2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

An Intelligent Email Classification System

Zili Luo
School of Computing
Queen's University
Kingston, Canada
zili.luo@queensu.ca

Farhana Zulkernine
School of Computing
Queen's University
Kingston, Canada
farhana.zulkernine@queensu.ca

***Abstract*—Email is one of the most common methods of official and personal communication to exchange information. For the administration department, dealing with hundreds of emails with the same type of inquiries or requests results in a huge operational overhead. In this study, we explore email classification models. An email classification system should understand the topics in the email content for categorizing emails and indicate if an incoming email should be handled by the mailbox owner. Email categorization based on topics is a multi-label classification task. Most existing email categorization models perform binary classification to identify spam, phishing, or malware attacks. We propose a CNN-BiLSTM model for multi-class email classification. Our experiments show that compared to the two other models that we implemented namely CNN (76.19%) and BiLSTM (61.9%) models, the CNN-BiLSTM (83.33%) and Hierarchical CNN-BiLSTM models (85.33%) have much better performance.**

***Index Terms*—Email Classification, CNN-BiLSTM, Natural Language Processing**

I. Introduction

Despite the emerging communication technology and the existence of a variety of meeting platforms, emails remain to be the most used communication method in all organizations. In 2019, 293.6 billion emails were sent and received each day, which was expected to exceed 347.3 billion daily by 2022 [1]. Despite the availability of a FAQ (Frequently Asked Questions) database, or information posted on websites, people often do not spend enough time searching for necessary information, and instead send emails to inquire about the information they need. Especially at the university, many emails contain similar questions during the time of admission, the beginning of the term, exams, and graduation. With the growth of Machine Learning (ML) and Deep Learning (DL) techniques, intelligent systems can be built to offload some of this work from the administration to automatically categorize emails [2]–[6]. An intelligent email response system can reduce the workload of answering these similar questions by categorizing emails based on the topic.

Historically, email classification algorithms were based primarily on probabilistic methods, with only two categories to identify: spam and non-spam emails [2]. Machine learning methods perform better than traditional probabilistic methods and have become increasingly popular in recent years [7]–[10]. In the word semantic vector space, a word embedding scheme such as GloVe [11] enables computers to understand natural language by linking a word or character to its corresponding vector (usually unique). A computer could use this vector to understand and calculate the meaning and correlation between words, find the nearest neighbors (synonyms), and visualize concepts and relationships between words and characters. It is possible to classify emails into multiple categories using the embedding vectors as inputs to machine learning and deep learning models [9], [12], [13].

This paper presents our work on multiclass email classification based on the email content. The challenges include the following: a) an email can include multiple topics which requires the email to be categorized under multiple topic classes which also increases the difficulty in generating automatic responses, b) absence of good training datasets, c) domain specific vocabulary and word/sentence context must be interpreted properly, d) email body can contain email chain i.e., previous emails and responses, and e) privacy of emails must be taken into consideration. In this study, we address all the above challenges by proposing a deep learning model for multi-class email classification and create a real life dataset by developing an email extraction tool for Microsoft Outlook. We apply natural language processing (NLP) and word embedding techniques to process the text and generate semantically rich embedding vectors to feed into our machine learning models. We compare the classification results with other models and demonstrate a superior performance in classifying emails for both a benchmark data set and our generated data set.

The rest of the paper is organized as follows. Section II presents the related work. The email extraction tool is described in Section III. Implementation, training and validation of the deep learning models are illustrated in Section IV. Section V presents the conclusion including a list of future work.

II. Related Work

The recent Convolutional Neural Networks (CNNs) such as ConvNets [14] achieved superior performance in topic classification when applied at the character level. In Natural Language Processing (NLP), the Long Short-Term Memory (LSTM) [15] model has demonstrated superior performance because it can not only extract data features from the input text but can also learn long sequential patterns in the text. Researchers have reported that CNN-based NLP models can also be used to classify documents, including emails, and achieve competitive results.

