SUMMATIVE ASSIGNMENT COVERSHEET

Module Name:	Natural Language Analysis (22/23)
Module code:	COMP42115
Z0 code:	Z0187631

Content:

1. Dataset Overview

Our dataset incorporates SMS messages classed as either "ham" (legitimate messages) or "spam" (unsolicited messages). Here is a quick snapshot of the data:

- Total Messages: Our dataset includes a total of 5,572 SMS messages.
- Classification:
    - Ham: A total of 4825 messages are labelled as ham. These are genuine messages that recipients anticipate to receive.
    - Spam: 747 messages are classified as spam. These are unsolicited messages that can vary from promotional content to phishing attempts.
- Message Length: The SMS messages in our dataset have an average length of roughly 80.5 characters.

2. Feature Extraction Techniques

2. a. TF-IDF:

TF-IDF is a numerical representation approach that reflects a word's importance to a particular document within a more extensive corpus. Let us break down its elements:

TF (Term Frequency): This quantifies the occurrence of a word in a specific document. For instance, if the word "cash" appears ten times in an SMS about a lottery and the total number of words in that SMS is 100, the TF for "cash" in that message is 0.1 or 10%. This could be valuable for identifying frequently occurring words within individual messages in our SMS dataset.

IDF (Inverse Document Frequency): This metric evaluates the importance of a word across the entire corpus. It is calculated by taking the logarithm of the total number of documents divided by the number of documents incorporating the word. In the context of our SMS dataset, if the word "cash" appears predominantly in spam messages but not in many ham messages, its IDF will increase, indicating its importance.

2. a. i. Advantages:

- **Local and Global Importance:** TF-IDF evaluates the frequency of a term in a single document (local importance) and how prevalent that term is across all documents (global importance). This dual nature makes it apt for our SMS dataset, as it can help determine between common spam terms and standard idiomatic words.
- **Downweighting Common Words:** In our SMS dataset, words like "and", "the", or "is" might appear frequently but offer little value in classifying a message as

spam or ham. TF-IDF innately downweights such words since they will likely appear across numerous messages.

2. a. ii. Disadvantages:

- **Independence Assumption:** TF-IDF assumes terms are independent of each other. This can be tricky in our dataset, where certain combinations of words might be more signifying spam.
- **Lacks Semantic Understanding**: TF-IDF does not capture the context or the semantic intention behind words. For example, the phrases "win money" and "earn cash" might have equivalent implications in our dataset, but TF-IDF would treat them differently.

2. b. Word2Vec:

Word2Vec is a cutting-edge Natural Language Processing (NLP) technique that uses neural networks to map words to a high-dimensional vector space. The brilliance of this approach resides in its capacity to encapsulate semantic meaning; words in texts with comparable contexts tend to be placed closer together in this vector space.

The model is trained using either the Continuous Bag of Words (CBOW) or the Skip-Gram architecture. In CBOW, the model predicts a target word (such as "apple") from its context terms (such as "red" and "fruit"). The Skip-Gram architecture, on the other hand, employs a target word to forecast its surrounding context, effectively reversing the CBOW technique.

2. b. i. Advantages:

The primary strength of Word2Vec is its adeptness at grasping semantic relationships between words. For instance, the vector representation of "king" minus "man" plus "woman" might result in a vector closing to "queen", showcasing its ability to comprehend relationships and metaphors. As a result of this semantic mapping, terms with synonymous or related meanings are compared in the vector space. This feature is invaluable in tasks like semantic search, document clustering, and even constructing chatbots that understand the context.

2. b. ii Disadvantages:

However, Word2Vec has its challenges. Training a Word2Vec model demands a tremendous corpus to be effective, making it computationally intensive. While pre-trained models are available, they often cater to general use cases. These models might fall short for specialized domains, such as medical or legal texts, as they might

require to be trained on domain-specific vocabularies. This necessitates additional training on domain-specific data, which might only sometimes be feasible.

3. Machine Learning Model

3. a. TFIDF

The SVM model showcased outstanding results in our SMS classification task. With a quick training time of just 0.009 seconds, the model attained an accuracy of 96%. Delving deeper, the precision for categorising legitimate messages (ham) was 0.95, and unsolicited messages (spam) was 0.97. This implies a high rate of accurately predicted labels for both categories. The recall, describing the model's ability to identify all relevant samples, was 0.97 for ham and barely lower at 0.95 for spam. The F1-score, a harmonic mean of precision and recall, stood invariably at 0.96 for both categories, suggesting a balanced performance. The cross-validation score of 0.92 further attests to the model's robustness.

