Comparative Analysis of Machine Learning and Deep Learning Algorithms for Spam Detection

Srivatsa Chidara
s.chidara@student.utwente.nl
s3105520

Abstract

This project investigates spam classification performance by comparing various machine learning and deep learning models, this include Support Vector Machine(SVM), Random Forest(RF), Bidirectional Long Short-Term Memory RNN(BiLSTM), and ALBERT. The analysis examines the effects of two feature extraction methods, namely Term Frequency - Inverse Document Frequency (TF-IDF) and Word2Vec on machine learning algorithms. Various pre-processing steps are examined to determine their effect on model performance. Results indicate that SVM with TF-IDF performs best. Pre-processing contributes positively to most models, but shows minimal impact on ALBERT. This project provide insights into feature selection and role of pre-processing, which can be used in research of robust spam detection.

1 Introduction

In today‘s digitally connected world, the amount of information exchanged through electronic communication channels, such as email, messaging applications, and social media has grown exponentially. Sadly, this growth has been accompanied by a significance rise in potentially harmful messages which are called as spam. Spam messages are not just junk, and clutter in the inbox. They serve as the gateway for many malicious activities such as phishing, malware, and various other kinds of malicious activities which can compromise the security. This project aims to explore various methods with a goal of improving the accuracy for the spam classification. Specially, I focus on comparing the performance among 4 algorithms,machine learning algorithms such as Random Forest, and Support Vector Machine, alongside the deep learning models, such Bidirectional Long Short-Term Memory (BiLSTM) (BILSTM) which is a type of RNN and transformer based model which is A Lite version of Bidirectional Encoder Representation from Transformers (ALBERT) (Lan et al., 2019). Additionally, I examine how various feature extraction methods, including TF-IDF and Word2Vec on ML algorithms, not DL algorithms..
By addressing these questions and finding the best performing algorithm along with feature extraction methods, this project seeks to contribute the development of the model that has large amount of accuracy and that can adapt to threats.

2 Related Work

(Sadia et al., 2023) compared multiple machine learning algorithms for spam detection focusing on the tweets and they employed multiple classification techniques like Naive bias, KNN, SVM, and more. This paper emphasizes the efficiency of content-based feature extraction and importance of algorithm selection. (Teja Nallamothu and Shais Khan, 2023) explored various ML techniques for spam detection. Their study evaluates that SVM particularly well with an accurate feature extraction for effective spam filtering. (Ghosh and Senthilrajan, 2023) conducted a comparative analysis of ML techniques. Their work evaluates various models across different datasets focusing on email spam. The study highlights that SVM and Random Forest consistently deliver Superior performance. (González-Carvajal and Garrido-Merchán, 2020) compared BERT, the transformer based model with traditional ML techniques for text classification. Their study highlights the superior performance in capturing contextual information outperforming SVM across various datasets. (Sahmoud and Mikki, 2022) their research demonstrated that BERT effectively captures contextual nuances in text, outperforming traditional ML models on various datasets. It highlights the BERT‘s ability to improve accuracy, especially for more complex language patterns in spam messages. (Yaseen et al., 2021) explored models such as CNN and LSTM. The study demonstrates that in particular LSTM provides a significant performance boost in identifying spam, especially in the large datasets. The paper emphasizes the advantages of deep learning over traditional methods in spam detection.

3 Data

This dataset for this project is acquired from Hugging-Face website (Hugging Face Dataset) which comprises the spam emails, subject of the email, and label(spam, ham). The data set is specifically designed for spam classification. There are 31,700 rows for training, and 2000 for testing. But, I joined all the data into single data-frame for better control over training, validation, and testing of the data.
Example: " *new clalls softtabs = lnstant rockhard erectlons simply disolve half a plll under your tongue 10 min before action , for results that last all weekend . normal retail is 19 / plll order from us today at the price of 3 . 67 not interested (0 pt - 0 ut) ." : spam*