Michailoff et al. [3] apply the NB classifier and Deep Neural Network (DNN) to classify emails under different root folders such as Mobile and Fixed telephony, and other. The result shows that the DNN (84.2% Accuracy) performs better than the NB classifier (76.21% Accuracy) but it is ($2.5215 * 10^{11}$ complexity) more compute-intensive than the NB classifier ($1.28 * 10^{7}$ complexity).

Yasin et al. [7] compare performances of the RF (0.991 F1-score), the NB classifier (0.945 F1-score), the SVM (0.969 F1-score), the NN (0.977 F1-score), and the DT (0.984 F1-score) using F-measure for detecting phishing emails. The result shows that the RF achieves the best performance.

Alsmadi et al. [2] use three different SVMs to detect spam emails. Each of SVM is formulated on the top 100 frequent words, the top 100 frequent words after removing stop words, and N-Gram terms. They report that all three SVMs show a very high True Positive(TP) rate while the SVM with N-Gram shows the best False Positive Rate.

Kiritchenko et al. [4] improves the learning speed of the SVM (94.01% Accuracy) and the NB (91.51% Accuracy) classifier by applying a co-training algorithm. The algorithm trains two classifiers based on two different sets of features and each time one model only adds the most confident prediction to the class. They report that the SVM performs very well with the co-training algorithm while the NV classifier performs very poorly.

III. Email Extraction Tool

We develop a tool for extracting emails from a popular email management system such as Outlook to extract and use real emails to validate our proposed models.

The "pypff" library is used to extract emails from the Outlook email data file (.pst) and export them to a CSV file for further processing. This way the tool can run locally on the users' machine which would extract, anonymize, and package the data to send to our proposed IERS that runs locally or on a back-end server. The code can be packaged as an executable file to transfer and deploy on the user's machine. The entire system uses "os.outlook" library to be able to process email internally and send responses immediately.

A. Limitations

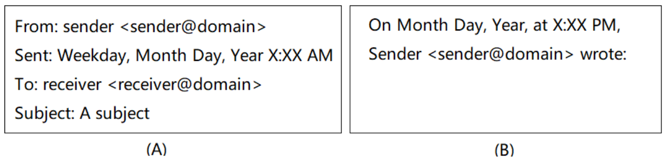
The tool we developed can only handle .pst and .ost files from the Outlook application and currently process only the English language. However, it can be extended to other languages with modifications to the regex, using different splitters and NER models matching the target language.

B. Email Processing

The data is stored in Pandas data frame format after being extracted from the Outlook email data file or directly from Outlook.

Processing Subject Line: We remove "re:" from the subject of the conversation.

Cleaning Email Body: The email body is processed using five regex expressions:


From: sender <sender@domain>
Sent: Weekday, Month Day, Year X:XX AM
To: receiver <receiver@domain>
Subject: A subject

(A)

On Month Day, Year, at X:XX PM,
Sender <sender@domain> wrote:

(B)

Fig. 1: Email Splitters(A:Splitter 1, B: Splitter 2)

1) Remove embedded hyperlinks
2) Remove unembedded hyperlinks
3) Remove redundant newline
4) Remove any invalid characters (anything other than alphabets, numbers, @, commas, colons, and dots)
5) Remove redundant spaces

Extracting Complete Conversation from Multiple Emails: Replies of emails often include the content from the previous email between two parties. To extract conversations between the same parties from subsequent emails, we select emails having the same subject line, and then extract the longest email, which contains all the dialogues denoting a complete conversation between two parties.

Splitting Conversation: Next, we split different topics of conversation based on the context with two splitters as shown in Fig. 1.

1) Email header (email header contains 4 new lines with "property:" and ending in "subject: subject of the whole conversation")
2) On [date] sender wrote:

A description of our algorithm is shown in Algorithm 1. Each conversation is assigned a unique "conversation_index" property that indicates whether an email with the same subject belongs to the same conversation.

Anonymization: Anonymization of email text is done using Stanford Named Entity Recognition tagger (Stanford NER tagger) [16], which can identify words that have the "person" property. A temporal dictionary is established for each conversion to store the occurrence of the same name in that conversion. The names are then replaced with random names from a name database. In the case the NER tagger fails to detect all the names, we apply a backtrack step that searches the email text for the same word in the temporal name dictionary and replaces it. It is possible to ensure that all the same names in the conversation will be replaced in the same way by utilizing the longest email in the conversation. However, NER tags cannot recognize non-English names or names within sentence fragments. The processed data can be exported as a CSV file or stored in memory for the next processing step as described below.

