A Naïve Bayes Classifier for Arabic SMS Spam Messages

Dr. Irfan Ullah Abdurrab[1], Khalid Alsantali[2, 3], Omar Alsantali[2], Bandar Almutairi[2], Badr Alkhaldi[2], Sakher Alroque[2]

[1] AI Department, Imam Abdulrahman bin Faisal University, Dammam, Saudi Arabia
[2] Senior CS Student, Imam Abdulrahman bin Faisal University, Dammam, Saudi Arabia
[3] Correspondence: K.Alsantali@Gmail.com

***Abstract*— SMS Spam messages are an incredibly common attack vector on laypeople in regard to identity theft, bank fraud, etc. In the Arabic-speaking world, there is very little research regarding stopping this manner of attack, leaving users there very vulnerable. From this motivation we went about to propose a Naïve Bayes classifier with a specific set of preprocessing steps and an Arabic SMS Spam dataset geared towards the most common types of attacks. Results achieved are 96% accuracy, 97% precision, and 90% recall. Matching other classifiers in accuracy while at the same time being a lot faster and more efficient.**

***Index Terms*— Arabic SMS Spam, Naïve Bayes, Classification, Machine Learning**

I. Introduction

Although the popularity of the casual use of SMS messages as a communication method may have dwindled significantly recently as the rise of internet-based messaging services came to a head, they are still very frequently used to authenticate, inform, and communicate with users. Official organizations often use SMS messages to send important messages to users. This expectation from users leads laypeople to often mistrust spam messages and willingly give up their information to a malicious party.

Spam refers to any message or communication to the user that is intended to masquerade or otherwise deceive the user into believing it is sent from an official source. Examples include messages telling users to contact a scam number to avoid getting their bank account frozen. Other types of spam include phishing links urging users to sign into a fake website, and advertisements that were not explicitly subscribed to by the user.

According to RoboKiller, 13.1 billion spam texts in the US were sent in April 2023. [1] A survey illustrates that 68% of mobile phone users are affected by SMS spam, which often contains those above mentioned harmful messages. [2]

In comparison with Email, spam detection is not as robust with SMS messages, the former able to harness the processing power of the Email host provider and the ability of an internet connection. SMS messages must rely on the comparatively minuscule processing power of the smartphone and is not always able to harness an internet connection.

The importance of identifying spam is well-known and classifying it is well-studied, but only in English. For Arabic users this still continues to be a problem as the lack of research and development is ongoing. As well as the lack of data regarding the importance of this subject. During the course of this research we also aimed to collect a useful and important dataset to further push and accelerate the rate of research in this space. This is a challenge in and of itself as data is difficult to come by and so collection was done manually through local phones.

Other challenges were also present during this research. A central goal was the ability to actually deliver the results of classification to users in an easy way. This is done best by creating an app which could, on the spot, classify messages into spam or non-spam. This, of course, implies that the ML model must be able to run quickly and efficiently, which yet presented another challenge.

After much consideration and comparison of statistical classification models and previous literature based on English spam detection, we found that Naïve Bayes classifiers were both accurate enough in NLP contexts and were incredibly efficient with system resources when compared with other models such as Deep Learning.

And so, we present the main contributions of this paper as:

- A dataset that has a wide array of types of spam that is incredibly useful for conducting future studies on this subject
- A Naïve Bayes model for detecting Arabic spam which boasts an accuracy of 96%
- An android application that can, on the spot, quickly identify spam messages and inform the user.

II. Previous Literature

[3] is the most informative piece of literature regarding Arabic SMS spam filtering. The researchers proposed a hybrid

deep learning model for detecting spam SMS messages. It is based on a combination of two deep learning models, which are CNN and LTSM.

It was devised with the intention of recognizing two languages, Arabic and English. The authors also compared this algorithm to other common methods, of which are SVM, KNN, Multinomial Naïve Bayes, and many others. This study used two separate datasets for each language, the first one being an English dataset from the UCI repository, and the other being one the authors collected using local smartphones in Saudi Arabia.

