Email Spam Detection

Basant Abdelaal
Computer Engineering Department
The American University in Cairo
basantelhussein@aucegypt.edu

Dalia Elnagar
Computer Engineering Department
The American University in Cairo
daliawk@aucegypt.edu

Mariam Fawzy
Computer Engineering Department
The American University in Cairo
mariam_fawzy@aucegypt.edu

***Abstract*—Spam email is unsolicited and unwanted junk email sent out to random people for commercial purposes. The growing volume of spam emails raised the urge of solving this problem. This paper proposed a model for a combined dataset. After balancing, cleaning, and preprocessing the data, different models, such as CNN and LSTM, are used to classify the spam emails. To improve the accuracy, several features were added to both models; one of the features was the polarity of the email which was obtained through sentiment analysis. Sentiment analysis is the use of natural language processing to extract subjective information from the data. After trying different architectures and manipulating their hyper parameters, we found that the best model is the combination of LSTM and sentiment analysis. This model got an accuracy of 97.4%, which beats the baseline.**

Keywords—CNN, LSTM, Sentiment Analysis

I. INTRODUCTION

With the increased usage of the Internet for social and professional networking, online communication became an essential part of our daily life. E-Mails represent one of the most used mediums for formal communication in institutions, businesses, and universities due to its quickness and reliability. Due to its usefulness and spreading, spam emails were rapidly rising as well. Spam emails are considered a major threat in daily online communication. Spam emails account for generating around 57% of the total email traffic per year in 2020 [1]. This has resulted in the attacking the privacy of many users through, spreading of offensive material like pornographic content, unsolicited messages in the form of advertising and promotional materials and moreover, led to more financial strain and increased requirement of storage.

Spam Detection is an important research topic in the area of Natural Language Processing (NLP). Most of the work was carried out using traditional machine learning classifiers like SVM, Naïve Bayes, Decision trees, etc. In this paper, we will implement Deep Neural Network classifiers which are mainly Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM). In addition, attempts to extract features and integration with sentiment analysis model are experimented with.

II. RELATED WORK

Most of the work related to spam detection utilizes machine learning models, like support vector machines (SVM), Linguistic Inquiry, Word Count (LIWC), random forest (RF) and naive Bayes (NB) [2][3], which obtain detection rates up to a 96 % [4].

More recent work explored deep learning technologies. In [5], convolutional neural network (CNN) is proposed for spam detection with the addition of a semantic layer on the top of it. During preprocessing, WordNet and ConceptNet were used to detect word similarity. For dataset, the research used SMS Spam dataset (UCI repository) and Twitter dataset. The model achieves 98.65% accuracy on SMS spam dataset and 94.40% accuracy on Twitter dataset.

Another popular solution for this problem is the Long Short Term Memory (LSTM) architecture which is a variant of the Recursive Neural Network (RNN). Similar to CNN, LSTM has the ability to learn abstract features [6]. In [6], they used the same dataset as [5] and the data was preprocessed using word2vec, WordNet and ConceptNet. It was proven that LSTM outperforms traditional machine learning models.

One interesting approach was [3] which investigates the assumption that spam is a form of commercial communication where the semantics of its content should be shaped with a positive meaning. Thus, the paper investigates if the polarity of the message is a useful feature for spam detection. The polarity was integrated with traditional machine learning classifiers and proved to improve their accuracy in detecting spam.

III. Proposed Methodology

Our problem statement is to classify spam emails. Hence, we need to find the best model for the problem. We decided to try both CNN models and LSTM models as they are both common solutions in the literature.

Since sentiment analysis has been proven to improve spam detection in basic machine learning models, we will integrate sentiment analysis in the CNN and LSTM models and record its results.

Moreover, we intend to extract more features to be added as inputs by analyzing the available datasets.

A. *Datasets*

In this work, two open-source datasets were used. Both of the datasets composed of two columns, one for body text of the email and the other for the class label, spam or ham. Both datasets are from open-source datasets in Kaggle. The first dataset is Spam Filter dataset. It contains 5695 emails, 1368 of which are spam. The second dataset is Spam Mails dataset. It contains 4993 emails, 1462 of which are spam.

When exploring the distribution of the spam and ham classes, it is clear that both datasets are imbalanced where the ham class is the major class. In order to solve this imbalance and avoid biasing for the ham class, a new balanced dataset is created. The dataset is created by merging the spam emails from both of the datasets, and then randomly selecting approximately the same number of emails for the ham class, with ensuring that there are no duplicate records.

