CIS 4190/5190: Applied Machine Learning Fall 2024

Classification of News Headlines

Team Members: Serena Gu, Amber Wang, Yamin Zhang *Project: [News Source]*

1 Summary

This project focused on classifying news headlines into two sources, Fox News and NBC News, using Support Vector Machines (SVM) and BERT models. We started by collecting and cleaning a dataset of 3,805 headlines. Data preprocessing included text normalization, stopword removal, and lemmatization.

For the SVM model, we used TF-IDF vectorization with n-grams and punctuation counts as features, achieving a peak accuracy of 80.03% when combining text complexity features like readability. The BERT model, utilizing a pre-trained transformer architecture, performed best on unprocessed text, reaching an accuracy of 79.11%. Through experiments, we observed that preprocessing was critical for SVM but detrimental for BERT, demonstrating the latter's capacity to handle raw inputs effectively.

Our results from the exploratory questions demonstrated the strengths and trade-offs of traditional machine learning (SVM) versus pre-trained language models (BERT). While SVM required extensive feature engineering, it provided interpretable results. BERT, on the other hand, simplified preprocessing but was less customizable for downstream tasks. Despite different training processes we took, our two best models achieved comparable results, reflecting their unique strengths in natural language processing tasks. Limitations include potential overfitting in BERT and the scope of features explored for SVM. Future work could involve larger datasets, advanced regularization techniques, and additional feature engineering.

2 Core Components

2.1 Data Collection and Dataset

2.1.1 Data Collection

The dataset is provided with a CSV file containing URL and Source columns, where Source indicated the news source (Fox News and NBC News). To gather the headlines, a web crawler was developed using Python's BeautifulSoup and requests libraries to extract the Title for corresponding URL.

2.1.2 Data Cleaning

After collecting the raw data, a thorough cleaning process was undertaken to prepare the dataset for analysis and model training. The cleaning steps involved:

1. **Text Normalization:** The original Title was converted to lowercase and stripped of non-alphanumeric characters using regular expressions. This standardization helped in reducing variability caused by capitalization and punctuation.

2. **Stopword Removal:** Common English stopwords were removed to retain only the meaningful words that contribute to the headline's semantic content.

3. **Lemmatization:** Words were lemmatized to their base forms using NLTK's WordNetLemmatizer, which helps in reducing different forms of a word to a common base, thereby improving the model's ability to generalize.

4. **Handling Missing Data:** Any rows with missing or empty Title values after cleaning were removed to ensure data integrity.

2.1.3 Data Splitting for Evaluation

To ensure robust model evaluation, Stratified K-Fold Cross-Validation with five folds was employed. This method maintained the class distribution across each fold, ensuring that each subset had a representative proportion of Fox News and NBC News headlines. For each fold, four parts were used for training and one part for validation, allowing every data point to be utilized for both training and evaluation.

2.1.4 Final Dataset Composition

The final dataset, prepared for training and evaluation, encompassed a total of 3,805 news headlines, evenly distributed between the two sources:

- **Fox News:** 2000 headlines.
- **NBC News:** 1805 headlines.

Each entry in the dataset included:

- **Title:** The original headline extracted from the news article.
- **Source:** The binary label indicating the news outlet (0 for Fox News, 1 for NBC News).
- **Cleaned_Title:** The processed headline, free from non-alphanumeric characters, stopwords, and lemmatized.

2.2 Model Design

2.2.1 SVM Model

To classify text data (`Cleaned_Title`) into two `Source` categories (NBC and Fox), we used TF-IDF vectorization (top 10,000 features) with n-grams (1 to 3) for contextual patterns and added punctuation counts as a feature to improve accuracy. A SVM with a linear kernel was chosen for its effectiveness in high-dimensional data, using `class_weight='balanced'` to address the potential class imbalance. Performance was evaluated using a stratified train-test split and metrics like accuracy, precision and recall, achieving an accuracy of 80.42%.

To improve the model accuracy, we conducted a grid search with $C' : [0.01, 0.1, 1, 10]$ and found that the regularization parameter C should be set to 1, aiming for a balance between maximizing the margin and minimizing misclassifications, ensuring that the model does not overfit to the training data.

Next, we extracted sentiment scores for each text and classified them into positive, neutral, negative, and compound categories, adding more context for classification. After running the SVM with Sentiment Features, we obtained an accuracy of 79.89%.

