NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

AY2023/24 SEMESTER 1

HG2051 Language and the Computer

Project 2 - Group 03

Tutor: Hiram Ring

Name	Matriculation Number
CHOO SHUEN MING	U1931232D
DANIEL KAINOVAN HANDOYO	U2230837E
LIW JUN LE	U2221922D

Spam Text Classification with Machine Learning

1. INTRODUCTION

Defined as unwanted messages that are often sent in bulk and are commercial or fraudulent in nature, spam has become a pervasive issue in the form of emails and SMS messages (Androutsopoulos et al., 2000). Improving spam filtering processes is therefore key to mitigate spam's impact on messaging platform users.

The project aims to train machine learning classifiers to differentiate spam from ham (non-spam) text data. The "SMSSpamCollection.txt" dataset, containing "HAM" and "SPAM" labeled data, is used to train the classifier model. Our evaluation is based on testing this model with 'unseen' email text data from the "email_corpus_lingspam.txt" corpus. Four metrics are used to assess the efficacy of our model: accuracy, precision, recall, and F1 score. Our model should be flexible enough to classify beyond SMS text, thus the test evaluates how well it adapts to the unique characteristics of email text data.

Our goal is to develop an accurate yet adaptable spam classifier. We employ Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction to evaluate the significance of words with regard to individual SMS messages and the whole corpus. Using this approach, we develop a text classification tool suited for different text genres.

2. METHODOLOGY

2.1 Dataset

The first dataset is the SMS Spam Collection, comprising 5574 real SMS messages from publicly available datasets. Sourced from the UC Irvine Machine Learning Repository (Almeida & Hidalgo, 2012), the messages have been manually extracted and labeled as either 'ham' or 'spam', representing a human-curated gold standard of data. This collection, acknowledged for its reliability, accuracy and comprehensiveness, serves as the training dataset for our classifier.

The second dataset is the Ling-Spam dataset (Androutsopoulos et al., 2000). This dataset consists of 2412 ham email messages from the spam-free moderated Linguist mailing list as well as 481 spam email messages. The Ling-Spam dataset offers a diverse range of email content, making it a valuable resource for the evaluation of our classifier.

2.2 Exploratory Data Analysis

Given the substantial size of the training dataset, we embarked on exploratory data analysis (EDA) to visualise its composition. This stage is crucial in guiding our data manipulation and model training techniques, especially when utilising Bayesian classifiers, where the overall distribution of ham/spam data influences the probabilistic nature of the machine learning model.

To illustrate the distribution of ham and spam messages within the dataset, we generated a count plot using the Seaborn library (Figure 1). The resulting count plot determined that the dataset was imbalanced since there were significantly more ham than spam messages. This implies that the model may not be able to accurately identify spam messages due to a lower sample size. If necessary, resampling can be conducted after reviewing the results.

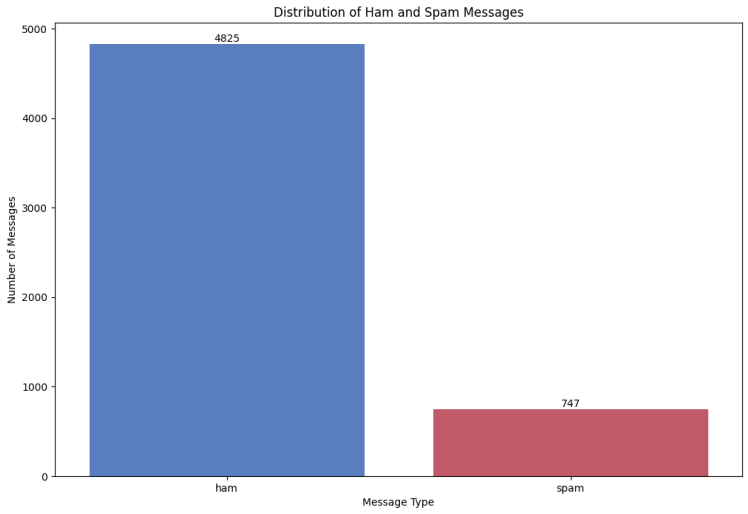

Figure 1. Distribution of Training Dataset

2.3 Data Preprocessing

Preprocessing is a crucial step in preparation for model training. The objective is to refine the raw SMS message data into a format suitable for machine learning.

