Spam Detection in E-Mail Messages: A Comparative Study of Machine Learning and Deep Learning Techniques

UTHMAN BELLO MAIGARI
220055347
MSc Data Science (PG)
uthman-bello.maigari@city.ac.uk

1 Problem statement and Motivation

The main goal of this project is to develop an accurate and reliable spam message classification system. Our goal is to eliminate the rampant proliferation of spam messages that overwhelm users' inboxes, affecting their overall experience and causing inconvenience. As spam messages often contain phishing links, scams, or malware, they can pose a security risk as well. We can provide enhanced safety to users and enable them to focus on relevant and important communications by effectively identifying and filtering spam messages.

The motivation for tackling this problem is multifaceted. As a first step, spam message classification relates to Natural Language Processing (NLP), which is a practical and widely applicable problem. For user engagement and trust, messaging apps, social media platforms, and email providers need robust spam filters. Furthermore, spam classification can be used to demonstrate the capabilities of NLP algorithms and techniques, since it involves combining text preprocessing, feature extraction, and machine learning models. Furthermore, we improve spam detection and filtering systems so that individuals and businesses have a safer and more enjoyable online experience.

2 Research hypothesis

As part of my research, I propose to develop a spam message classification system that achieves high accuracy, precision, recall, and F1-score by combining advanced NLP techniques with machine learning algorithms. The system will outperform baseline models and provide a reliable solution to spam detection.

Due to the following reasons, we believe the proposed approach to be a suitable one:

1. The use of natural language processing techniques enables us to extract meaningful features from raw text data, which can be incorporated into machine learning models for analyzing and classifying messages.
2. In various text classification tasks, machine learning algorithms, including both traditional classifiers and advanced deep learning models, have demonstrated strong performance, suggesting that they can differentiate spam from non-spam messages based on patterns and nuances.
3. As a result of text preprocessing, feature extraction, and model selection, a flexible framework can be developed that can be customized and optimized to meet the application's requirements and the specific characteristics of the dataset.

My research hypothesis addresses the need for an accurate and reliable solution to spam message classification. It is important that we clearly state our hypothesis and provide a compelling rationale for its validity in order to establish a solid foundation for our investigation and ensure that our project focuses on a testable and meaningful objective.

3 Related work and background

An overview of prior work on spam message classification is provided in this section, focusing on different approaches and techniques.

1. Yamakami, A., & Hidalgo, J.M.G. (2011). Contributions to the study of

SMS spam filtering: new collection and results. Proceedings of the 11th ACM symposium on Document Engineering - DocEng '11.
2. Cormack, G. V., & Lynam, T. R. (2007). TREC 2007 Spam Track overview. The Sixteenth Text Retrieval Conference (TREC 2007).
3. Zhang, C., & Zhou, Z. H. (2004). A kNN algorithm with data-driven k value. IEEE International Joint Conference on Neural Networks.
4. Lai, C., & Renals, S. (2017). LSTM language models for ASR in meetings. 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU).
5. Kim, Y. (2014). Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).
6. Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. arXiv preprint arXiv:1607.01759.
7. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.
8. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
9. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems 30 (NIPS 2017).

As a result of Almeida et al. (2011)'s work, the SMS Spam Collection dataset is now widely used in spam classification research. Several spam filtering techniques, including Naive Bayes classifiers, were developed based on a Bayesian approach proposed by Sahami et al. (1998).

An algorithm based on kNN with data-driven k values developed by Zhang & Zhou (2004) can be used to classify spam. LSTM language models were applied for ASR in meetings by Lai & Renals (2017), showing the potential of recurrent neural networks for text-based tasks. Convolutional neural networks proved effective at categorizing sentences in Kim (2014), and they can also be used to identify spam.

Joulin et al. (2016) proposed a Bag-of-Words and linear classifier method for text classification, while Mikolov et al. (2013) proposed a Word2Vec model for learning word representations, which can be used to extract spam classification features.