IV. Email Classification

We develop and validate several classification models to determine the categories of emails after they have been extracted.

CNN models have a parallel structure that captures spatial information better but ignores the sequence information from the input data. Therefore, we combine the CNN model with

Algorithm 1 Email Data Preprocessing

Input : A pandas data frame (DF_in) contains all the data from outlook data file

Output : A pandas data frame (DF_r) with emails in each conversation split down

```
1:  Remove all re: in the subject of the input data frame $DF\_in$
2:  Create an empty data frame $DF\_r$ to store the result
3:  Initialize a subject index $i$ = 0
4:  **for** each subject $S$ in data frame $DF\_in$: **do**
5:      Create a data frame $DF$ containing all emails with that subject sorted by DateTime while the top-most email is the latest email under that subject.
6:      **while** DF is not empty **do**:
7:          Get the first email $FE$ in $DF$
8:          Split $FE$ by the email splitters mentioned in Section III-B to get a list $dialogue\_list$ that contains each dialogue in this conversation.
9:          Create a temporary dictionary $nameDict$ to store names
10:         Processing each dialogue of the conversion with NER tagger, Replace all words with person entity with a random name. Depending on the length of continuous existence, will be replaced with first name (middle name) (last name) in the $nameDict$, if the name replace pair does not exist, replace with random and store the origin name: replaced name pair into the dictionary
11:         Append each dialogue to the $DF\_r$ with the subject index $i$
12:         Remove other email conversations that contain the same dialogues in $dialogue\_list$ in $DF$
13:     **end while**
14:     $i = i + 1$
15: **end for**
16: Return the result data frame $DF\_r$
```

the LSTM model which is good at learning sequences, into a CNN-BiLSTM model that can perform better in multi-label topic classification. CNN performs well on concise email data, but can not process long sequences, whereas LSTM assists with learning the sequence information.

We designed and implemented 4 different deep learning models: a) CNN, b) BiLSTM, c) CNN-BiLSTM, and d) Hierarchical CNN-BiLSTM model. The models are described below.

For training and validating the models, we use a publicly available dataset, Dataset_1, which has 18 different topic classes. The best performing model is then applied to a real life dataset, Dataset_2, having two classes. This dataset was created using our email extraction tool and the labels were added based on the email content and the sender information. The datasets are also described below in this section.

A. Model Description

1) CNN model: Rather than using a sequential architecture, CNN on the word level uses a parallel architecture

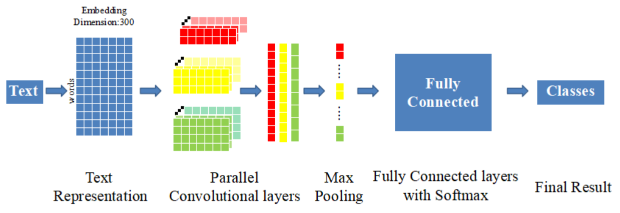


Fig. 2: CNN model structure

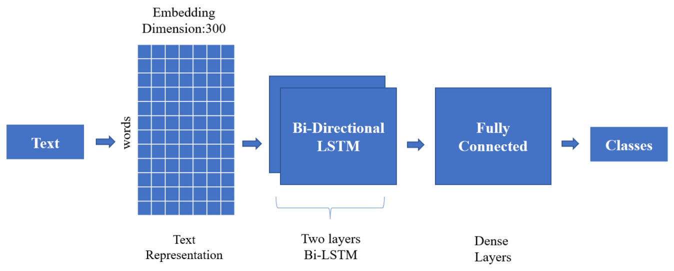


Fig. 3: BiLSTM model structure

similar to that used for image processing. By applying different filter sizes to the text representation, CNN models are able to extract features of varying lengths from the subtext of the email content. Following the max-pooling layer, these features reconstruct a new embedding of the entire email content to feed to the fully connected layer.