We created a function for loading the dataset, incorporating exception handling to ensure execution. The ‘label’ column, originally containing ‘ham’ and ‘spam’ labels, is encoded into 0 for ‘ham’ and 1 for ‘spam’. Subsequently, the ‘preprocess_text’ function standardises the text, converting it to lowercase, tokenising it into individual words, and removing stopwords. Each remaining word is lemmatized to its base form, ensuring uniformity in the dataset.

2.4 Feature Extraction

The feature extraction process involves converting the text into numerical features suitable for machine learning models. Our approach involves leveraging TF-IDF, which evaluates the importance of words by considering their occurrences within a document (term frequency), and across the entire corpus (inverse document frequency) (Hassan & Mtetwa, 2018).

We defined the 'extract_features' function to operationalise TF-IDF. If no existing vectorizer is provided, the function initialises a 'TfidfVectorizer' from the scikit-learn library, which is fit to the training data, generating a feature matrix ('X'). It produces an array of ham/spam labels ('y') and returns the trained vectorizer for consistent feature extraction with new datasets.

2.5 Classification

The classification stage involves splitting the data into training and test sets, training the model, and evaluating its performance. The dataset was split into 80% for training and 20% for testing, with the 'stratify' parameter ensuring the distribution of spam and ham messages was maintained. We then trained the Multinomial Naive Bayes classifier, Support Vector Machine (SVM) classifier and the logistic regression classifier using the TF-IDF extracted features.

To evaluate the model, we employed stratified k-fold cross-validation with 10 folds. The dataset was divided into 10 subsets, maintaining the same distribution of labels as the original dataset. The trained model was subsequently evaluated using the Lingspam dataset, which provides insights into the model's adaptability to different linguistic contexts.

3. RESULTS

Four key metrics were used to evaluate the performance of our spam classifier, based on calculations regarding the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) in the model output.

1) Accuracy refers to the percentage of correctly classified messages for both ham and spam:

   (*TP + TN) / (TP+TN +FP+FN)*

   i.e. *correct predictions / all predictions made*

2) Precision refers to the percentage of correctly classified spam messages out of all spam-labeled messages, penalizing false positives:

   TP/(TP + FP),

   i.e. *true spam / (true spam + ham marked as spam).*

3) Recall, also known as the true positive rate, refers to percentage of spam caught, penalizing missed spam:

   TP/(TP+FN)
   i.e. *true spam / (true spam + missed spam)*

4) The F1 score combines precision and recall according to the formula:

   2 (precision*recall)/(precision+recall)*

   This takes into account the tradeoff between precision and recall, since precision prioritises avoiding false positives (mislabeling ham as spam), while recall prioritises catching more spam, even at the cost of more false positives. This is an especially relevant metric for our spam filtering use case. Dalse positives are arguably more costly, since it means missing out on a legitimate message, whereas the occasional uncaught spam can be manually flagged as spam with little cost to time or effort. Achieving a high F1 score is thus a primary goal, as it reflects a good balance between catching spam while not sending too much legitimate mail to the junk folder.

		Average Metrics	**Lingspam Metrics**
Naive Bayes	Accuracy	0.97	0.84
	Precision	0.98	0.64
	Recall	0.83	0.10
	F1	0.89	0.18
SVM	Accuracy	0.98	0.83
	Precision	0.98	0.47
	Recall	0.85	0.41
	F1	0.91	0.44
Logistic Regression	Accuracy	0.95	0.85
	Precision	0.97	0.98
	Recall	0.68	0.12
	F1	0.80	0.22

Figure 2. Naive Bayes, SVM and Logistic Regression using TF-IDF

As seen from the table above, the Naive Bayes, SVM, and Logistic Regression models all demonstrated strong accuracy scores of 97%, 98%, and 95% respectively on the SMS dataset (average metrics). These high accuracy scores underscore the classifiers' success in making correct overall predictions on the SMS dataset. While they have the same precision scores, SVM has a slightly higher recall and F1 score than the rest. This indicates that SVM is more effective in capturing all instances of spam, achieving a better balance between precision and recall.