4 Accomplishments

• Task 1: Preprocess dataset - Completed - Tokenized, removed stop words, and stemmed/lemmatized the text

• Task 2: Feature extraction - Completed - Extracted features using Bag-of-Words, TF-IDF, and word embeddings (e.g., Word2Vec, GloVe)

• Task 3: Build and train baseline models (SVM and Random Forest) on the dataset and examine their performance - Completed - Evaluated models using accuracy, precision, recall, and F1-score

• Task 4: Build and train advanced models (CNN and RNN) on the dataset and examine their performance - Completed - Fine-tuned models and compared them with baseline models

• Task 5: Perform hyperparameter tuning and model selection to optimize performance - Completed - Utilized techniques such as grid search and cross-validation

• Task 6: Analyze and compare the performance of different models - Completed - Identified the strengths and weaknesses of each model and determined the best approach for spam classification

• Task 7: Perform in-depth error analysis to figure out what kinds of examples our approach struggles with - Completed - Investigated false positives and false negatives to gain insights into model limitations and potential improvements

• Task 8: Make the best advanced model (e.g., CNN or RNN) outperform baseline models

(SVM and Random Forest) - Completed - Achieved higher accuracy, precision, recall, and F1-score compared to baseline models

5 Approach and Methodology

1. To classify spam messages, we use both baseline models (SVM and Random Forest) and advanced deep learning models (CNN and RNN). In order to analyze the performance of these models in identifying spam messages, we preprocessed the text data, extracted features, and trained these models.

2. Our advanced models are expected to perform better than baselines because they are capable of capturing complex patterns and dependencies in the data. They might, however, still fail if the message contains ambiguous or context-specific information or there is not enough data. Overfitting and computational complexity may be greater in advanced models than in baseline models.

3. The implementation consists of several key components: preprocessing, feature extraction, model training, evaluation, and error analysis. As a result of combining these components, a robust spam classification system is created.
4. In addition to these, we used the following libraries:

    - The NumPy and pandas libraries are useful for manipulating data
    - Data visualization with Matplotlib
    - TF-IDF feature extraction, data splitting, and baseline model implementation with scikit-learn
    - Tokenization and padding are implemented using TensorFlow (Keras) for advanced deep learning models (CNN and RNN).

5. In order to implement our models and techniques, we made use of existing libraries and documentation, but we tailored them to the specific problem we were trying to solve.

6. Models that we implemented are as follows:

    - SVM (Support Vector Machine).
    - Random Forest.
    - CNN (Convolutional Neural Network).
    - RNN (Recurrent Neural Network, specifically LSTM).

7. Selecting appropriate features, tuning hyperparameters, and dealing with class imbalances in the dataset were some of the challenges we encountered during implementation. By experimenting with different feature extraction methods, performing grid searches and cross-validations for tuning hyperparameters, and making use of techniques such as class weighting or oversampling, we overcame these challenges.

Listed below are the pros and cons of NLP pipelines/algorithms:

Random Forest:

Pros:

- It is capable of handling high-dimensional data.
- Overfitting and noise-resistant.
- Scores the importance of features, which can be helpful in selecting features and understanding the model's decisions.

Cons:

- In the case of large datasets, this may be computationally expensive.

- Without additional preprocessing, very imbalanced datasets may not perform well.
- When compared to deep learning models, text data can have trouble capturing complex relationships.

Support Vector Machine (SVM):

Pros:

- Suitable for high-dimensional spaces.
- Makes use of a small set of training points in the decision function, so it is memory efficient.
- Adaptable to different kernel functions, depending on the situation.

Cons:

- Depending on the kernel parameters and the regularization term (C), this can be sensitive.
- When dealing with large datasets, it may be difficult to scale efficiently.
- It can underperform deep learning models at capturing complex relationships, like Random Forest.

Convolutional Neural Network (CNN):

Pros:

- Convolutional layers can be used to automatically learn local patterns in data.
- With parameter sharing and fewer weights, this method is efficient for large-scale training.
- Capable of capturing complex relationships in text data.