There are 100 filters of sizes 3, 4, and 5. Therefore, the embedding feed to a fully connected layer has a dimension of 300. As a result, there are three fully connected layers with dimensions of (300, 128), (128, 32), and (32, 14) respectively. All fully-connected layers except the last layer are activated using the ReLu activation function. All dense layers have a bias. Fig. 2 presents a visualization of the model architecture.

B. BiLSTM model

To compare the performance, we also implement a BiLSTM model as shown in Fig. 3. In this model, we use two layers of BiLSTM with a hidden size of 64 for each BiLSTM layer. There are two fully connected layers before prediction with sizes of (128,32) and (32,14) respectively.

1) CNN-BiLSTM model: In the CNN model, we add BiLSTM layers between the max-pooling layer and the fully connected dense layer. BiLSTM models are fed directly with data from the max-pooling layer. In this study, two BiLSTM layers with a hidden layer of size 64 were used to avoid over-fitting because the dataset is quite small. In the CNN-BiLSTM model, the BiLSTM replaces one of the dense layers with an input size of 300 and an output size of 128. Fig. 4 presents a visualization of the model architecture.

2) Hierarchical CNN-BiLSTM model: As the dataset is structured hierarchically, we apply a hierarchical classification model containing two parallel CNN-BiLSTM branches inspired by Kolisnik et al. [17], we have modified our CNN-BiLSTM model into a hierarchical model, which will attempt to classify the main-category and subcategories simultane-

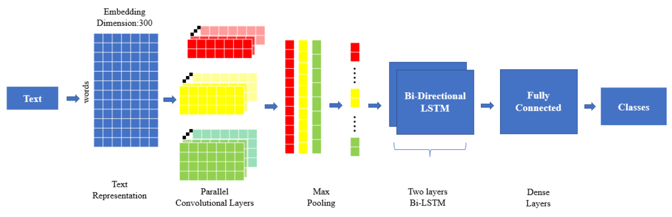


Fig. 4: CNN-BiLSTM model structure

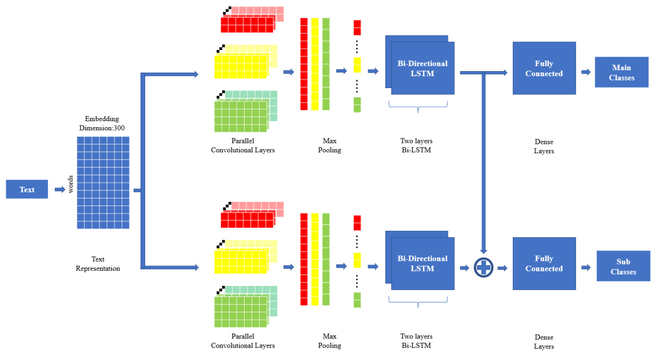


Fig. 5: Hierarchical CNN-BiLSTM model structure

ously. One CNN-BiLSTM branch (teacher branch) predicts the main-category in the same manner as the combined model (main pipeline) while the other CNN-BiLSTM pipeline incorporates input from the hidden state of the BiLSTM layer of the main pipeline into the hidden state of the BiLSTM layer in this pipeline. The hierarchical structure of the CNN-BiLSTM model is shown in Fig. 5.

We use a decaying weight w to control the bias of the loss function. Its initial value is 0.99, decays by 0.03 for each epoch, and has a minimum value of 0.1.

C. Datasets

We use the dataset proposed by Singh et al. [18] in this study as Dataset_1. The database contains 255 emails which are divided into three main-categories and 14 subcategories separating emails from an administrative section of a university, with nearly equal numbers of emails in each subcategory. For example, an IT department receiving requests for assistance with computer problems will have the department name as the main-category and the specific request type such as a clicker problem or a WiFi outage as a sub-category. Dataset_1 contains only the email body. Other information such as the subject, time, and sender is not included. The labels indicate the recipient of the email and the request type. A few examples from Dataset_1 are shown in Fig. 6.

Since Dataset_1 only contains the email body, we do not apply our Algorithm 1 described in Section III-B. However, we apply our algorithm to a real-life dataset, Dataset_2, as explained later. We created Dataset_2 for testing our method for a real-life use case scenario.