4 Methodology

4.1 Overview of Approach

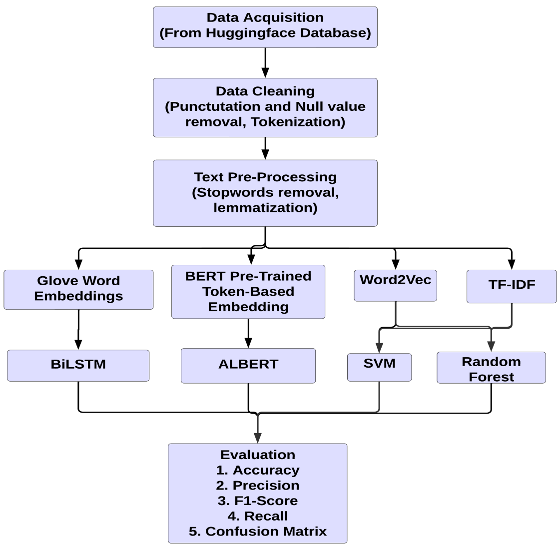


Figure 1: Overview of approach

4.2 Libraries and Tools

In this project, I have made use of various tools and libraries for different algorithms. For cleaning and pre-processing the data: **Pandas, Numpy, and Regex** are used. To normalize the text, **NLTK** library used with tools like **Tokenizer, stop-words, and Lemmatizer** are used. For vectorization of the text: **scikit-learn** library and **TF-IDF Vectorizer**, **Gensim** library : **Word2Vec** are used. For optimizing the hyper-parameters for all the algorithms **Optuna** library is used. For implementation of **SVM** and **Random Forest**: **Scikit-Learn** is used. For the implementation of **ALBERT and BiLSTM: PyTorch, Transformers from Hugging-face** (ALBERT Model Documentation) libraries are used. For accelerating the training of the deep learning models, **GPU(RTX 3050 Ti)** is leveraged by use of **NVIDIA CUDA**.

4.3 Dataset and Pre-Processing

As this project goal is spam detection, the dataset analyzed is "Enron Spam Email Dataset". It has 7 columns, but for this project two of them are needed, i.e, "label", and "text". The "text" column is concatenation of subject and message of the email. The dataset is balanced with spam:ham with 50.8% : 49.2% of dataset. There are few missing missing values, the rows are dropped. The data is further processed by lower-casing, removal of punctuations, removal of numbers and special characters, removal of white spaces, tokenization, stop-words removal, and lemmatization of words. As TF-IDF cannot process tokens of text, text is joined as a sentence and stored in a new column. As it is spam detection the numbers, special characters and punctuation hold a lot of semantic meaning for a spam text, in section 5.1, experiments are conduced with and without removal of these elements.

4.4 Feature Representation

There were a total of 4 types of word embedding methods are used, with different parameters which can compliment the algorithm used.

4.4.1 TF-IDF

The Term Frequency - Inverse Document Frequency vectorizer transforms the data into numerical features based on the importance of the words. More importance to frequent words, and less importance to common words. For the data after optimizing for the best parameters, it is implemented with max_features=12000 (vocabulary size), ngram_range=(1, 2).

4.4.2 Word2Vec

It is a neural-network based model that converts the words into vectors representation of many dimensions such that semantic meaning of the word is maintained which places alike words close in multi-dimensional space. For the data, after optimizing for the best parameters, these are used vector_size=300 , window=7, min_count=3, workers=7, sg=1 (skip-gram).

4.4.3 ALBERT: Pre-Trained Token-Based Embedding

The ALBERT tokenizer breakdown the text into tokens and convert them into vectors with numerical ID and map them to a unique integer from a pre-trained vocabulary. This is a type of embedding-based vectorization. The text is vectorized into 128 dimensions. Padding and Truncation is performed on each token.

4.4.4 BiLSTM: GlOve Embeddings (Glove Embeddings)

GlOve is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Pre-trained GlOve embeddings of 300 dimensions are used in which embedding layer converts input tokens into dense vectors.

4.5 Model Architecture

4.5.1 Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised learning model used commonly for classifications. In this project, SVM in optimized using Optuna for both TF-IDF and Word2Vec datasets. The svm model‘s hyper-parameters are regularization parameter "C" (0.1 to10), kernel type (linear, rbf, or poly), and polynomial degree (2 to 4), which are all optimized to maximize the F1-score. The model uses cache_size=10240 (10GB) and max_iter =2000 for efficient and reproducible training. StandarScalar is applied to dense Word2Vec data for consistency.