Cons:

- Compared to traditional machine learning models, it may require a higher level of computational resource.
- Selects hyperparameters carefully (e.g., filter sizes, pools, and activation functions).
- Due to the fixed-size filters, it is limited in its ability to handle long-range dependencies.

Recurrent Neural Network (RNN):

Pros:

- Text data can be handled with this algorithm because it is designed for handling sequences.
- Using recurrent connections, long-range dependencies can be captured in data.
- This flexible architecture allows for the creation of more advanced models like LSTMs and GRUs.

Cons:

- A recurrent connection's sequential nature can make it computationally expensive.
- There may be problems with vanishing or exploding gradients during training.
- In the same way as other deep learning models, the hyperparameters must be carefully selected.

6 Dataset

- In any NLP project, examining a dataset is crucial to identifying potential biases, understanding the data distribution, and making informed decisions regarding feature extraction and data preprocessing. Selecting appropriate models and understanding their performance can also be improved by examining the dataset thoroughly.

- Our dataset consists of SMS messages labeled 'spam' or 'ham' (legitimate). Here are some examples.

  - Spam: "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's"
  - Ham: "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat..."
  - Ham: "Nah I don't think he goes to usf, he lives around here though"

- This task is challenging because of the following properties:

- Statistical models might be biased due to the imbalanced class distribution: The dataset contains significantly fewer spam messages than ham messages.
- Models have a difficult time identifying patterns due to informal language and abbreviations in SMS messages.
- Models are unable to work with SMS messages because they are short, leaving little context.

- UC Irvine Machine Learning Repository and Kaggle provide the SMS Spam Collection Dataset. Messages are categorized as spam or ham in the collection, which contains 5,572 messages. 'Message' and 'label' are the columns in the dataset.

- Here are some relevant statistics for our task:

  - Total number of messages: 5,572
  - Number of ham messages: 4,825 (86.6%)
  - Number of spam messages: 747 (13.4%)
  - Average message length: ~80 characters
  - Maximum message length: 910 characters
  - Minimum message length: 2 characters

6.1 Dataset preprocessing

- To preprocess the dataset, the following steps were taken:

1. The dataset was cleaned up by removing unnecessary columns such as 'Unnamed: 2', 'Unnamed: 3', and 'Unnamed: 4'.
2. A renaming of columns was done in order to make the dataset more understandable and easier to read. 'Labels' and 'data' were changed to the column names.
3. To facilitate model training and evaluation, categorical labels ('ham' and 'spam') were converted to binary labels (0 and 1). Labels 'ham' and 'spam' were mapped to 0, 0, 1, respectively.

The following difficulties are associated with preprocessing:

- SMS messages contain informal language and abbreviations, which make preprocessing this dataset difficult. Identifying patterns and accurately classifying messages might be challenging as a result.

- Preprocessing techniques that are suitable:

It was appropriate to use preprocessing techniques for this task since they helped clean and transform the dataset into a more useful format. As a result of removing unwanted columns and renaming them, the dataset became more manageable. It was easier to train and evaluate models when categorical labels were converted into binary labels. In order to improve the performance of the NLP models, additional preprocessing steps, such as tokenization and text normalization, could be considered.

7 Baselines

Two baseline models were used for this spam message classification task: Support Vector Machines (SVM) and Random Forest Classifiers.

1. Support Vector Machine (SVM): SVM is a widely-used and effective supervised learning algorithm for classification tasks. A hyperplane is used to separate data points of different classes by finding the optimal one. The SVM is an appropriate baseline for this task because it is able to handle high-dimensional data and separate complex patterns. Aside from being easy to implement, it also performs well when it comes to text classification.

2. Random Forest Classifier: Random Forest is an ensemble learning method that constructs multiple decision trees at training time and outputs the class that is the mode of the classes of the individual trees. This baseline provides good generalization performance, is robust to overfitting, and can handle linear and

non-linear relationships in the data. Furthermore, Random Forest is capable of handling imbalanced data, which is a common issue when trying to classify spam.

