Natural Language Analysis Report

1. Feature Extraction

1.1 TF-IDF

TF-IDF measures the importance of a term within a document relative to its frequency across all documents in the corpus. The Term Frequency (TF) computes the frequency of a word in the document. The more times a term appears, the more relevant it is to that document, while the Inverse Document Frequency (IDF) diminishes the weight of terms that occur very frequently across documents and increases the weight of terms that occur rarely (Ibanez, 2023) [1]. A higher TF-IDF score indicates that the term is more relevant for that document within the corpus.

1.2 Word2Vec

Word2Vec is a technique that learns word embeddings, representing words as dense vectors in a high-dimensional space. Processing a large text corpus generates a vector space where each word is assigned a vector. Using CBOW or Skip-gram models to predict words from context or vice versa, Word2Vec adjusts vectors so that semantically similar words cluster together in this space.

1.3 Advantages and Disadvantages

A. Advantages of TF-IDF

- Easy to implement and understand: It is a straightforward method that can be computed quickly, making it suitable for large document collections [3].
- Effective for identifying relevant terms.
- Works well with sparse datasets and traditional machine learning models.

B. Disadvantages of TF-IDF

- Ignores the context and order of words, leading to potential loss of semantic information [3].
- High-dimensional output can be computationally expensive for large vocabularies.

C. Advantages of Word2Vec

- Captures semantic and syntactic relationships: Word2Vec allows words with similar meanings or usage patterns to have similar vector representations [4].

- Efficient training and Improved performance: Word2Vec offers dense, low-dimensional, and semantically rich vector representations, compared to TF-IDF, making computations more efficient.

D. Disadvantages of Word2Vec

- Requires a large amount of text data to train well.
- Lack of context awareness: Word2Vec treats each word as a single vector, failing to capture the different meanings of a word in different contexts (polysemy) [5].
- Inability to handle out-of-vocabulary words: Word2Vec cannot generate embeddings for words that are not present in the training corpus, limiting its applicability to new or rare words [5].
- Static representations: Word2Vec embeddings are static and cannot capture changes in word meanings over time or across different domains [5].

2. SVM Training Results

We chose an SVM classifier to train both sets because SVM can effectively handle sparse data and high-dimensional input. Besides, SVM uses kernel functions to map the input data into a higher-dimensional space, where the data may become more linearly separable. This kernel trick allows SVM to capture non-linear relationships in the data, which is beneficial when working with dense embeddings like Word2Vec [6]. However, despite SVM's adeptness at handling high-dimensional data, the input data must be provided in a two-dimensional array format, where each row corresponds to an individual sample and each column represents a feature. Therefore, we averaged these vectors to form a single vector per document.

TF-IDF

	precision	recall	f1-score	support
ham	0.98	1.00	0.99	954
spam	0.97	0.88	0.93	161
accuracy			0.98	1115
macro avg	0.98	0.94	0.96	1115
weighted avg	0.98	0.98	0.98	1115

Word2Vec

	precision	recall	f1-score	support
ham	0.96	0.97	0.97	954
spam	0.82	0.77	0.79	161
accuracy			0.94	1115
macro avg	0.89	0.87	0.88	1115
weighted avg	0.94	0.94	0.94	1115

Figure 1 – classification report of the testing set

Based on Figure 1, the model using TF-IDF features outperformed the one using Word2Vec in terms of accuracy, which is 0.98 when considering class imbalance. For the 'ham' category, the recall is 1.00 with TF-IDF, indicating that genuine 'ham' messages were correctly identified. However, the recall for 'spam' is slightly lower at 0.88, which suggests that some 'spam' messages were missed. In contrast, with

Word2Vec features, the recall for 'ham' is slightly reduced to 0.97, and more notably, the recall for 'spam' drops to 0.77, indicating that a larger number of 'spam' messages were incorrectly classified. Consequently, the model trained on TF-IDF features demonstrates superior performance over the Word2Vec-based model, particularly in terms of precision, recall, and F1-scores for 'spam'.

3. LSTM Model

We chose LSTM networks, which are particularly effective for sentiment analysis due to their ability to process and retain information over long sequences. It uses a series of gates (forget, input, and output) to selectively retain and discard information through its cell state, ensuring that only relevant sentiments affecting the overall polarity of the text are considered.