The new dataset contains a total of 6088 email, 2830 of which are spam emails. After shuffling the obtained dataset, 20% of the data was hold out for testing the persistence of the model against unseen samples.

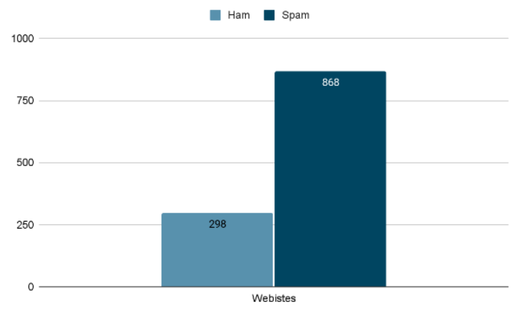
	Spam	**Ham**	**Total**
Kaggle Spam Filter dataset	1368	4327	5695
Kaggle Spam Mails dataset	1462	3531	4993
Our Balanced Dataset	2830	3258	6088

B. *Feature Analysis*

In an attempt to extract relevant features for the spam-ham problem so that we can use non-sequential models to further solve the classification problem with higher accuracy, experiments and analysis were carried out on the training dataset to identify and explore those features.

The first experimental analysis was to determine how relevant is website presence to the spam-ham classification problem. The motivation behind this feature is that spam emails are mostly sending malicious websites to users for further communication.

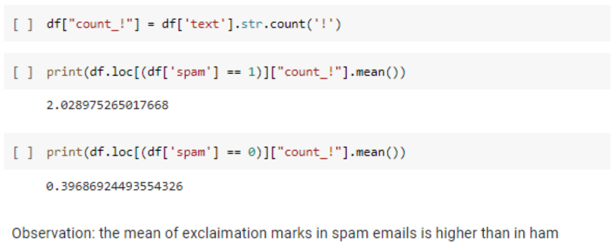
The analysis was done by counting the number of spam emails vs the number of ham emails containing the prefix "http" which refers to website's presence. The results are reported in the chard below.



According to the analysis results, of all the emails that have websites, 74% were actually spam emails. Therefore, another column was added to the dataset for the "website presence" feature, where the value for each email was either 0: “not present” or 1: “present”.

The second experimental analysis was to determine if the count of exclamation marks in the email is relevant to the spam-ham classification problem. The motivation behind this feature is that spam emails are mostly having an exciting tone and thus use many exclamation marks as an indication to this excitement.

The analysis was done by calculating the arithmetic mean of the number of exclamation marks in the spam vs the ham class. The results are reported below.



```
[ ]  df["count_!"] = df['text'].str.count('!')

[ ]  print(df.loc[(df['spam'] == 1)]["count_!"].mean())

     2.028975265017668

[ ]  print(df.loc[(df['spam'] == 0)]["count_!"].mean())

     0.39686924493554326
```

Observation: the mean of exclaimation marks in spam emails is higher than in ham

According to the analysis results, the mean of the exclamation mark count in the spam class was noticeably higher than that of the ham class. Therefore, a new feature was added to the dataset which the “exclamation count”, where the value for each email is the count of exclamation marks in it.

Lastly, according to the literature, we found a close relation between detecting spam and sentiment analysis of the email, where past works exploited this as a feature in basic ML models [3]. Accordingly, sentiment analysis was decided to be our third feature to experiment with. However, the feature value for each email is determined by the help of a pre-trained model to be further used in the training.

C. *Data Preprocessing*

The input dataset is raw text which needs to be preprocessed before using it as input to the model. First, contractions were expanded using Python Contractions Package. Afterwards, integers, newlines, tabs, extra spaces, and any character other than alphabet is removed using RegEx library and all characters were changed to lowercase.

Afterwards, the text was segmented to create an array of words for corresponding to each email before stemming and lemmatizing the words using WordNetLemmatizer. When the data was analyzed after this step, it was noticed that the words with highest frequency where stop words like “the”, “to”, “be” and “and”. These stop words were removed, by utilizing nltk.corpus, along with words with least frequency in order to minimize the dictionary and keep only the useful words.

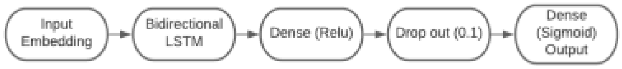
Finally, each word was given a token and the data was embedded using Keras Embedding function in order to be entered in the models.

Later in the project, an additional step was added which was analyzing the sentiment of each email using TextBlob.