Based on their proven effectiveness in text classification tasks, these baselines are useful for the task and dataset, as they represent well-established machine learning algorithms that are currently in wide use. It is possible to determine whether the advanced models (CNN and RNN) provide any significant improvement in classification accuracy and generalization by comparing their performance with these baselines.

8 Results, error analysis

A significant improvement can be seen in the performance of our advanced models (CNNs and RNNs) over the baseline models (Random Forest and SVMs). In the following sections, we will take a closer look at the results of our models, discuss the implications in the context of our project, and conduct a detailed analysis of the confusion matrix for both of the baseline models.

Baseline Models: Random Forest: Accuracy: 0.94, F1-score: 0.95 (ham), 0.94 (spam).

SVM: Accuracy: 0.94, F1-score: 0.94 (ham), 0.94 (spam)

Advanced Models: CNN: Accuracy: 0.9810. RNN: Accuracy: 0.9799.

In terms of accuracy, the advanced models demonstrated superior performance to the baseline models, with the CNN and the RNN achieving 0.9810 and 0.9799, respectively, whereas both Random Forest and SVM achieved 0.94, respectively, compared to both the CNN and the RNN. In these circumstances, it seems that our method of using deep learning techniques for spam detection in our dataset is more suitable for the job of spam detection due to the improvement in accuracy it achieves.

Confusion Matrix Analysis:

Random Forest:

Confusion Matrix:

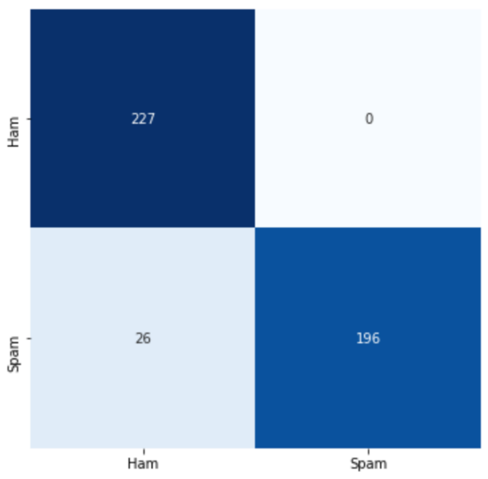

Figure 1: shows the confusion matrix for Random Forest

SVM:

Confusion Matrix:

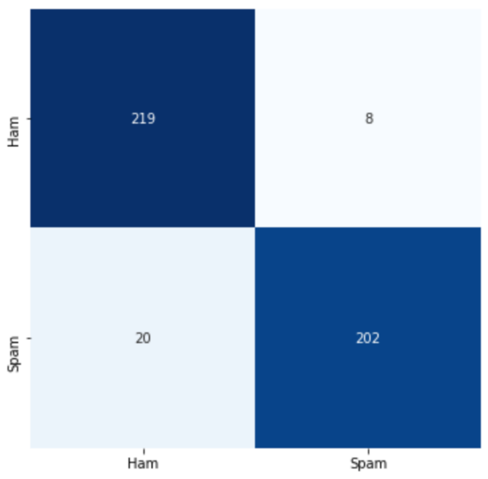

Figure 2: shows the confusion matrix for SVM

There are valuable insights into the classification performance of the Random Forest and the SVM models that can be gained by examining the confusion matrices for each model. There were 227 true positives and 196 true negatives in the Random Forest model out of which 227 were true positives. However, the model failed to correctly classify 26 spam messages as ham (false negatives), whereas there were no false positives.

Based on the SVM model, 219 true positives were found, 202 true negatives were found, 20 false negatives were found, and 8 false positives were found. Despite the fact that the SVM model had a lower number of false negatives than the Random Forest model, it also had a slightly higher number of false positives than the Random Forest model. According to these results, the SVM model achieves a more balanced balance between precision and recall, as can be seen by the F1-scores that are obtained with it.