In the evaluation on the Lingspam dataset, the classifiers produced mixed results, likely due to the different linguistic contexts and characteristics. While accuracy remained reasonably high at 84%, 83%, and 85% respectively, the recall was significantly lower for Naive Bayes (10%) and Logistic Regression (12%) compared to SVM (41%). This indicates that those two models faces challenges in identifying a significant portion of spam emails within the dataset. However, precision is higher for Naive Bayes (64%) and even higher for Logistic Regression (98%), suggesting that they are much better at accurately avoiding mislabeling of ham as spam.

As illustrated by the classifiers' varying performance between SMS and email datasets, models should be adapted to different linguistic contexts to prevent overfitting. Overall, the application of the TF-IDF feature extraction method contributed to exceptionally high accuracies, and further refinements can be made to enhance the models' adaptability across other text genres.

4. DISCUSSION

4.1 Methodological Choices

4.1.1 K-Fold Cross Validation

K-fold cross validation was used as a technique to assess the predictive performance of the trained models where the dataset was split into k-equal sized subsets. One split is reserved as validation data for testing models while the remaining k-1 splits are used as training data. In the case of our project this process is repeated 10 times with each of the 10 splits used exactly once as validation data. The results are then averaged to produce the metrics in the output file. We chose to use StratifiedKFold to maintain a consistent class distribution across each split and attempt to increase the model's ability to generalize to new data for more accurate assessment. Where the model's performance would not be overly dependent on a particular random split, ensuring more robust evaluation (Brownlee, 2023).

4.1.2 Train_Test_Split

The train_test_split method is used in main.py to split the sms data collection dataset into training and test sets with a test size of 20% and we ensured that the training and test sets have the same proportion of spam and ham messages as the original dataset. We did not include the metrics for the test split of our data as we used the LingSpam dataset as our Test Set. In our project, K-Fold Cross Validation is used for model evaluation and balancing variance in an

unbalanced dataset where the train_test_split method is used as a simple method to reliably segment training data for the different models.

4.1.3 Model Comparison

Multinomial Naive Bayes

This classifier is based on Bayes' theorem with the assumption of independence between every pair of features. The multinomial variant of this classifier means that this model is best suited for features that represent frequency counts which is common in text classification tasks. A great limitation of this model is its assumption of feature independence which is unrealistic in real-world data in the field of natural language processing.

Logistic Regression

This classifier estimates the probabilities using a logistic function to model a binary independent variable. This model is very efficient to train and is very fast at classifying unknown records. It is more robust than other linear models to the assumption of linear separability. However, its performance can be greatly influenced by the presence of irrelevant features and assume a linear relationship between dependent and independent variables which also may not be true for real-world data (Raj, 2021).

Support Vector Machine

The SVM works by finding a hyperplane that best divides a dataset into classes. It uses kernel functions to transform the input space into a higher-dimensional space where it is easier to separate the data linearly. It provides greater classification capabilities than the previous two models that comes at the cost of increased computational complexity and sensitivity to parameter tuning. To have an effective SVM model, iterative hyperparameter tuning is required and since the decision boundary is not immediately apparent in the original feature space it requires some sort of subject matter expertise in its approach (Gandhi, 2018).

4.2 Alternative Platforms for collaboration

Since it is a machine learning project working with large datasets, it may be preferable to work on Google Colab which provides free GPU for computationally intensive operations. A great appeal of Google Colab is the ability to run blocks of code iteratively which enables easy debugging and removes the need to retrain the model whenever the code is run without bringing additional libraries for model saving in VSCode. Additionally, we are also working on a large dataset and it would be great to conduct exploratory data analysis (EDA) before determining on the direction of the project and how to manipulate the data to ensure that it is used in the most efficient way possible. Google Colab enables EDA to be easily executed and the development process to be documented easily.