Lastly, we added text complexity features by incorporating the average sentence length and readability of the titles. For the SVM with Complexity Features, we obtained an accuracy of 80.03%. Title length can vary based on the writing style of different sources, and readability is useful for detecting differences in target audiences.

2.2.2 BERT Model

While building the BERT model, we used `AutoTokenizer` to tokenize text data input compatible with the transformer model, padding sequences to a uniform length of 128 tokens for consistency. We used the pre-trained `bert-base-uncased` model to adapt for binary classification, with `num_labels` parameter set to 2 to specify the two output classes. We trained our model using `learning_rate=2e-5`, `num_train_epochs=2`, and `class_weight` effectively handled by the pre-trained architecture. Performance was evaluated using metrics such as accuracy, precision, and recall, achieving an accuracy of 79.11%.

To enhance model accuracy, we performed hyperparameter tuning on both batch size and learning rate, we determined that the optimal batch size was 16 and the best learning rate was $2e-5$, which both matched the initial setting.

We also experimented with variations of the original BERT model. Using the pre-trained `distilbert-base-uncased` model using the same tokenization process and training parameters, DistilBERT achieved a slightly improved test accuracy of 79.50%. However, when applied to the subset of test data, both BERT and DistilBERT predicted approximately 80% of the Sources correctly, suggesting that the models performed similarly in identifying the correct Source categories on the test subset.

Similarly, we used the pre-trained `roberta-base` model, which is trained differently from BERT. RoBERTa removes the Next Sentence Prediction task and trains with longer sequences and bigger batches, helping it to better understand language patterns. With RoBERTa, we fine-tuned the hyperparameters and determined that the optimal learning rate was $5e-5$ and the optimal weight-decay was 0.2. Using these fine-tuned parameters, we achieved a significantly improved test accuracy of 92.38%. However, when applied to a test data subset, its performance dropped to below 50%, indicating potential overfitting. Therefore we decided not to select this model for our final implementation.

2.3 Evaluation and Model Performance

To evaluate our models, we built an `evaluate_model` function to assess the model's performance using key metrics. These included accuracy, which captures the proportion of correctly classified instances, and a classification report providing detailed metrics like precision, recall, and F1-score for each class. It also generates a confusion matrix to visualize true positives, true negatives, false positives, and false negatives, identifying misclassification patterns.

To test our models internally, we used cross-validation to evaluate their performance by training and testing on different subsets of the data. The results showed that BERT achieved an average accuracy of 80.66%, while RoBERTa outperformed it with an average accuracy of 92.27%. The cross-validated result for SVM, with an accuracy of 80.32%, was also consistent. The results indicated that all of our models were reliable and robust in their performance across different subsets of the data.

Achieving an accuracy of 80.03% with the SVM model utilizing Complexity Features and 79.11% with the BERT model, we chose BERT as our final model due to its greater stability in handling diverse input formats and its ability to generalize effectively. In our experiments, the performance of the SVM model varied depending on the chosen vectorization method, whereas BERT demonstrated consistent stability across different hyperparameters and input variations.

3 Exploratory Questions

3.1 Question and Motivation

In this project, we explored two primary approaches: comparing the performance of Support Vector Machines (SVMs) with various BERT variations for NLP tasks. Surprisingly, despite the distinct preprocessing and training procedures we followed, both models achieved comparable accuracies after extensive experimentation. Specifically, for models with approximately 80% accuracy, we ran the SVM on preprocessed and vectorized text, whereas BERT operated on the original, unprocessed titles. This observation led us to investigate the following exploratory question:

Does SVM rely more heavily on text preprocessing and feature engineering compared to pre-trained language models?

3.2 Course Material and Prior Work

From the lectures, we learned that linear models, such as linear regression and logistic regression, are highly sensitive to feature engineering. For example, adding different input features can transform the

decision boundary from linear to non-linear, significantly affecting prediction results. In contrast, a key advantage of deep neural networks is their ability to automatically learn hierarchical representations of data with increasing complexity, reducing the reliance on manually designed features. This difference is also highlighted in the paper "A Comparison of SVM against Pre-trained Language Models (PLMs)".[1] In this study, preprocessing steps such as special character removal, tokenization, and lemmatization are applied before feeding data into SVM models. Additionally, the significance of feature engineering is demonstrated by a notable 15% performance drop when meta-data is excluded. In comparison, minimal to no feature engineering is performed for pre-trained language models. These findings led us to feed unprocessed input to BERT, while preprocessing the title for SVM when constructing our models. We also expected BERT to outperform the simpler SVM structures due to its ability to learn contextual and more hidden feature representations.