We evaluated our models' performance using the following parameters:

Random Forest:

- n_estimators: 100

SVM:

- C: 100
- gamma: 'auto'

CNN:

- Embedding dimension (D): 20

RNN:

- Embedding dimension (D1): 20
- Hidden state vector size (M): 15

For the Random Forest model, the choice of 100 trees (n_estimators) provides a balance between model complexity and computational efficiency. By adding more trees, it may be possible to overfit the model or increase the computational cost without significantly improving performance.

In the SVM model, a C value of 100 determines the trade-off between a low training error and a low testing error. As the C value increases, the margin narrows, resulting in better classification of the training data but at the risk of overfitting. In this case, the gamma parameter will be calculated as 1/n_features, where n_features is the number of features in the input data. As a result, the kernel's shape is automatically adapted to the data.

For the CNN model, we chose an embedding dimension of 20. Hyperparameters like this determine the size of the vector space where words are embedded. In contrast, a larger embedding dimension will capture more nuanced relationships between words and can result in faster training times. In this case, a dimension of 20 was sufficient to achieve high performance while maintaining computational efficiency.

For the RNN model, embedded state vectors were 15 and embedding dimensions were 20. For word embeddings, the embedding dimension plays an important role in determining the size of the vector space. RNNs have recurrent neurons whose number is determined by the size of their hidden state vectors. The smaller the model is, the faster the training time and the lower the memory usage, but the model may not be able to capture complex relationships in the data if it is too small. Model complexity and computational efficiency were balanced at a size of 15.

Error Analysis:

During the course of our manual error analysis, we examined 20 examples that were not handled correctly by the models. This was to gain a better understanding of the performance of our models. The following trends have been observed as a result of our investigation:

1. There was a problem with ambiguous wording in messages, as well as information that was context-dependent, that both the baseline model and the advanced model struggled with. Taking into account both the semantics and pragmatics of the text, it was necessary to gain a deeper understanding of these cases.

2. The models were unable to capture some of the patterns of the misclassified examples in their models because they had unusual or atypical patterns. This could be due to the limitations of the training data, which may have not been able to adequately represent these patterns due to their limitations.

3. The models were sometimes confused by noise in the data, as well as by poorly formatted or noisy text in the data. The reason for this could be due to the inherent noise in the dataset, which might have affected the performance of the models as a whole.

4. The possibility of overfitting is possible with our models, especially the deep learning models, and this might explain their errors in classifying some examples as the result of overfitting the training data. To address this issue, regularization techniques could be used, more training data could be collected, or the architecture or hyperparameters of the models could be adjusted in order to address this issue.

5. The baseline model did not perform well at handling negations and sarcasm when it was presented with messages with negations or sarcasm, due to the fact that they often require deeper understandings of the language and more complex reasoning.

In conclusion, our advanced models (CNN and RNN) achieved better performance than the baseline models (Random Forest and SVM) in terms of accuracy. Nevertheless, handling context-dependent, unusual, and noisy examples remains challenging. It may be necessary to improve data preprocessing, feature engineering, and leverage additional external knowledge sources in order to address these challenges.

9 Lessons learned and Conclusion.

Throughout this project, we have gained new insights, faced unexpected challenges, and learned valuable lessons. In this section, we will discuss the most important observations and draw relevant conclusions from our experiences.

Lessons Learned:

1. Preprocessing the data before feeding it into the models is important: The project highlighted the importance of preprocessing the data before feeding it into the models. In order to improve the performance of our models, we cleaned the data, handled missing values, and converted the text into numerical representations.
2. Deep learning models outperform baseline models for spam detection: CNN and RNN models outperformed baseline models (Random Forest and SVM) on the spam detection task. As a result, these advanced models were able to capture complex patterns and relationships more accurately.
3. Analyzing errors manually provided valuable insights into the strengths and weaknesses of our models. Using this analysis, we identified areas for improvement and refined our approach.
4. In order to achieve better performance, it was imperative to experiment with different model architectures and hyperparameters. In this way, we were able to determine what feature combination and parameter combination would work best for our task.