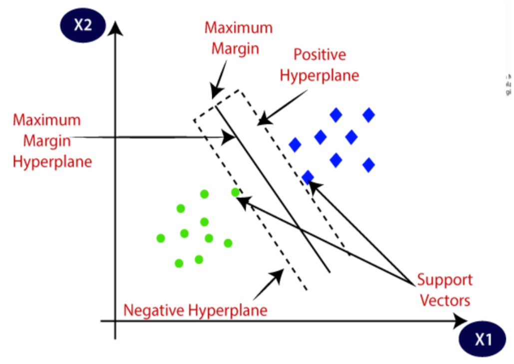


Figure 2: Support Vector Machine

4.5.2 Random Forest

Random Forest (RF) is a robust ensemble learning method that is used for classification tasks, which combines multiple decision trees to improve predictive accuracy and control over-fitting. In this project, like SVM, RF is also optimized using Optuna to maximize the F1-score for both datasets. Hyper-parameters such as, number of trees: n_estimators= (100 to 300) with step of 10, maximum depth of tree: max_depth= (None,10,20,30,40,50), the minimum samples required to split an internal node: min_sample_split= (2 to 10) are tuned for better model. The model is trained using n_jobs=-1, which uses all the resources of computer for parallel processing.

To avoid over-fitting for both SVM, and RF, the training set is split into training and validation subsets, with Optuna tuning based on the validation set while leaving the test set for final evaluation. 5-Cross-validation is used during the optimization to ensure robust hyper-parameter selection across the multiple splits of the data.After 50 trials, the best configurations for the algorithms are used to train on both vectorized data (TF-IDF, Word2Vec).

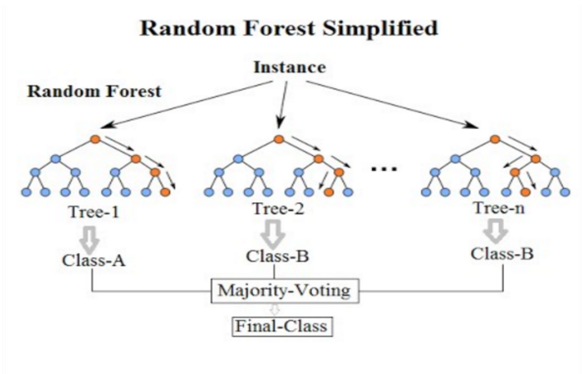


Figure 3: Random Forest

4.5.3 ALBERT (Lan et al., 2019)

ALBERT is a transformers model pre-trained on a large corpus of English data in a self-supervised fashion, the name of the model is "albert-base-v2". It has repeating layers which have same weights, it has small memory footprint but, has same computational cost as it has BERT like architecture. It has 12 repeating layers, 12 embedding dimension, 768 hidden dimension, 12 attention heads, and 11M parameters.This model is the fine tuned for spam classification task. For this project, the "Albert-Tokenizer" inputs maximum length of 128 tokens. Batching is done with batch_size=20 as the GPU cannot compute more. The model is trained for 6 epochs, optimized by "AdamW" with a learning rate of "5e-5" and weight decay of "1e-5" to prevent over-fitting. Gradient clipping (max 1.0) ensures stables updates, while StepLRscheduler reduces lr by 90% for every 2 epochs. Validation occurs at the end of each epoch, computing accuracy. Training of the model leverages GPU acceleration, with memory cleared afterwards to handle model's size in GPU.

