misclassification. The "Sport" and "Politics" categories show the highest precision and recall, while the "Tech" and "Entertainment" categories slightly lag in recall due to a few misclassifications. Overall, the model is highly effective for this text classification task.

- **Precision, Recall, F1-Score for Each Class:**
  - **Tech**: Precision = 96.30%, Recall = 97.50%, F1-Score = 96.89%
  - **Sport**: Precision = 100%, Recall = 98.04%, F1-Score = 99.01%
  - **Entertainment**: Precision = 100%, Recall = 96.10%, F1-Score = 98.01%
  - **Business**: Precision = 97.14%, Recall = 100%, F1-Score = 98.55%
  - **Politics**: Precision = 98.82%, Recall = 100%, F1-Score = 99.41%

BERT Tranformer and LSTM Performance

MODEL	Epochs	Training Time	Train Accuracy	Test Accuracy
BERT Transformer	3	124 secs	99%	97%
LSTM	20	31 secs	85%	86%

Table 2: Comparison of model performance and training statistics

The comparison between the BERT Transformer and LSTM models highlights significant differences in performance and training efficiency.

The BERT Transformer outperforms LSTM in both training and testing accuracy, achieving 99% and 97% respectively, compared to LSTM's 85% training accuracy and 86% test accuracy. This indicates that BERT is much better at capturing complex patterns in the data.

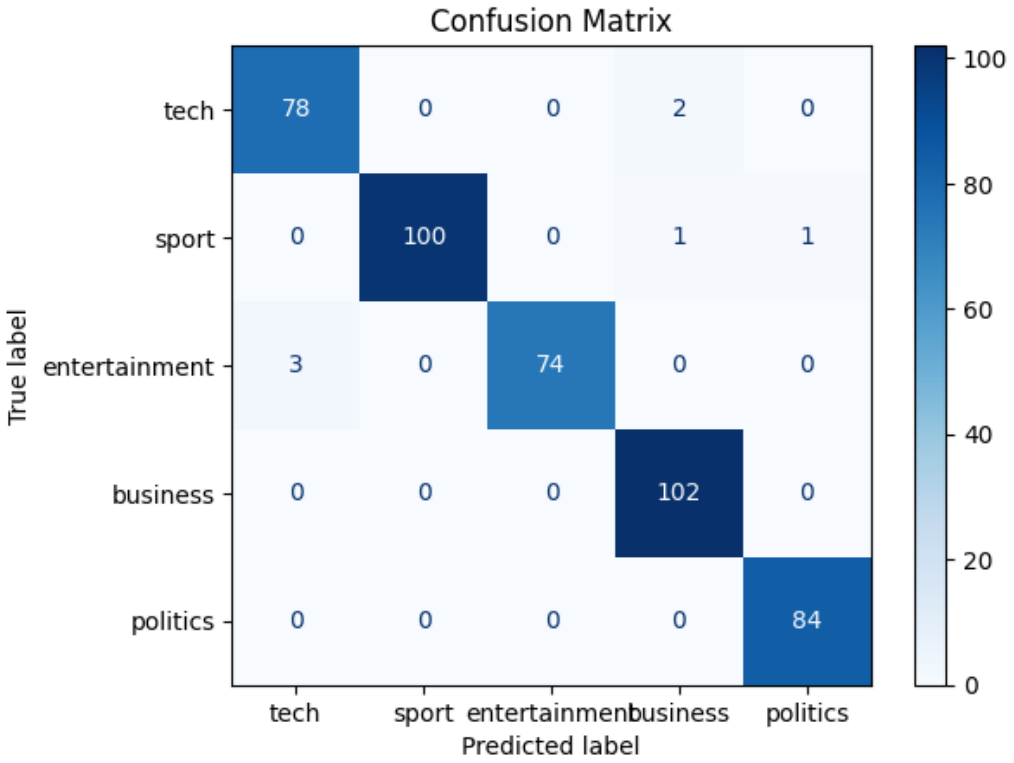

Figure 5: Confusion matrix showing the accuracy for each predicted Label by the BERT.

Despite its superior performance, BERT requires more time per epoch compared to LSTM . However, BERT achieves its results in just 3 epochs, whereas LSTM requires 20 epochs. For LSTM to get good accuracy, much epochs and trainging time might be required.

BERT demonstrates excellent generalization, with only a 2% drop between training and test accuracy, compared to LSTM's 1%. However, LSTM's lower overall accuracy limits its practical applicability. While BERT Transformer is computationally more expensive per epoch, its higher accuracy and ability to converge quickly make it the superior choice for the BBC text classification tasks. LSTM, on the other hand, may be suitable for scenarios with limited computational resources but is less effective for tasks requiring high accuracy.