Unfortunately, the latter is nowhere to be downloaded and used by us in this project. Regarding the authors' conclusion, they found that their proposed CNN-LTSM hybrid algorithm performed the best with a 98.37% accuracy compared to 97.83%, 90%, and 97.83% for SVM, KNN, and Multinomial Naïve Bayes respectively.

[4] is a paper describing a proposed system for filtering SMS messages that uses a Naïve Bayes filter as a first classifier and a Neural Network as a second classifier. The paper details the methodology the authors used. There are two methods used to collect SMS spam messages.

1. A Collaboration-based method in which users submit feedback and experience regarding the data. Eliminating guess work or other more time consuming tasks to determine the type of SMS message.
2. A Content-based method which relies on analyzing the textual content of messages. This is more common due to the difficulty of acquiring data related to SMS messages in general and for Arabic specifically.

Preprocessing is also mentioned as a critical part of the algorithm's preparation. White spaces are removed and stop words are extracted and used as features to help filter SMS messages more efficiently. Six features are described in the paper such as the presence of a phone number, and non-alphanumeric characters. As well as others.

The results of this system for Arabic filtering were 95% accuracy with six features and 70% of the dataset for training. This paper describes an important part of AI filtering which is preprocessing and the multiple steps taken to prepare the data for the classifiers.

[5] is another paper that utilized Naïve Bayes filter as a classifier for a proposed Arabic SMS spam filtering system. The researchers adopted a workflow of multiple phases before feeding the NB filter with the SMS messages. The first phase is the preprocessing phase which eliminates white spaces and stop words. The second phase is feature extraction in which the researchers used a total of 15 features, some of which had a higher effect on how NB classified SMS messages.

For example, one of the extracted features is the words ratio in a message and it had the most effect on classifying non-spam messages, while the number of non-alphanumeric characters had the highest effect on classifying spam messages.

After extracting the features, normalization is applied to reduce the variance of values, and then features will be selected before classification to reduce the number of irrelevant and redundant features. Removing such features proved to have an increase in the system performance, the researchers applied TF, IG, and GR selection methods.

As for the findings, with a dataset of 400 messages in which 70% were considered for training an accuracy of 83% was reached, and it was raised up to 88% after applying TF with 12 features. And thus, concluding that to achieve the best results it is suggested to use a feature selection method along with NB classifier.

III. Proposed Methodology

A. Design of the overall model

In this section we describe the machine learning model intended to accomplish the task of classifying spam messages. This model is based on a Naïve Bayes classifier, of course, the preprocessing steps play a large part, especially in a language such as Arabic.

The proposed system is comprised of a few steps. First, training the model. To accomplish this we collected approximately 495 SMS messages including phishing, advertisements, informational, and conversational messages. They are preprocessed, vectorized, saved into a dictionary, and then using Naïve Bayes, statistically classified into spam or non-spam.

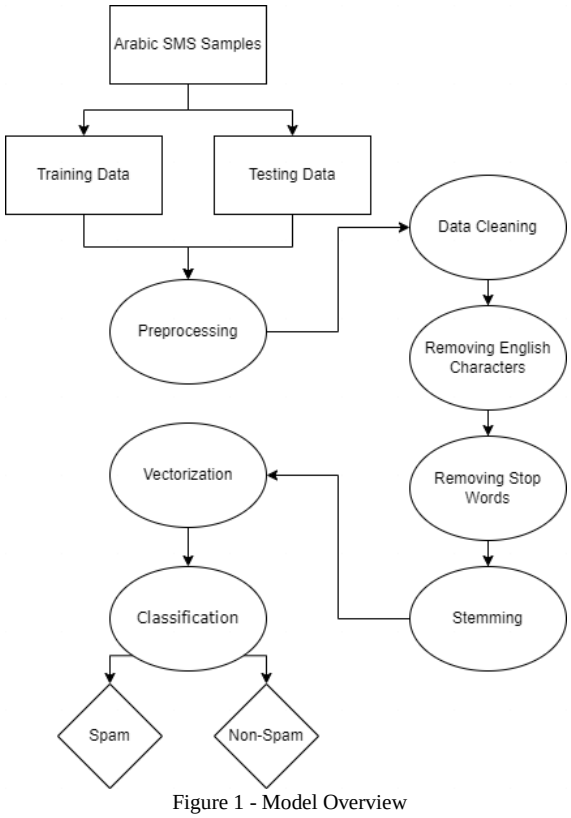

Figure 1 - Model Overview

The dataset was specifically geared towards the types of common attacks and harmful messages found here in Saudi Arabia. A majority of phishing messages were completely centered around bank account fraud.