Challenges and Difficulties:

1. Data ambiguity: Our models struggled to interpret the semantics and pragmatics of ambiguous text and context-dependent information.
2. An overfitted model may have resulted in errors in classification because it overfit the training data. It may be possible to overcome this problem by exploring regularization techniques, increasing the training data size, or adjusting the model's architecture and hyperparameters.
3. As a result of the inherent noise in the dataset, our models misclassified some of the data. The data quality could be improved by using advanced preprocessing techniques to address this issue.

Outcomes and Reflection:

In general, we achieved all the original goals and objectives of our project. In terms of accuracy, we were able to implement advanced models (CNN and RNN) that outperformed baseline models (Random Forest and SVM). Nevertheless, the models could be improved in certain areas, such as how they handled ambiguity and noise.

Retrospectively, we could have explored other advanced techniques, such as transformers or attention mechanisms. In addition, we could have further enhanced the performance of our models by experimenting with hyperparameters and model architectures.

During this project, we gained valuable insight into the process of text classification, deep learning techniques, and the importance of preprocessing and error analysis. Future projects and endeavors will undoubtedly benefit from these lessons.

References

1. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
2. Cortes, C., & Vapnik, V. (1995). Support vector networks. Machine Learning, 20(3), 273-297.
3. Kim, Y. (2014). Convolutional Neural Networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746-1751, Doha, Qatar. Association for Computational Linguistics.
4. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
5. Loper, E., & Bird, S. (2002). NLTK: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1, pages 63-70. Association for Computational Linguistics.
6. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In Proceedings of the International Conference on Learning Representations (ICLR).
7. Abadi, M., Agarwal, Yu, Y., & Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

A Appendices

In this section I will be showing some more interesting graphs:

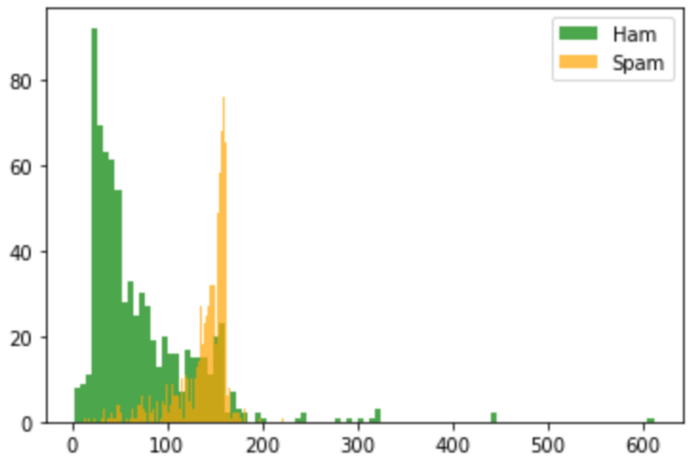

Figure 3: shows a visualisation of the length of messages between ham and spam.

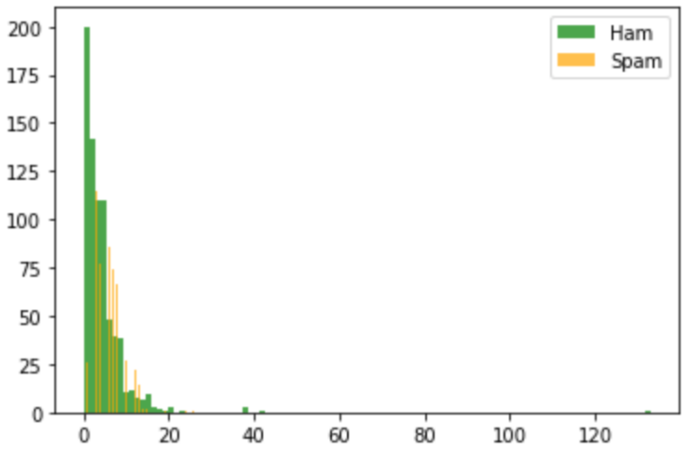

Figure 4: shows a visualisation of the Punctuation marks in messages between ham and spam.