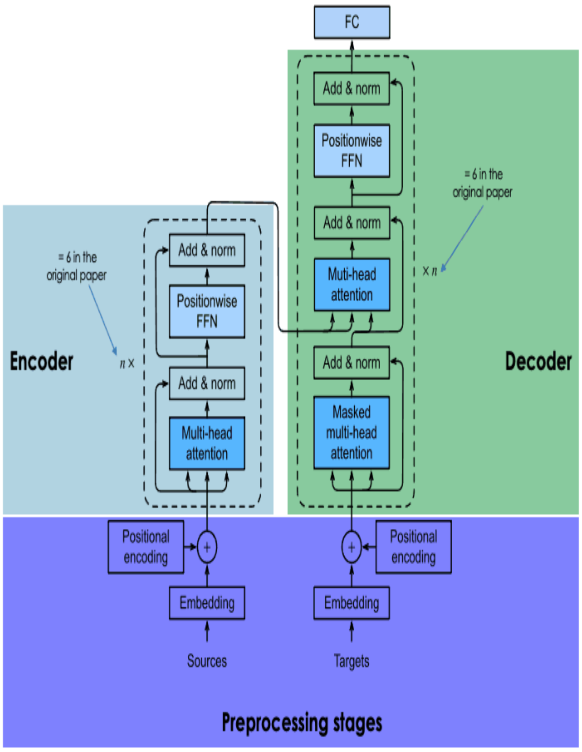


Figure 4: BERT

4.5.4 BiLSTM (Pytorch LSTM)

A Bidirectional LSTM(Long-Short-Term-Memory) is type of Recurring Neural Network know for their ability in capturing sequential data by processing the input data in both forward and reverse direction. This methods helps to capture patterns that might be missed when the data hidden layer only has the knowledge about past data by also passing the data from future. In this project, the data is sent to embedding layer to capture semantic relationships by leveraging GlOve embeddings (300 dimensions). The hyperparameters are number of layers stacked are: num_layers=2, hidden neurons in each layers are hidden_size=128, batch_size=20, learning_rate= 5e-5 and initial training epochs =10 with early stopping to prevent over-fitting. The AdamW optimizer enhanced by weight_decay= 5e-5 ensure effective parameter updates. Their is dropout layer with rate=0.5, and early stopping which halts if validation loss doesn't improve for three consecutive epochs to avoid over-fitting. Validation occurs at the end of each epoch with cross-entropy loss function for performance on a separate validation dataset. This model leverages GPU acceleration for a faster processing.

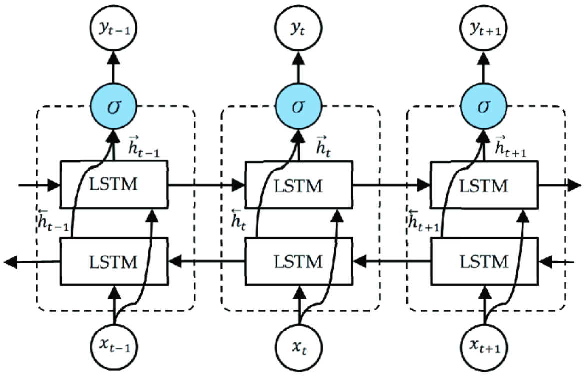


Figure 5: Bidirectional LSTM

4.5.5 Evaluation Metrics

The performance of the models are evaluated on the basis of different evaluation metrics like, **Accuracy**: it represents overall correctness of the mode; by showing percentage of correctly predicted instances. **Precision**: it tells the number of instances of spam or ham are correctly identified as spam or ham. **Recall**: it tells, how effectively the model captures actual spam or ham messages. **F1-Score**: it balances precision and recall, making it very important as there is a need to balance the avoiding false positive with true positive. **Confusion Matrix**: it gives more detailed view of the model with TP, FP, TN, and FN states. By which the detailed breakdown of the model can be possible.

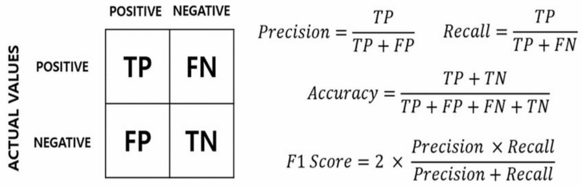


Figure 6: Evaluation Metrics

5 Experiments and Results

5.1 Experiments

As, the goal the project is to compare the algorithms, several experiments are conducted on the pre-processing steps, vectorization methods, and hyper-parameters.

5.1.1 Pre-Processing

The raw data is cleaned of null values and before the stop words removal, the pre-processing of data such as: **lower-casing, removal of numbers, removal of punctuations, and removal of empty space** in the text are performed. But, as it is spam detection, the punctuations, numbers, and empty space between the words have a lot of sematic meaning in determining a text spam or ham. So, the experiments are conducted twice such that all the algorithms are evaluated **with and without pre-processing** that are mentioned above.