	A	B
1	SMS	Sentiment
2	مصرف الراجحي تم إيقاف حسابك لتنشيط الحساب اتصل بنا ٢٧٥٠٧٣٥	1
3	!! إغلاق !! حسابك في الراجحي اتصل الان. ٠٥٨١٨٩٢٨٠٤	1

Figure 2 - Sample of dataset

B. *Preprocessing*

After splitting the data into training and testing randomly with 25-75 split, we applied the following preprocessing steps.

1. Data cleaning: this part of the preprocessing procedure simply removed all characters that do not carry meaning towards classifying the message as spam or non-spam. Examples being punctuation, diacritics, symbols, emojis, etc.
2. Removing all English characters from the message: during the course of this research we concluded that English characters, especially in Arabic spam messages, simply carry links or a translation of the same message, therefore they were better removed.
3. Stop words: stop words are tokens which are incredibly common during speech and carry essentially no meaning by themselves. Words in Arabic such as على, كلا, إلى, في, من etc. In English, examples would be “to, from, in”.
4. Stemming: stemming is the process of reducing a word into its most basic form. In English for example the words “jumped” and “jumps” are essentially the same, but a vectorizer would treat them differently because they are not identical words. Therefore, they are both turned into the basic “jump”. This is, of course, done in Arabic.

C. *Vectorization*

Simply applying the above does not make the messages ready for machine learning. Next, we need to transform the cleaned text into a numerical format for the classifier. We chose to do this using a count vectorizer. Vectorization separates each message into a vector with each word occupying one section. Coupled with each word is its count in the message. This is then encoded such that each word (better known as token) is given its own unique number. So for example the sentence “Out of the trunk, the branches grow, out of them, the twigs.” becomes

Out	Of	The	Trunk	Branches	Grow	Them	twigs
2	2	3	1	1	1	1	1

And is then further encoded into:

0	1	2	3	4	5	6	7
2	2	3	1	1	1	1	1

This is done for every message and, again, associated with the labels we put on each message, is fed into the Naïve Bayes classifier to organize into spam or non-spam.

IV. Naïve Bayes

Very briefly, Naïve Bayes is a classification algorithm based on Bayes’ theorem with the “Naïve” assumption of conditional independence between pairs of features. We used a multinomial Naïve Bayes implementation which is best suited for text classification purposes. Bayes’ theorem states the following, that given class variable y and feature vector $x_1, \dots, x_n$: [6]

$$P(y|\, x_1, \dots, x_n) = \frac{P(y)\; P(x_1, \dots, x_n \,|y)}{P(x_1, \dots, x_n)}$$

Where $P(y|\, x_1, \dots, x_n)$ indicates the posterior probability of the class, $P(y)$ indicates the prior probability, $P(x_1, \dots, x_n \,|y)$ indicates the probability of predictor given class, and $P(x_1, \dots, x_n)$ indicates the predictor’s prior probability.

V. Android Application

Of course, a major part of this research was applying the NB model to an android application. To accomplish this, we used Chaquopy, which is a plugin for Android’s Gradle-based build system that allows full integration of Python and Java inside Android. Using Chaquopy, we integrated a special Python file within an Android app that, after the Java section retrieves the SMS messages from the phone’s local storage, stores them and classifies them on the spot when the user taps to view the message.

In Python, we used pickle to serialize the model and fitted vectorizer after training was completed and packaged them in the Android app. This vastly reduces the overhead on the mobile processor and also lowers the requirements to run the app.