3. b. Word2Vec

The decision tree model was trained in just 0.245 seconds, showcasing its efficiency in handling our SMS dataset. It attained an accuracy of 72%, signifying that it correctly classified 72% of the messages as either spam or ham. The precision and recall for 'ham' and 'spam' messages stood at 0.72. This suggests the model is equally proficient at recognising genuine messages and detecting spam. The F1-score, the harmonic mean of precision and recall, also tallied at 0.72 for both categories, demonstrating the model's balanced performance. However, a cross-validation score of 0.60 points to some possible overfitting, as there is a noticeable difference between this score and the accuracy.

4. Deep Learning Model with Word2Vec Features

4. a. Hyper-parameters for both LSTM and RNN model:

- Learning Rate: Default for the Adam optimizer.
- Batch Size: 32 - A moderate size balances computational efficiency and model update frequency.
- Number of Epochs: 50 - This allows ample iterations for the model to learn from the data.
- Optimizer: Adam - Popular for its adaptive learning rate.
- Loss Function: Binary Cross Entropy. It measures the difference between the predicted probabilities and the actual labels, making it a standard choice for such tasks.

4. b. Performance Insights:

Model	LSTM	RNN
Training Time	386.48 seconds	83.30 seconds
Test Loss	0.6933	0.6931
Test Accuracy	49.5%	50.5%

Both models show practically random-guessing accuracy on the test set, suggesting possible challenges like overfitting or the need for further hyperparameter tuning. The LSTM model, being more complex, took longer to train than the simpler RNN. Nevertheless, both models' performance was similar, indicating that complexity only sometimes guarantees better results.

5. Performance and Training Time Comparison of all models

The Decision Tree, a machine learning model, shows efficiency with a training time of only 0.245 seconds and 72% accuracy. The Deep Learning models, LSTM and RNN, on the other hand, took significantly longer to train, taking 386.48 and 83.30 seconds, respectively. However, both DL models showed unsatisfactory accuracy rates of around 50%, comparable to random guessing. The additional complexity of the LSTM did not result in better performance than the simpler RNN. Despite the rich representations provided by Word2Vec features and the prowess of deep learning architectures, the more straightforward Decision Tree outperformed this scenario regarding speed and accuracy.

6. Text Generation Model

A previously designed LSTM model, in section 4, is used to generate spam-like text. Beginning with a seed text, the model predicts the next word or token by considering the prior sequence. This procedure uses tokenization to convert text into a series of integers. The sequence undergoes padding to match the mandated input length. As words are anticipated, they are appended to the sequence, which the model uses for subsequent forecasts. This iterative process continues until an end-of-sequence token (e.g., a period) is predicted, or the maximum email length is attained.

7. Testing Using Generated Samples

Using the artificially generated samples, the performance metrics for the models are as follows:

ML Model	Accuracy	Precision	Recall	F1-score	Explanation
SVM	100%	1	1	1	This model performed exceptionally well with perfect scores across all metrics. It flawlessly classified the generated spam.
Decision Tree	88%	Ham – 0 Spam - 1	Ham – 0 Spam – 0.88	Ham -0 Spam – 0.94	Interestingly, it could not detect any genuine ('ham') messages. However, it demonstrated a strong capability to detect spam.

- **LSTM**: Despite its previous average performance, it impeccably classified the generated samples, with an accuracy of 1.0 and a loss of 0.6839.
- **RNN**: Contrarily, the RNN failed to correctly classify any generated samples, yielding an accuracy of 0.0 and a loss of 0.7089.

8. Discussion

The SVM and LSTM models' ideal scores indicate that the generated samples might differ distinctly from genuine messages, making them easily distinguishable. The Decision Tree's incapacity to recognize any genuine message indicates that the generated samples might lack specific attributes of real 'ham' messages. RNN's total misclassification raises troubles about its capability or the distinctiveness of generated samples.

The generated samples might include patterns or need more subtleties and nuances in genuine messages. This could make them easily identifiable, as recalled in the almost binary results from our models. Future work might involve refining the generation process to produce more authentic and challenging samples for classification.