D. *BaseLine Model*

To properly compare our results with the current approaches to the problem, the model implemented by Isra’a, et al. is used as our baseline model [7]. The used dataset in this work is the same Spam Filter Dataset from Kaggle. Therefore, it is reasonable to compare the performance of our model to the model in this work.

The proposed model in this work was using BiLSTM. It takes input from the embedding layer and its output vectors are fed to a Dense Layer. The activation function used in the dense layer is Relu. Then a drop out of 0.1 is applied to avoid overfitting. Finally, a dense layer with Sigmoid activation function is used for classification to normalize the output. Below is illustration of the model.



The final results of the model was achieving 96.43% on the testing data.

IV. Experiments and Results

The training phase is divided into 3 phases. In the first phase, we had trained our data using CNN and LSTM separately. After that, we started to add additional features to the best model we got out of the CNN and LSTM models as explained below:

Phase 1(CNN & LSTM separately)

CNN:

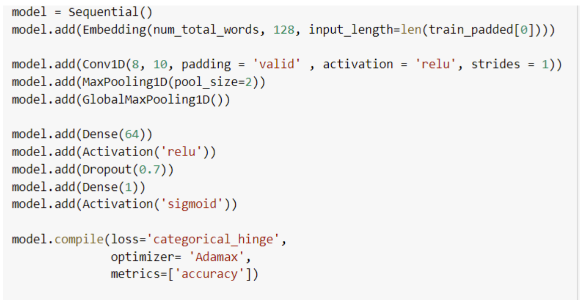
In the CNN model, we manipulated the following hyper parameters:

- Loss Function
- Optimizer

- Embedding Depth
- Number of filters
- Number of convolutional layers

Architecture:

After tuning the hyper parameters, we found that the best CNN architecture for training this combined data set is the following:

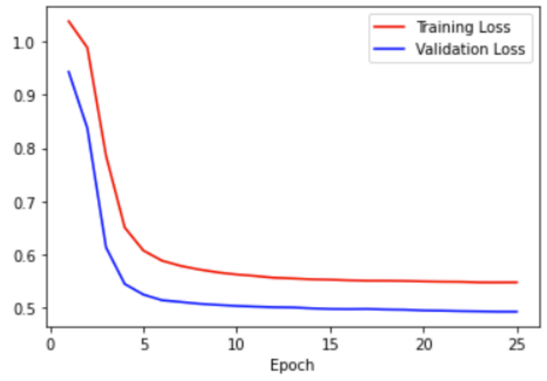

```
model = Sequential()
model.add(Embedding(num_total_words, 128, input_length=len(train_padded[0])))

model.add(Conv1D(8, 10, padding = 'valid' , activation = 'relu', strides = 1))
model.add(MaxPooling1D(pool_size=2))
model.add(GlobalMaxPooling1D())

model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.7))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='categorical_hinge',
              optimizer= 'Adamax',
              metrics=['accuracy'])
```

The output of the final CNN model:



```
Test score: 0.4920703172683716
Test accuracy: 0.9629005193710327
```

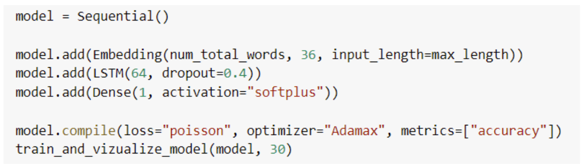
LSTM:

In the LSTM model, we manipulated the following hyper parameters:

- Loss Function
- Optimizer
- Embedding Depth
- Dropout ratio
- Depth of LSTM layer

Architecture:

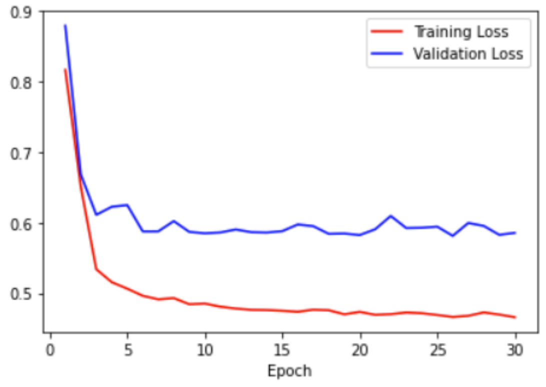
After tuning the hyper parameters, we found that the best LSTM architecture for training this combined data set is the following:


```
model = Sequential()

model.add(Embedding(num_total_words, 36, input_length=max_length))
model.add(LSTM(64, dropout=0.4))
model.add(Dense(1, activation="softplus"))

model.compile(loss="poisson", optimizer="Adamax", metrics=["accuracy"])
train_and_vizualize_model(model, 30)
```