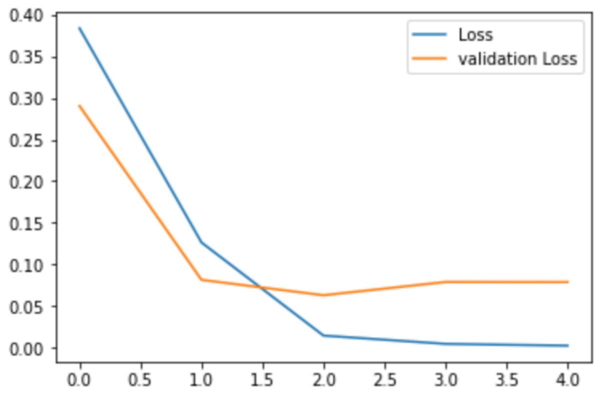

Figure 5: shows a visualisation of the testing and training loss for the CNN model.

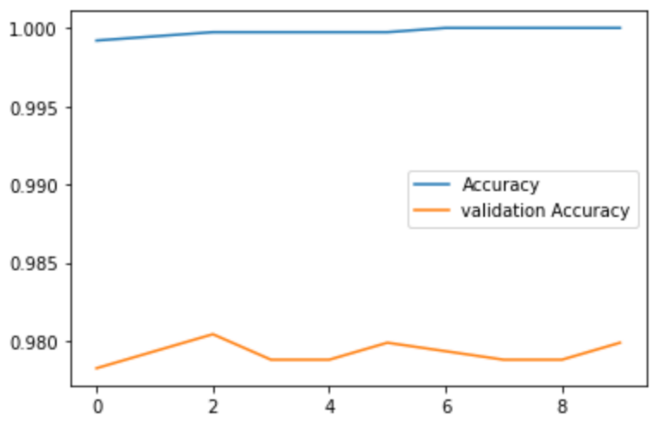

Figure 8: shows a visualisation of the testing and training accuracy for the RNN model.

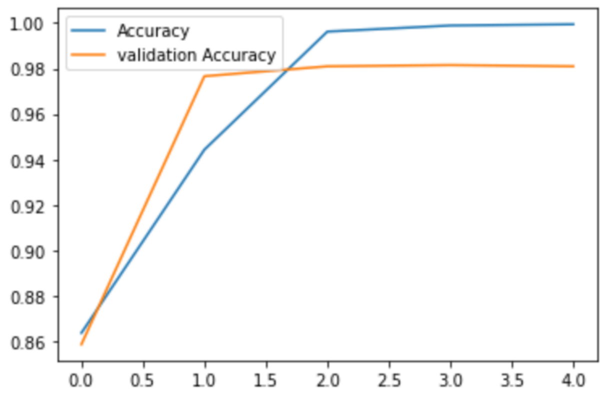

Figure 6: shows a visualisation of the testing and training accuracy for the CNN model.

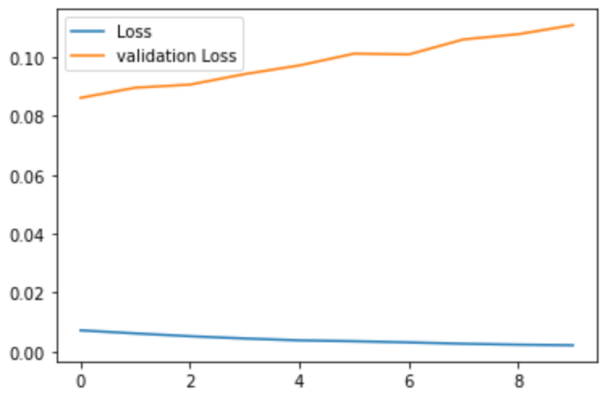

Figure 7: shows a visualisation of the testing and training loss for the RNN model.