5.1.2 Vectorization

In the project a total of four vectorization methods are used. To optimize the hyper parameters for TF-IDF, and Word2Vec, Optuna is used which gives the hyper parameters that performs best on these vectorization methods. For training the algorithms, the data that is fed is vectorized using these parameters. As ALBERT has its own text embeddings, and expects a certain dimension of vector. So, no experiments are conducted here. For BiLSTM, the embedding layer takes the vector from GlOve word embeddings, the experiments are conducted with 100d, 200d, and 300d (d = dimensions) of GlOve embeddings. Finally, 300d is giving the best performance of the model.

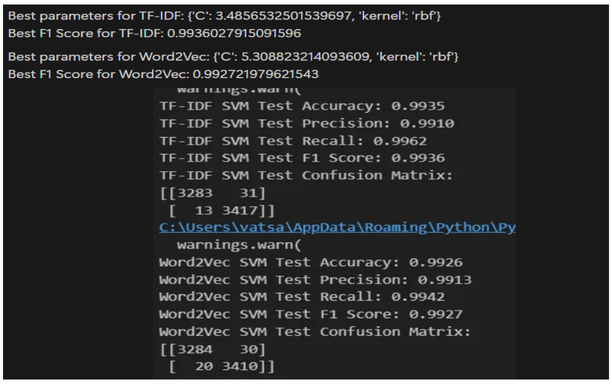
5.1.3 Hyper-Parameters

As, discussed in section 4.5, all the hyper-parameters of the ML models are optimized by using Optuna. As, deep learning models takes enormous time and computation to optimize model with Optuna, the parameters like learning rate, weight decay, optimizer, number of layers, batch size, and epochs are optimized manually by changing the values after each training cycle based on the evaluation of the models.

5.2 Results

5.2.1 Without Pre-Processing

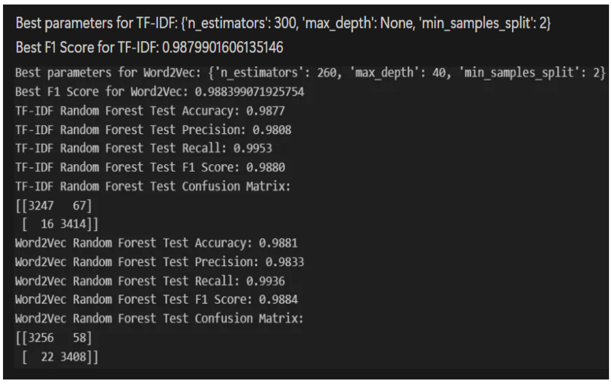
1. Support Vector Machine (SVM)



```
Best parameters for TF-IDF: {'C': 3.4856532501539697, 'kernel': 'rbf'}
Best F1 Score for TF-IDF: 0.9936027915091596
Best parameters for Word2Vec: {'C': 5.308823214093609, 'kernel': 'rbf'}
Best F1 Score for Word2Vec: 0.992721979621543
TF-IDF SVM Test Accuracy: 0.9935
TF-IDF SVM Test Precision: 0.9910
TF-IDF SVM Test Recall: 0.9962
TF-IDF SVM Test F1 Score: 0.9936
TF-IDF SVM Test Confusion Matrix:
[[3283   31]
 [  13 3417]]
C:\Users\vatsa\AppData\Roaming\Python\Py
  warnings.warn(
Word2Vec SVM Test Accuracy: 0.9926
Word2Vec SVM Test Precision: 0.9913
Word2Vec SVM Test Recall: 0.9942
Word2Vec SVM Test F1 Score: 0.9927
Word2Vec SVM Test Confusion Matrix:
[[3284   30]
 [  20 3410]]
```