The output of the final LSTM model:

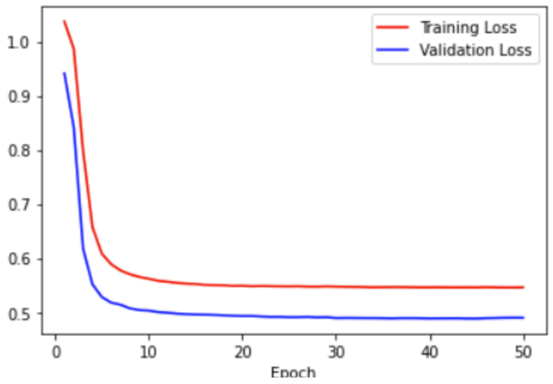


```
Test score: 0.5858830809593201
Test accuracy: 0.9696458578109741
```

Phase 2:

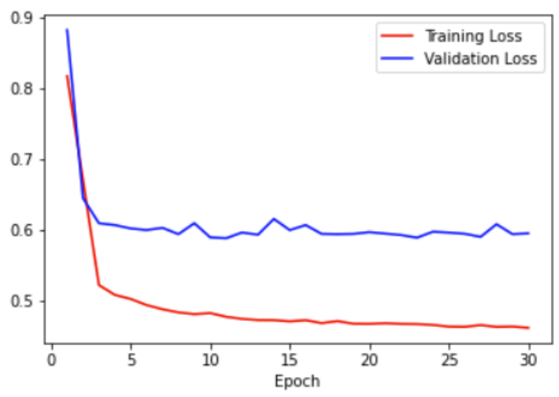
After getting the best CNN and LSTM architectures in phase 1, we started to add extra features. In this phase, we added sentiment analysis features to both model separately and the output was as following:

CNN:



```
Test score: 0.48976385593414307
Test accuracy: 0.9629005193710327
```

For LSTM:



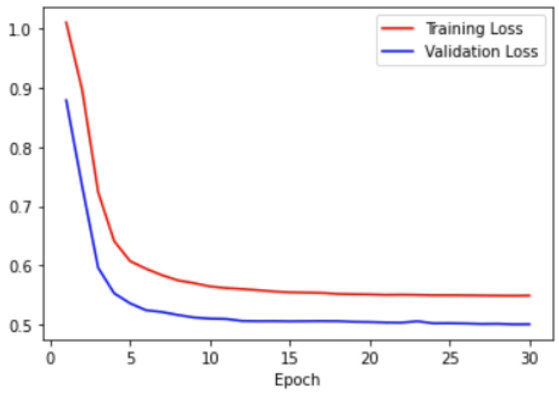
```
Test score: 0.594494640827179
Test accuracy: 0.9739933013916016
```

Phase 3:

In this phase we added multiple features one by one as following:

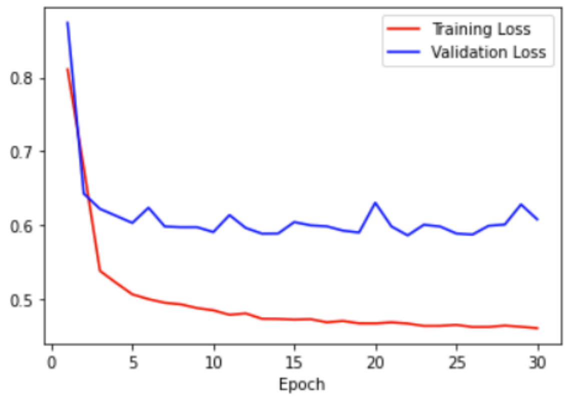
Output of adding the number of exclamation marks feature to both models:

For CNN:



```
Test score: 0.500859797000885
Test accuracy: 0.9537426233291626
```

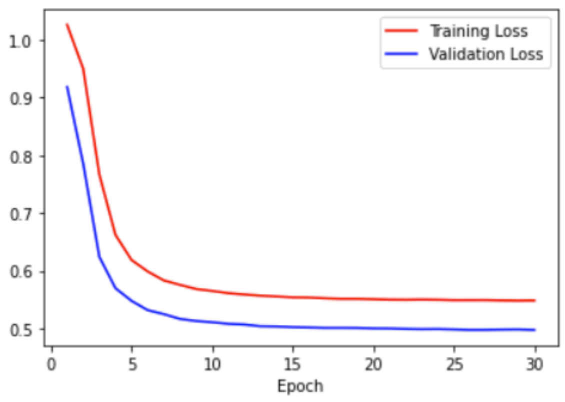
For LSTM:



```
Test score: 0.6080095767974854
Test accuracy: 0.9714045524597168
```