Firstly, since TF-IDF is a sparse matrix, we convert it into a dense array to better suit deep learning. However, TF-IDF features are not typical sequential data, so we expand the feature matrix into three-dimensional data (nb_sequence, nb_timestep, nb_feature) to meet the input requirements of the LSTM.

To ensure fairness in comparing the effects of TF-IDF and Word2Vec within LSTM models, we will use the same model architecture. The model is structured into the following layers:

- Input layer: the input to the model is a vector of dimension (1, 2676), which means that each sample is a sequence of length 1 and each sequence element is a 2676-dimensional TF-IDF vector.
- Bidirectional LSTM layer: a bidirectional LSTM with 64 cells is used to process the sequence data. The bi-directional LSTM can learn data dependencies from both directions, which is beneficial for textual data as it captures contextual information from both the preceding and the following text.
- Dropout layer: The dropout ratio is commonly set between 0.2 to 0.5 to ensure sufficient regularisation to mitigate the risk of overfitting.
- Fully Connected Layer: a fully connected layer containing 64 neurons and using the ReLU activation function is used to further process the information passed from the LSTM layer.
- Output Layer: a final output layer using a sigmoid activation function designed to compress the results between 0(ham) and 1(spam) for binary classification tasks.

Besides, the loss function, binary_crossentropy is useful for binary classification problems as it measures the discrepancy between the predicted probabilities and the actual binary labels. The optimization algorithm will choose Adam, because as go-to optimizer, Adam is better than other adaptive learning rate algorithms due to its faster convergence and robustness across problems [7].

After multiple tests, reducing batch size and increasing epochs showed minimal differences in accuracy and recall, thus the most efficient parameters, epochs=10 and batch size=64, were used for training.

4. SVM vs. LSTM Models

				W2V_LSTM	
	TF-IDF_SVM	W2V_SVM	TF-IDF_LSTM	Without Class Weights	With Class Weights
Test Set Accuracy	0.98	0.94	0.98	0.87	0.76
Training Time (Seconds)	188.59	197.23	55.52	149.70	207.50

Overall, the accuracy of the ham predictions of all four models is better than that of the spam predictions due to the imbalance of the data.

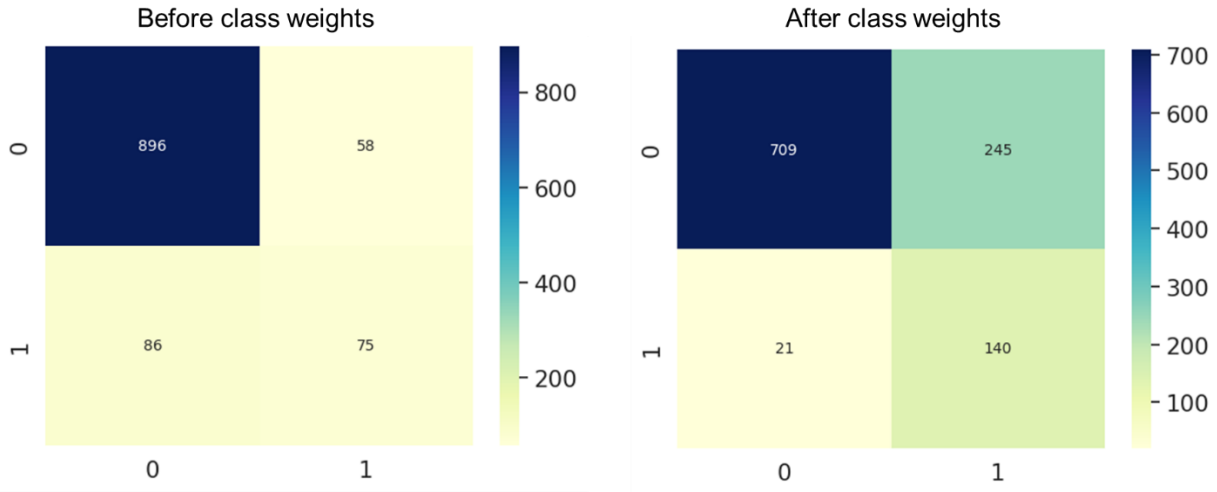

Figure 2 – The confusion matrix of W2V-LSTM using class weights or not

In the W2V_LSTM model, we attempted to compute class weights, allowing the model

to focus more on SPAM during training. This improved its accuracy in predicting spam, mitigating bias from data imbalance. Figure 2 shows that before class weighting, the model identified ham clearly but only a few spams correctly, with a recall of 0.47. After class weighting, the model became better at recognizing spam but also misclassified more hams as spam.