1) Extracted Real World Dataset: To demonstrate a real-world application of the model, we construct a simple dataset as dataset_2 by extracting emails from Outlook using the algorithm described in section III and divide the same into informative and non-informative emails. Emails that contain information about device maintenance, conference activities, meetings notice, and class broadcast are considered informative emails received from the Senior Systems Analyst, Graduate Program Assistant, and thesis Supervisor. In addition to informative emails, we label non-informative emails as those containing newsletters, event invitations, and surveys from the school. Dataset_2 contains 346 informative emails and 423 non-informative emails.

Description of the use case scenario including the validation results are given later in this paper.

V. Model Implementation

A brief overview and the architecture of the 4 models we implement in this study are presented below followed by the experimental validation the models.

A. Model Configuration

We use the categorical cross-entropy loss as (1) as the loss function and use the Adam optimizer for training the models. The learning rate for all the models is set to 3e-4 to avoid the model converging too fast to a sub-optimal solution, except for the Hierarchical CNN-BiLSTM which uses a decaying learning rate. All the layers in the model have a dropout of 0.2, and all the dense layers in the model have a bias.

All the BiLSTM layers use the Tanh activation function as (2). All the dense layers except the last before output use the ReLu activation function as shown in (3). The last dense layer before the output uses the Softmax activation function as shown in (4).

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{1}$$

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2}$$

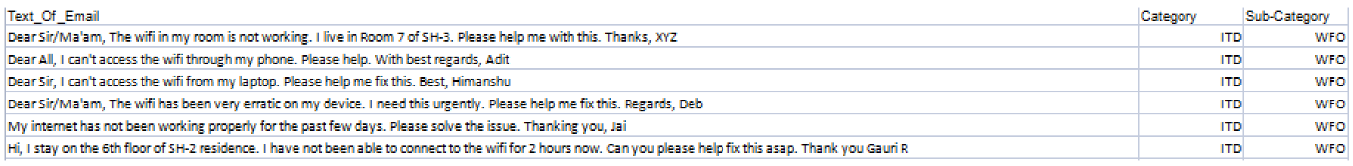
$$ReLu(z) = max(0,z) \tag{3}$$

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1,2,\dots,K \tag{4}$$

B. Experimental Setup

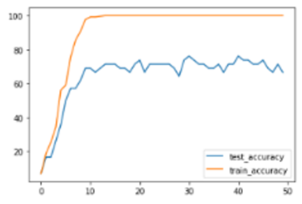
The email texts are padded with pad tokens to make all the emails have the same maximum length of 128 tokens. The text is tokenized using a tokenizer from the NLTK library [19], and each token is converted into a vector embedding of 300 dimensions using the GloVe library [11].

We select the same number of samples from each category as the testing set, which makes the leftover training data unbalanced. Thus, We use SMOTE (Synthetic Minority Oversampling Technique) library [20], on the training Dataset_2 to upsample and balance the training set.

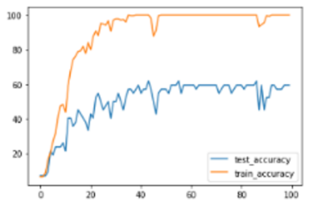


Text_Of_Email	Category	Sub-Category
Dear Sir/Ma'am, The wifi in my room is not working. I live in Room 7 of SH-3. Please help me with this. Thanks, XYZ	ITD	WFO
Dear All, I can't access the wifi through my phone. Please help. With best regards, Adit	ITD	WFO
Dear Sir, I can't access the wifi from my laptop. Please help me fix this. Best, Himanshu	ITD	WFO
Dear Sir/Ma'am, The wifi has been very erratic on my device. I need this urgently. Please help me fix this. Regards, Deb	ITD	WFO
My internet has not been working properly for the past few days. Please solve the issue. Thanking you, Jai	ITD	WFO
Hi, I stay on the 6th floor of SH-2 residence. I have not been able to connect to the wifi for 2 hours now. Can you please help fix this asap. Thank you Gauri R	ITD	WFO