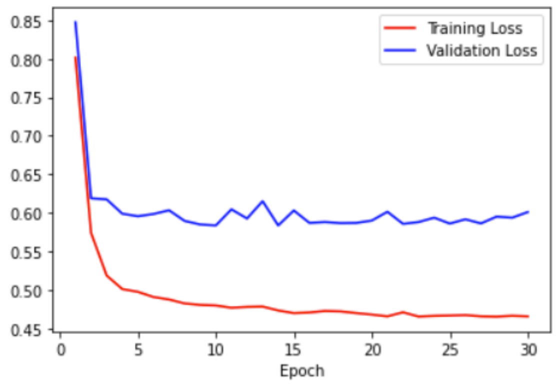
Output of adding the existence of website feature to both models:

For CNN:



```
Test score: 0.49781838059425354
Test accuracy: 0.9545837044715881
```

For LSTM:



```
Test score: 0.6011539697647095
Test accuracy: 0.9722455739974976
```

Summary of the 3 phases:

	LSTM	CNN
No extra features	96.9%	96.3%
Sentiment	97.4%	96.3%
Website Feature	97.2%	95.4%
Exclamation Count Feature	97.1%	95.3%
BaseLine BiLSTM	96.43%	

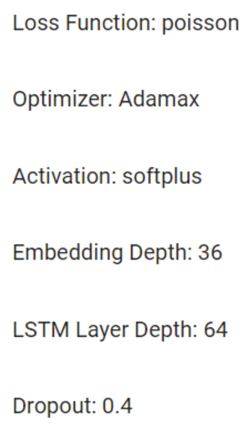
V. Future Work

People who are willing to work on the same problem in the future are recommended to do the following:

- Search for a bigger dataset or combine different datasets
- Make combinations of different models such as LSTM and CNN or add several features to one of them.
- Use Bert for the word embedding(state of art)

VI. Conclusion

CNN is not suitable for this problem as it overfits in the early epochs and it does not show any progress by adding extra features. Thus, the best architecture for training this combined data set is LSTM using the following hyper parameters in addition to adding the sentiment feature:



Loss Function: poisson

Optimizer: Adamax

Activation: softplus

Embedding Depth: 36

LSTM Layer Depth: 64

Dropout: 0.4

In conclusion, the best model for training this data set is:

LSTM
Sentiment Analysis
Best Model

References

[1] Johnson, J. (2021, July 20). Spam statistics: Spam E-mail traffic share 2019. Statista. Retrieved December 18, 2021, from https://www.statista.com/statistics/420391/spam-email-traffic-share/Neural Network," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 389-394, doi: 10.1109/ICSSIT48917.2020.9214159.

[2] Jochen Hartmann, Juliana Huppertz, Christina Schamp, Mark Heitmann, Comparing automated text classification methods, International Journal of Research in Marketing, Volume 36, Issue 1, 2019, Pages 20-38, ISSN 0167-8116, https://doi.org/10.1016/j.ijresmar.2018.09.009. (https://www.sciencedirect.com/science/article/pii/S0167811618300545).

[3] Ezpeleta E., Zurutuza U., Gómez Hidalgo J.M. (2016) Does Sentiment Analysis Help in Bayesian Spam Filtering?. In: Martínez-Álvarez F., Troncoso A., Quintián H., Corchado E. (eds) Hybrid Artificial Intelligent Systems. HAIS 2016. Lecture Notes in Computer Science, vol 9648. Springer, Cham. https://doi.org/10.1007/978-3-319-32034-2_7.

[4] Malarvizhi, R.: Content-based spam filtering and detection algorithms-an efficient analysis & comparison 1 (2013).

[5] Jain, G., Sharma, M., & Agarwal, B. (2018). Spam Detection on Social Media Using Semantic Convolutional Neural Network. International Journal of Knowledge Discovery in Bioinformatics (IJKDB), 8(1), 12-26. http://doi.org/10.4018/IJKDB.2018010102.

[6] Jain, G., Sharma, M. & Agarwal, B. Optimizing semantic LSTM for spam detection. Int. j. inf. tecnol. 11, 239–250 (2019). https://doi.org/10.1007/s41870-018-0157-5

[7] Isra'a AbdulNabi, Qussai Yaseen, Spam Email Detection Using Deep Learning Techniques, Procedia Computer Science, Volume 184, 2021, Pages 853-858, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.03.107.

[8]