Figure 7: SVM-Without Pre-processing

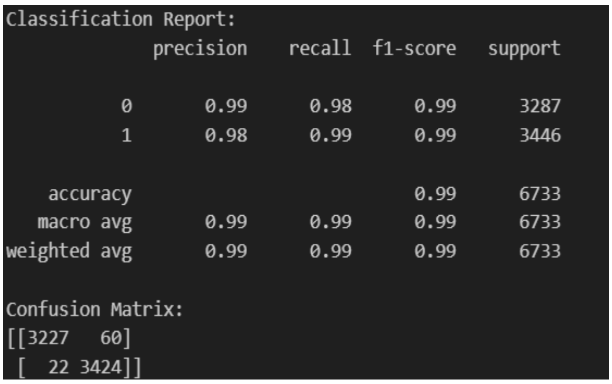
2. Random Forest (RF)



```
Best parameters for TF-IDF: {'n_estimators': 300, 'max_depth': None, 'min_samples_split': 2}
Best F1 Score for TF-IDF: 0.9879901606135146
Best parameters for Word2Vec: {'n_estimators': 260, 'max_depth': 40, 'min_samples_split': 2}
Best F1 Score for Word2Vec: 0.988399071925754
TF-IDF Random Forest Test Accuracy: 0.9877
TF-IDF Random Forest Test Precision: 0.9808
TF-IDF Random Forest Test Recall: 0.9953
TF-IDF Random Forest Test F1 Score: 0.9880
TF-IDF Random Forest Test Confusion Matrix:
[[3247   67]
 [  16 3414]]
Word2Vec Random Forest Test Accuracy: 0.9881
Word2Vec Random Forest Test Precision: 0.9833
Word2Vec Random Forest Test Recall: 0.9936
Word2Vec Random Forest Test F1 Score: 0.9884
Word2Vec Random Forest Test Confusion Matrix:
[[3256   58]
 [  22 3408]]
```

Figure 8: RF-Without Pre-processing

3. ALBERT



```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      3287
           1       0.98      0.99      0.99      3446

    accuracy                           0.99      6733
   macro avg       0.99      0.99      0.99      6733
weighted avg       0.99      0.99      0.99      6733

Confusion Matrix:
[[3227   60]
 [  22 3424]]
```

Figure 9: ALBERT-Without Pre-processing

4. BiLSTM

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      2510
           1       0.98      1.00      0.99      2540

    accuracy                           0.99      5050
   macro avg       0.99      0.99      0.99      5050
weighted avg       0.99      0.99      0.99      5050

[[2466   44]
 [   6 2534]]
```

Figure 10: BiLSTM -Without Pre-processing

5. Comparison of all algorithms

Algorithms/ Evaluation Metrics	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
SVM (TF-IDF)	99.35	99.10	99.62	99.36	[3283 31 13 3417]
SVM (WORD2VEC)	99.26	99.13	99.42	99.27	[3284 30 20 3410]
Random Forest (TF-IDF)	98.77	98.08	99.53	98.80	[3247 67 16 3414]
Random Forest (WORD2VEC)	98.81	98.33	99.36	98.84	[3284 30 20 3410]
AlBERT	98.78	98.28	99.6	98.82	[3227 60 22 3424]
BiLSTM	99.00	98.29	99.76	99.02	[2466 44 06 2534]

Figure 11: All algorithms -Without Pre-processing

5.2.2 With Pre-Processing

1. Support Vector Machine (SVM)

```
Best parameters for TF-IDF: {'C': 1.5432484576911387, 'kernel': 'rbf'}
Best F1 Score for TF-IDF: 0.9925839755707431

Best parameters for Word2Vec: {'C': 1.6393469589175644, 'kernel': 'rbf'}
Best F1 Score for Word2Vec: 0.991723537098882

TF-IDF SVM Test Accuracy: 0.9924
TF-IDF SVM Test Precision: 0.9901
TF-IDF SVM Test Recall: 0.9950
TF-IDF SVM Test F1 Score: 0.9926
TF-IDF SVM Test Confusion Matrix:
[[3280   34]
 [  17 3413]]
C:\Users\vatsa\AppData\Roaming\Python\Pyth
  warnings.warn(
Word2Vec SVM Test Accuracy: 0.9915
Word2Vec SVM Test Precision: 0.9879
Word2Vec SVM Test Recall: 0.9956
Word2Vec SVM Test F1 Score: 0.9917
Word2Vec SVM Test Confusion Matrix:
[[3272   42]
 [  15 3415]]
```

Figure 12: SVM-With Pre-processing

2. Random Forest (RF)

```
Best parameters for TF-IDF: {'n_estimators': 270, 'max_depth': None, 'min_samples_split': 4}
Best F1 Score for TF-IDF: 0.9874077290490665