4.3 Features to Improve Accuracy

Data Preprocessing and Feature Engineering

In an attempt to improve accuracy on sms dataset we tested by implementing features into model training such as singlish, common spam words and email specific features. However, we realised that there was a fine line between overtraining and overgeneralizing the model. In this case, when we added Singlish features, the ling spam evaluation accuracy decreased significantly which signaled overtraining of the model. Eventually, we removed custom features and implemented bigrams and reduced max features to 1500 from 5000 through iterative running of code.

Model Training and Evaluation

The k-fold cross validation method was used for a fair and comprehensive evaluation of model performance as it mitigates the risk of overfitting and provides a reliable estimate of the model's ability to generalize to unseen data. We see a significant drop in lingspam metrics as compared to the trained model average metrics as the SMS and LingSpam dataset differ greatly in terms of style where SMS messages are informal, concise and use colloquial language, abbreviations, or slang. The LingSpam emails on the other hand are more formal and longer with more complex sentence structures. For example, from SMS dataset "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat…". The text is conversational, informal and short. This can be compared to the LingSpam dataset "Subject: 8th seals, kuala lumpur, programme universiti kebangsaan malaysia south east asian linguistics society eighth annual meeting…" We can see that the email is formal with a clear structure and follows a specific terminology, using academic and professional terms.

The stark difference in vocabulary and style between the two datasets challenge the model trained on SMS dataset. The model may learn features and patterns specific to the SMS texts but will not be as effective in identifying spam in academic or formal emails. Moving forward, to improve the model, we can implement models such as BERT or GPT and perform transfer learning as they are pre-trained on a diverse corpus and can understand the context of words in sentences.

4.4 Resampling for Imbalanced Training Dataset

As discovered through our exploratory data analysis, one drawback of our models is that the SMS Spam Collection used for training is an imbalanced dataset with a disproportionate proportion of ham than spam data. This could have impacted model performance, and accounted for the generally low F1 scores across the board with many false positives and false negatives. We thus tried the use of resampling to rectify this imbalance. Both oversampling of the minority spam data and undersampling of the majority ham data was carried out using the

Imbalanced-learn open source library, with the resulting performance metrics after resampling reflected in Fig. 3 below:

Lingspam Performance		**No Resampling**	**Oversampling**	**Undersampling**
Naive Bayes	Accuracy	0.84	0.57	0.62
	Precision	0.64	0.87	0.68
	Recall	0.10	0.15	0.46
	F1	0.18	0.25	0.55
SVM	Accuracy	0.83	0.72	0.73
	Precision	0.47	0.76	0.74
	Recall	0.41	0.66	**0.70**
	F1	0.44	0.70	**0.72**
Logistic Regression	Accuracy	**0.85**	0.75	0.74
	Precision	**0.98**	0.83	0.87
	Recall	0.12	0.63	0.57
	F1	0.22	**0.72**	0.68

Fig 3. Performance metrics on LingSpam data with no/over/under-(re)sampling. The highest value in the data of each metric has been bolded to aid interpretation of relative performance.

Compared to the existing performance with no resampling, both over and under sampling show success with greatly increased F1 scores, resulting from increases in model precision, recall, or both. This did however come with a slight hit to accuracy across the board of 10-20%. In one extreme case for instance, oversampling with the Naive Bayes classifier lowered accuracy to 0.57, just barely above chance, while precision however increased from 0.64 to 0.87. Recall remained low at 0.15. This suggested an overly cautious spam filter, that missed a lot of spam while trying to avoid false positives. Undersampling however greatly benefited the Naive Bayes classifier performance, increasing recall from 0.10 to 0.46 while maintaining similar precision. This could be understood as undersampling reduces the ham samples while maintaining the spam samples, increasing the baseline prior probability of spam in the Bayes model, and thus the overall filtering of spam. Recall similarly increased for SVM (0.41 to 0.66 and 0.70), and for Logistic Regression (0.12 to 0.63 and 0.57). In summary for resampling, the resulting models' performances are generally more balanced overall, with minor sacrifices in accuracy for greater F1 score. In the case of a spam classifier, this can be a major improvement. As mentioned earlier, false positives (ham marked as spam) can incur a greater cost than false negatives (missed spam),