The difference in training time between the two vectorization methods is not significant for SVM, but it's substantial for LSTM. This is because TF-IDF features have lower dimensions, requiring fewer parameters for the LSTM model to learn. Additionally, TF-IDF preprocessing is simpler, and once computed, its values remain static throughout training iterations. In contrast, word2vec vectors typically require dynamic updates during model training, leading to increased computational costs [8].

5. Text Generation Model

   We use LSTM to generate a "Spam" message. First, we filter out all emails labelled as "spam" from the dataset, then use a Tokenizer to convert these texts into a series of numbers, each representing a specific word. Next, we standardize the length of all texts using pad_sequences, truncating texts that are too long and padding those that are too short to make their lengths uniform, facilitating model processing.

   The model architecture includes an embedding layer that transforms word numbers into fixed-length vectors. Two LSTM layers allow the model to learn long-term dependencies in the text data, with a Dropout layer following each to prevent overfitting. This is followed by two Dense layers, with the final layer using a softmax activation function to predict the probability of the next word.

   After training this model, we define a generate_text function to create new texts. This function continuously predicts the next most likely word and appends it to the seed text to generate new text sequences, repeating the process until the desired text length is achieved. Finally, by selecting random seed texts, we use this function to generate 100 samples that simulate the style and content of spam messages.

6. Assessing Performance with Generated Data

SVM Model Performance on Generated Texts:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	1.00	0.95	0.97	100
accuracy			0.95	100
macro avg	0.50	0.47	0.49	100
weighted avg	1.00	0.95	0.97	100

LSTM Model Performance on Generated Texts:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	100
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

Figure 3 – classification report of generated text

Figure 3 shows that when we fed the messages generated by our LSTM model to developed SVM and LSTM model, it achieves 95% and 100% classification accuracy. In particular, the SVM model has a recall of 0.95 for label 1, which means it correctly identifies 95% of all actual spam emails. However, it misses 5% of spam emails.

Therefore, we have the following speculation:

- Data Overlap: The generated texts closely mimic the spam in the model's training data, featuring specific recognizable patterns.
- High recognition power: Since the generated texts have similar features to the training data, the classification model can easily recognise them as spam, which explains why the accuracy of these generated texts reaches 100%.
- Lack of diversity: The output from the generative model is quite similar, often including phone numbers, which reflects its training data. This limits text diversity and complexity, resulting in homogeneous structure and vocabulary that are easy to categorize.

7. References

[1] Ibanez, D. (2023, March 16). *How to convert a text sequence to a vector | Baeldung on Computer Science*. Baeldung on Computer Science. https://www.baeldung.com/cs/text-sequence-to-vector

[2] Logunova, I. (2023, May 9). *Word2VEC: Why do we need word representations?*

Word2Vec: Explanation and Examples. https://serokell.io/blog/word2vec

[3] Jadon, A.K., & Kumar, S. (2023). A Comparative Study of CNNs and DNNs for Emotion Detection from text using TF-IDF. *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, 1329-1334.

[4] Zhu, J., & Ren, Z.J. (2023). The Evolution of Research in Resources, Conservation & Recycling Revealed by Word2vec-Enhanced Data Mining. *SSRN Electronic Journal*.

[5] Kale, A.S., Pandya, V., Di Troia, F., & Stamp, M. (2022). Malware classification with Word2Vec, HMM2Vec, BERT, and ELMo. *Journal of Computer Virology and Hacking Techniques, 19*, 1-16.

[6] Styawati, S., Nurkholis, A., Aldino, A.A., Samsugi, S., Suryati, E., & Cahyono, R.P. (2022). Sentiment Analysis on Online Transportation Reviews Using Word2Vec Text Embedding Model Feature Extraction and Support Vector Machine (SVM) Algorithm. *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, 163-167.

[7] Agarwal, R. (2023, September 13). Complete Guide to the Adam Optimization Algorithm. Builtin. https://builtin.com/machine learning/adam-optimization

[8] Zhou, H. (2022) 'Research of text classification based on TF-IDF and CNN-LSTM,' Journal of Physics. Conference Series, 2171(1), p. 012021. https://doi.org/10.1088/1742-6596/2171/1/012021.