Best parameters for Word2Vec: {'n_estimators': 130, 'max_depth': 40, 'min_samples_split': 2}
Best F1 Score for Word2Vec: 0.9882455376578145
TF-IDF Random Forest Test Accuracy: 0.9871
TF-IDF Random Forest Test Precision: 0.9805
TF-IDF Random Forest Test Recall: 0.9945
TF-IDF Random Forest Test F1 Score: 0.9874
TF-IDF Random Forest Test Confusion Matrix:
[[3246   68]
 [  19 3411]]
Word2Vec Random Forest Test Accuracy: 0.9880
Word2Vec Random Forest Test Precision: 0.9838
Word2Vec Random Forest Test Recall: 0.9927
Word2Vec Random Forest Test F1 Score: 0.9882
Word2Vec Random Forest Test Confusion Matrix:
[[3258   56]
 [  25 3405]]
```

Figure 13: RF-With Pre-processing

3. ALBERT

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3332
           1       0.99      0.99      0.99      3399

    accuracy                           0.99      6731
   macro avg       0.99      0.99      0.99      6731
weighted avg       0.99      0.99      0.99      6731

Confusion Matrix:
[[3289   43]
 [  34 3365]]
```

Figure 14: ALBERT-With Pre-processing

4. BiLSTM

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      2522
           1       0.98      1.00      0.99      2527

    accuracy                           0.99      5049
   macro avg       0.99      0.99      0.99      5049
weighted avg       0.99      0.99      0.99      5049