meaning that accuracy is not the sole consideration as it does not consider either FPs or FNs. It also means that precision is arguably more important than recall for spam classifiers, as precision reflects the extent to which FPs are minimized. The results here thus suggest that ensuring balanced training datasets, or resampling to rectify imbalanced ones, is key to further boosting classifier performance.

4.5 Singlish and the Singaporean Context

As covered in the model comparison, a key limitation of Naive Bayes classifiers is the fundamental assumption it makes that each feature makes an independent and equal contribution to the outcome - in this case, whether a message is ham or spam. The strict independence assumption is generally untrue in most situations, and especially so in language, where each token and sentence is linked and therefore not independent from one another. The equal contribution assumption is similarly limited. However, we know that in identifying spam data positively, certain keywords or subject matters such as those identified in our initial exploratory data analysis are more heavily weighted. The same holds for ham messages, where informal speech such as slang and Singapore Colloquial English or Singlish can be an identifying feature of human-generated text. Singlish can be an especially useful feature in distinguishing spam and legitimate messages, considering that code-switching is a prominent feature in diglossic language ecologies such as Singapore. The formal variety of Singapore Standardised English (SSE) is often employed in corporate settings such as communications from companies and businesses, whereas Singapore Colloquial English (SCE) is the variety of choice in informal social communication (Cavallaro et al., 2014). Considering that spam and ham can also be differentiated along the lines of commerce versus social interaction, this allows us to use Singlish as an index of legitimate ham messages.

Future implementations of spam filters in a Singapore context aiming at identifying Singlish may adopt a few strategies. First could be to build a dictionary of Singlish words to pick up on Singlish lexical items. These could include Singlish's well known discourse particles, e.g. *la, hor, meh,* etc. A useful training dataset would be the National Speech Corpus by Singapore's Infocomm Media Development Authority, consisting of over 3000 hours of natural conversational Singlish data (Koh et al., 2019). More work has also been done by Hsieh et al. (2022) recently in building 'Singlish Checker', an NLP tool for Singlish detection and analysis. As part of their efforts, a list of 1628 distinct Singlish words and phrases has been assembled from publicly available sources such as forums and smaller pre-existing dictionaries. However a difficulty with using Singlish lexicon as the basis for detection arises as companies often seek to localise their advertisements and appeal to the local context, leading to the use of Singlish in marketing material in an attempt to better appeal to Singaporean consumers (Lou et al., 2023). This means that the efficacy of only relying on a Singlish lexicon search may be insufficient. It could also be useful to pick up on distinctive features of Singlish syntax. These could include

particle stacking (Boo et al., 2023), topic prominence (Zhiming & Min (2005), and agreement-drop (Lee, 2022).

For future work on text message classifiers, a current major concern in Singapore is scam messages, particularly those targeting vulnerable poor and/or elderly populations. These often take the form of phishing messages attempting to fool recipients into believing the message is from a legitimate organisation. The classification of legitimate and illegitimate messages mirrors the spam/ham challenge. Flagging scam messages is thus an important ongoing challenge for message classifiers as well. The challenge is arguably even greater as phishing messaging explicitly seeks to appear similar to legitimate messages, and can present fruitful avenues for further computational linguistic research, possibly starting with the assembly of a phishing message corpus.

5. CONCLUSION

In this project, several text classification models were developed and evaluated for differentiating spam from non-spam messages. Using TF-IDF feature extraction with Naive Bayes, SVM and logistic regression classifiers, the models showed remarkable accuracy when applied to SMS data. The results underscore the effectiveness of the models in generating predictions within the context of familiar linguistic patterns.