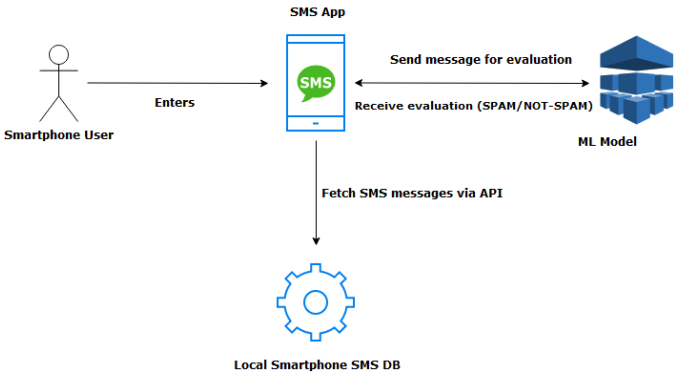

Figure 3 - Android System Design

The Android app runs a Python environment which runs the model and is able to call a predict function that classifies each message.

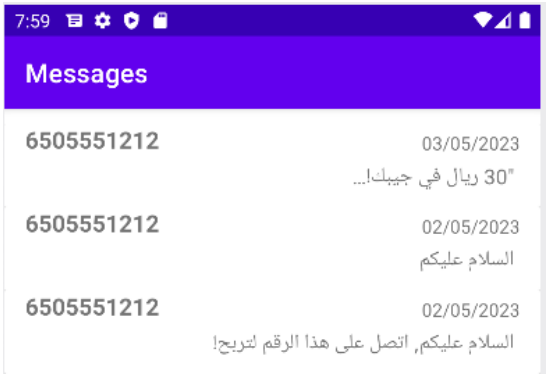

Figure 4 - Android App Main Menu

Figure 4 above showcases the finished main menu of the Android application. A preview of the message, the sender, and the date are specified for each message not unlike any common messaging app.

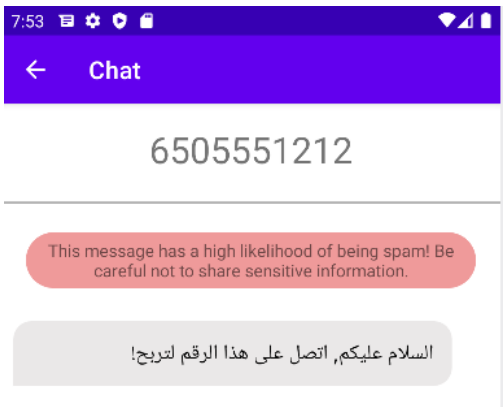

Figure 5 - Spam warning for a phishing message

Figure 5 shows a spam warning message for suspicious text. The text roughly translates to "Call this number to win!", which is common phrasing in phishing-based spam. Moreover, due to the uncertainty of the classification and the nature of error rates, a simple warning is shown instead of a definitive classification.

In Figure 6, we see a spam warning message for an advertisement. The text contains the usual advertisement phrasing, with keywords such as "coupon", "free", "discount", "riyal", "gift", and "customer", which are the biggest indicators of an advertisement message. These messages represent a very large portion of unwanted spam, and are extremely common in essentially every cellphone with an SMS feature.