[[2481   41]
 [   5 2522]]
```

Figure 15: BiLSTM -With Pre-processing

5. Comparison of all algorithms

Algorithms/ Evaluation Metrics	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
SVM (TF-IDF)	99.24	99.01	99.50	99.26	[3280 34 17 3413]
SVM (WORD2VEC)	99.15	98.79	99.56	99.17	[3272 42 15 3415]
Random Forest (TF-IDF)	98.71	98.05	99.45	98.74	[3246 68 19 3411]
Random Forest (WORD2VEC)	98.80	98.38	99.27	98.82	[3258 56 25 3405]
AlBERT	99.00	98.71	99.00	98.84	[3289 43 34 3365]
BiLSTM	99.09	98.40	99.80	99.06	[2481 41 05 2522]

Figure 16: All algorithms -With Pre-processing

5.3 Comparison of Models

In this section, I compare the performance of each model using the evaluation metrics, analyzing the strengths, and identifying best performing combinations.

5.3.1 Overall Model Performance

1. SVM (TF-IDF, Word2Vec): It shows the highest accuracy, and the F1-Score across both pre-processed and non-pre-processed dataset. The precision and recall remained high, demonstrating a robust performance. As, I tried to optimize the hyper-parameters, it takes a huge amount of time to train on and it cannot use all the resources in the computer as it cannot be defined in the implementation.
2. BiLSTM: It performed strongly as well, especially with pre-processing, achieving an F1-Score of 99.06% which indicates the capability to capture sequential information. However, as it is deep learning model with 2-layers stacked, even with GPU acceleration the training time is very heavy.
3. ALBERT: It achieved a commendable results across both the settings, performing close to Random Forest, but its computational requirements are very high, even-though I am using a vanilla model. If, I use a more complex model with more encoding and decoding layers, it may perform better than this setting. The algorithm works better with the preprocessed data than the raw data. The training time is highest among all the algorithms used.
4. Random Forest (TF-IDF, Word2Vec): It demonstrated slightly lower metrics than the other algorithms, but it was very efficient in finding the hyper-parameters for training the algorithm, As, it has implementation to use all the resources of computer. It also, offers balance between performance and computational efficiency as it has least amount of training time in all the algorithms.

5.3.2 Best Model and Feature Extraction Combination

1. Top Performance: The SVM with TF-IDF combination yielded the best results in terms of accuracy(99.35) and F1-Score (99.36) without pre-processing. This indicates that SVM can leverage TF-IDF effectively for spam detection.
2. Sequential Data Model: The BiLSTM with pre-processing followed closely, excelling in recall(99.80), which is essential for spam detection tasks where minimizing False-Negatives is critical. Here, unlike ALBERT, BiLSTM captures context over the sequence using the LSTM units memory cells to retain the information across time steps. ALBERT perform extremely well on complex relationships across entire sequence, this dataset doesn't have enough complexity to leverage ALBERT. It would be reason for little decrease in performance.

5.3.3 Trade-Off Observations

1. SVM model trained faster and required fewer resources, ideal for the rapid iterations, with TF-IDF showing an edge over Word2Vec.
2. BiLSTM and ALBERT, while yielding competitive results, they demand a huge amount of computational power and slower to train due to sequential nature and stacked LSTM, and transformer-based approach of ALBERT. This trade-off makes then suitable for higher recall over training time and complex patterns in data.
3. Random Forest offered a good compromise of, balancing the training time and moderate consumption with a high accuracy. With more complex data, Random Forest may have potential for better performance.

5.3.4 Impact of Pre-Processing

1. Pre-processing improved BiLSTM recall and overall consistency for SVM and RF. In contrast, ALBERT saw a minimal increase of metrics, indicating the it can effectively handles unprocessed data.
2. Models with pre-processing showed a slight improvement in reducing False Positives, demonstrating that pre-processing can enhance model robustness, particularly for BiLSTM.

5.4 Note-Worthy Findings

Pre-processing steps improved model performance, especially increasing the BiLSTM model's recall, indicating its sensitivity to input quality. The high accuracy of SVM with TF-IDF suggests its effectiveness in spam classification, aligning with prior research. ALBERT performed competitively but outperformed by SVM. RF struggled with Word2Vec, likely due to sparse features affecting its performance. As, the dataset is of 36,000 rows, it is on the edge for better performance using ML or DL algorithms for better results. Also, the data doesn't have complex structures or features to leverage DL algorithms. With more computational power, RF and DL algorithms can train well by increasing layers, batch size, neurons, learning rate optimization, and many more.

6 Discussion and Conclusion

This project aimed to find the best spam classification accuracy by evaluating the performance of SVM, RF, ALBERT, and BiLSTM across TF-IDF and Word2Vec feature extraction methods on ML algorithms. SVM with TF-IDF with pre-processing emerged as best-performing combination vs Word2Vec. This suggests that model and feature pairing significantly influence the performance. Pre-processing the data improved the performance significantly of all the algorithms, but not much evidently in ALBERT which can be inferred as ALBERT is immune to pre-processing until certain extent. Dataset limitations include potential noise, these can be mitigated with more refined pre-processing. Ethical considerations in deploying spam classification include potential bias and importance of responsible data usage to avoid increasing harmful stereotypes. Future approach to ethical challenged should involve model monitoring, periodic updates, and transparent model decision process to ensure fairness and reliability in the system. In conclusion, this project underscore the significance of choosing best model and feature combination for effective spam classification. Future work might integrate advanced deep learning architectures or ensemble methods to further boost the performance and robustness of automated spam detection systems.

References

ALBERT Model Documentation. Albert model documentation. https://huggingface.co/docs/transformers/en/model_doc/albert#transformers.AlbertForSequenceClassification.

BILSTM. Lstm. https://medium.com/@abhinav.mishra123/bidirectional-lstm-for-mnist-classification-using-pytorch

Argha Ghosh and A Senthilrajan. 2023. Comparison of machine learning techniques for spam detection. *Multimedia Tools and Applications*, 82(19):29227–29254.

Glove Embeddings. Glove embeddings. https://nlp.stanford.edu/projects/glove/.

Santiago González-Carvajal and Eduardo C Garrido-Merchán. 2020. Comparing bert against traditional machine learning text classification. *arXiv preprint arXiv:2005.13012*.

Hugging Face Dataset. Spam dataset. https://huggingface.co/datasets/SetFit/enron_spam.

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

Pytorch LSTM. Pytorch lstm. https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html.

Azeema Sadia, Fatima Bashir, Reema Qaiser Khan, and A Khalid. 2023. Comparison of machine learning algorithms for spam detection. *Journal of Advances in Information Technology*, 14(2):178–184.

Thaer Sahmoud and Dr Mohammad Mikki. 2022. Spam detection using bert. *arXiv preprint arXiv:2206.02443*.

Phani Teja Nallamothu and Mohd Shais Khan. 2023. Machine learning for spam detection. *Asian Journal of Advances in Research*, 6(1):167–179.

Qussai Yaseen et al. 2021. Spam email detection using deep learning techniques. *Procedia Computer Science*, 184:853–858.