However, our findings highlight a limitation of the models in classifying 'unseen' data, particularly from the LingSpam dataset. While our classifiers maintained a reasonable accuracy, they struggled with delivering good precision and recall scores. To circumvent this issue, we implemented oversampling and undersampling techniques, demonstrating their potential to improve model performance.

Moving forward, the findings provide a foundation for further analysis, aiming to optimise and identify the best coupling of feature extraction and classifiers. By evaluating the trained models on diverse and independent datasets spanning various communication platforms, our understanding of their effectiveness can be improved. Further research can also address the evolving challenges of the communication landscape, as scam messages have become a persistent challenge that demands innovative solutions beyond the capabilities of traditional classifiers. In the context of machine learning projects, we found that more often than not less is more and it is all about iterative hyperparameter tuning, there is no "one size fits all" solution for approaching a model because every dataset is unique in terms of the relationship between features which affects how the model should be chosen and trained.

References

Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2011). *Contributions to the study of SMS spam filtering.* Proceedings of the 11th ACM Symposium on Document Engineering - DocEng '11. doi:10.1145/2034691.2034742

Almeida, Tiago and Hidalgo, Jos. (2012). *SMS Spam Collection.* UCI Machine Learning Repository. https://doi.org/10.24432/C5CC84.

Androutsopoulos, J. Koutsias, K.V. Chandrinos, George Paliouras, and C.D. Spyropoulos, (2000). *"An Evaluation of Naive Bayesian Anti-Spam Filtering"*. In Potamias, G., Moustakis, V. and van Someren, M. (Eds.), Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000), Barcelona, Spain, 9-17.

Boo, A., Lee, J., & Tan, Y. Y. (2023). Particle stacking in Singlish–New data from the National Speech Corpus. Lingua, 287, 103513.

Brownlee, J. (2023, October 3). *A gentle introduction to k-fold cross-validation.* MachineLearningMastery.com. https://machinelearningmastery.com/k-fold-cross-validation/

Cavallaro, F., Ng, B. C., & Seilhamer, M. F. (2014). Singapore Colloquial English: Issues of prestige and identity. World Englishes, 33(3), 378-397.

Gandhi, R. (2018, July 5). Support Vector Machine - introduction to machine learning algorithms. Medium. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Hassan, M. A., & Mtetwa, N. (2018). Feature extraction and classification of spam emails. *2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 93–98. https://doi.org/10.1109/ISCMI.2018.8703222

Hsieh, L. H., Chua, N. C., Kwee, A. T., Lo, P. C., Lee, Y. Y., & Lim, E. P. (2022). Singlish Checker: A Tool for Understanding and Analysing an English Creole Language. In International Conference on Asian Digital Libraries (pp. 115-124). Cham: Springer International Publishing.

Koh, Jia & Mislan, Aqilah & Khoo, Kevin & Ang, Brian & Ang, Wilson & Ng, Charmaine & Tan, Ying-Ying. (2019). Building the Singapore English National Speech Corpus. 321-325. 10.21437/Interspeech.2019-1525.

Lee, S. K. (2022). On agreement-drop in Singlish: topics never agree. Glossa: a journal of general linguistics, 45(1).

Lou, C., Zhou, X., Huang, X., Xin Yun, G., Yan Jie, L., Kin Tat Bryan, L., & Xin Rong, P. (2023). Flattering Me in the Right Way: Exploring Language Use and Ethnic Cues in Localized Advertising Among Singaporean Consumers. Journal of Interactive Advertising, 23(1), 55-72.

Raj, A. (2021, January 5). The perfect recipe for classification using logistic regression. Medium. https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592

Zhiming, B., & Min, L. H. (2005). Systemic transfer, topic prominence, and the bare conditional in Singapore English. Journal of Pidgin and Creole Languages, 20(2), 269-291.