3.3 Methods for Investigation

For both SVM and BERT, we will keep the model structure and hyperparameters unchanged while varying the form of input fed into the models. Specifically, we will use the following SVM configuration across the board:

SVC(kernel='linear', C=1, random_state=42)

The following experiments will be carried out:

- SVM on the original title using Bag of Words (BOW)
  CountVectorizer(max_features=5000, ngram_range=(1, 2))
- SVM on the original title using TfidfVectorizer
  TfidfVectorizer(max_features=8000, ngram_range=(1,3))
- SVM on the original title using BERT Tokenizer
- SVM on the processed title using Bag of Words (BOW)
- SVM on the processed title using TF-IDF Vectorizer
- SVM on the processed title using BERT Tokenizer
- SVM with additional manually designed features, including average sentence length and readability

Similarly, we will also explore the following for BERT and its variations:

BERT: learning_rate = 2e-5, per_device_train_batch_size = 16, per_device_eval_batch_size = 16, num_train_epochs = 3, weight_decay = 0.01

RoBERTa: learning_rate = 5e-5, per_device_train_batch_size = 16, per_device_eval_batch_size = 16, num_train_epochs = 3, weight_decay = 0.2

- BERT on the original title
- BERT on the processed title
- RoBERTa on the original title
- RoBERTa on the processed title

Table 2: BERT Variant Models Accuracies

Data Type	**BERT**	**RoBERTa**
Unprocessed Title	79.11	90.54
Processed Title	75.43	77.40

[1] Yasmen Wahba, Nazim Madhavji, and John Steinbacher, *A Comparison of SVM against Pre-trained Language Models (PLMs) for Text Classification Tasks*, submitted on November 4, 2022

Table 1: SVM Model Accuracies for Different Input Representations

Data Type	BOW	Tfidf	BERT	Manual Features
Unprocessed Title	76.22	81.47	74.51	-
Processed Title	77.00	80.03	73.19	80.29

3.4 Discussion

From the SVM results, we observe that the choice of vectorization method has a greater impact on performance than the text preprocessing techniques. For the BERT variants, performance is generally higher on unprocessed titles, which suggests that pre-trained language models can effectively handle raw input. However, it is worth noting that the RoBERTa model's performance on unprocessed titles may be overfitting, as there is a significant discrepancy between our evaluations and its performance on the unreleased leaderboard test data. Another finding is that while the performance of the SVM model varies depending on many choices we made, the performance of BERT remains relatively stable, even with extensive efforts in hyperparameter tuning and input processing.

These results align with our expectation that SVM performance is more influenced by input representation, whereas pre-trained language models like BERT and RoBERTa rely less on traditional preprocessing and feature engineering, due to their pretraining processes. What surprised us is that our two best models, each optimized in separate realms—feature engineering for SVM and hyperparameter tuning for BERT—achieve similar accuracies. This highlights the distinct advantages of the two types of machine learning models. Pre-trained models like BERT save time by eliminating much of the need for manual feature engineering. At the same time, however, it is more constrained by the base model training and doesn't necessarily have a superior performance on downstream tasks. This adds more complexity to finetuning for customized goals. On the other hand, SVM offers a more interpretable structure. Despite greater demand for manual design, it has the potential to fit specific tasks very well.

3.5 Limitation

Our experimental results, particularly for the BERT variants, is likely being influenced by overfitting, which often occurs when we fit highly complex models to rather limited datasets. To address this issue, we could explore additional regularization techniques, such as incorporating dropout layers and L2 regularization during fine-tuning.

Moreover, the observation that our pre-trained language models show limited sensitivity to preprocessing variations in titles and other changes we made is likely because the models have not been customized enough for downstream tasks. Effectively addressing this could require more careful fine-tuning and potentially more training data, ensuring that what the model learns is beyond its pre-training process.

Finally, in terms of the manual feature engineering phase for SVM. What we have explored so far are features like readability, amount of punctuation used, word count, and sentiment score. These are selected from the basic exploratory data analysis we performed on the title scraped. Given more time, we could explore other possibilities like number of person/organization names mentioned and topic-related keywords. We could also explore different vectorization methods such as applying TFIDF with N-grams and Byte-Pair Encoding tokenization.

4 Team Contributions

We worked on all parts of this project together.