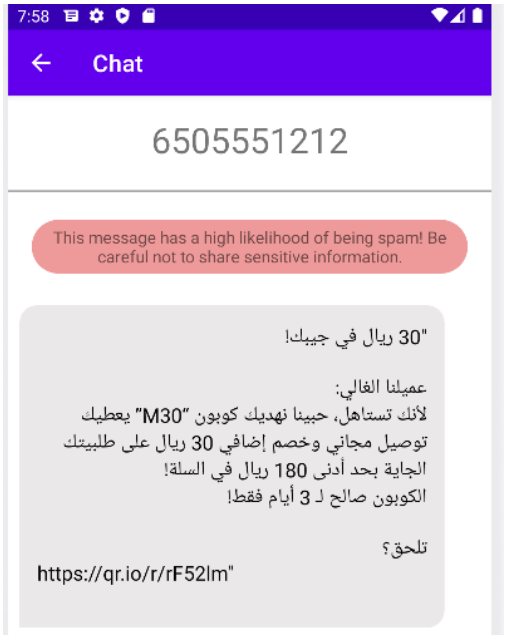

Figure 6 - Spam warning for an unwanted advertisement message

VI. Results & Benchmarking

We used three metrics to judge our model and compare it to other potential classifiers. First, accuracy, which is the percentage of correct predictions the classifier had in comparison to all predictions. Our model achieved an accuracy of 96% using unseen testing data. Second, precision, which is the accuracy of positive prediction (in this case, a message being a spam message is a positive prediction). Our model achieved 97% precision. The third is recall, which measures how many actual positive (spam) instances were correctly identified. Our model has a recall score of 90%.

Table 1 - Confusion Matrix

Accuracy: 96% Precision: 97% Recall: 90%		Actual	
		Spam	Non-spam
Predicted	Spam	37	1
	Non-spam	4	82

These results should provide high confidence in the model to accurately predict whether a message is potentially harmful to the user or not and reduce the risk of harm as the result of falling victim to such messages. Snippets of code and results can be found in Appendix.

Benchmarking for the Android application was also conducted on several devices and we found that the application takes very little processing power owing to its Naïve Bayes implementation which is a very quick and efficient algorithm. The major bottleneck of the application is loading the messages from system into RAM, which takes a few seconds, and initializing the Python environment also takes in practice upwards of 2 seconds, but after which the program runs almost instantly and very smoothly. Profiling for

the RAM usage was also done and was found to be less than 230MBs overall due to the recycling of resources when displaying and classifying messages.

VII. Conclusion

A. *Findings & Contributions*

As discussed in the results & benchmarks section, we developed a model that runs in an Android application that can classify whether an Arabic SMS message is spam or non-spam. Our model achieved an accuracy of 96%, 97% precision, and a recall score of 90%. As well as narrowing a very large research gap, we also produced a complete dataset that can be incredibly useful in future research in this field or even other fields such as general sentiment analysis.

Developing an Android application also allows this model to be used very easily by any user. A lightweight algorithm such as Naïve Bayes also allows a wide array of devices to be able to run this application as discussed in the benchmarking section.

Our proposed methodology helps to increase research and data in this field and decrease the potential harm of SMS spam messages in Arabic-speaking languages in general, but in Saudi Arabia in particular. Running the model on Android has the massive advantage of wide distribution and ease-of-use that the users of today need.

B. *Recommendation for future works*

Although we have created a high accuracy model for Arabic SMS messages, it is obviously still not perfect. Due to a lot of limitations discussed, there might be edge-cases where the model fails to recognize spam. The fact that the model's learning ability is essentially locked-in and it cannot continue to grow means it could be very easily circumvented in the future. Therefore, a more complex and intensive model that can use a reporting system which continues to learn would be an excellent continuation of our work.

VIII. References

[1] RoboKiller, "2022 Phone Scam Report," 2023. [Online]. Available: https://www.robokiller.com/robokiller-2022-phone-scam-report.

[2] Tatango, "Text Message Spam Inforgraphic," 2011. [Online]. Available: https://www.tatango.com/blog/text-message-spam-infographic/.

[3] A. Ghourabi, M. A. Mahmood and Q. M. Alzubi, "A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages," *Future Internet,* vol. 12, p. 156, 2020.

[4] H. H. Mansoor and S. H. Shaker, "Using Classification Techniques to SMS Spam Filter," *International Journal of Innovative Technology and Exploring Engineering,* vol. 8, no. 12, pp. 1734-1739, 2021.

[5] M. B. e. al., "researchGate," October 2018. [Online]. Available: https://www.researchgate.net/publication/329017526_Towords_building_an_anti-spam_Arabic_SMS. [Accessed 3 October 2022].

[6] scikit-learn, "Naive Bayes - Scikit-learn," [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html. [Accessed April 2023].