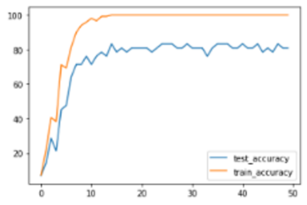
Fig. 6: Example data for email classification



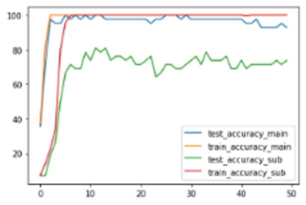
A) CNN model



B) Bi-LSTM model



C) CNN-Bi-LSTM model



D) Hierarchical CNN-Bi-LSTM model

Fig. 7: Accuracy curves during training process

C. Model Training

Dataset_1 was split at a 80-20 ratio with 80% training and 20% test data to train and validate the model performance. We tested other ratios, but increasing the testing set size would result in performance loss due to overtraining. Dataset_1 is small with only 255 emails which are divided into three main-categories and 14 sub-categories. We validate all 4 models (CNN, BiLSTM, CNN-BiLSTM, and hierarchical CNN-BiLSTM) and test their performances.

For the BiLSTM model, CNN and pooling layers were removed and the embedding was fed directly into the BiLSTM model. Each model except the BiLSTM was trained for 50 epochs due to fast convergence of the models. The BiLSTM model was trained for 100 epochs. We measure the performance of the model using accuracy as our validation method as shown in (5).

$$Accuracy = \frac{Number of correct predictions}{Number of total predictions} \tag{5}$$

D. Results

We compared the performances of LSTM and Bidirectional LSTM having one LSTM layer, two LSTM layers, and two Bidirectional LSTM layers. The latter (BiLSTM) achieved the best results in terms of classification accuracy. The charts showing training performance in Fig. 7 for CNN and CNN-BiLSTM models indicate that, in general, CNNs and CNN-BiLSTMs converge very quickly, reaching the best performance within 30 epochs, while the BiLSTM models converge slowly, and require many more epochs to achieve optimal performance.

TABLE I: Results on classification models

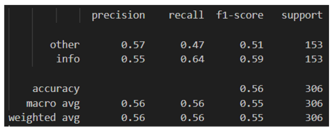
Model	Training Accuracy	Testing Accuracy
CNN	100%	76.19%
Bidirectional-LSTM	99.2%	61.9%
CNN-BiLSTM	100%	83.33%
Hierarchical CNN-BiLSTM	100% main 100% sub	96% main **85.33%** sub

From our results in Table I, all the models have a huge gap between their training and test accuracy, which might be due to overfitting issues. We also found that the LSTM model does not learn from the dataset under some conditions, resulting in very low accuracy for both training and testing. We tested using the direct output and final hidden state and changing the number of layers. The BiLSTM model with the final hidden states used as the input to dense layers gave the best performance. We also tested the LSTM model, which gives 17% training accuracy and 13% testing accuracy. The bad performance of the BiLSTM model may be caused by the short input data or the embedding layers feeding embeddings in an unexpected manner.

Compared to the CNN or BiLSTM, the combined model achieved better results. Additionally, we examined how SMOTE affected the training dataset. In general, SMOTE resulted in approximately 40 more training examples, but did not have a significant impact on the result with the maximum length of each sentence being set to 128 characters. When the maximum sentence length was set to a larger value, SMOTE produced even worse results.

We investigated the accuracy of each class. The model always seemed to have difficulties with ADC (Adding Course), COF (Course Offered), and WFO (WiFi Outage). By reviewing the dataset, we discovered that most of the emails in the WFO category mention "WiFi". Although a very small percentage in this category did not have this indicator, emails in the other categories had similar indicators which affected the model performance. The other two categories experience similar results.

In general, since the dataset is so small and has too many categories, each category has only around 18 emails. Only 80% of these emails are used for training. Therefore, there may be no similarity in features among the emails within the same category. The model cannot learn the important features

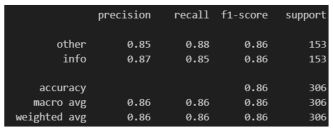


```
              precision    recall  f1-score   support

       other       0.57      0.47      0.51       153
        info       0.55      0.64      0.59       153

    accuracy                           0.56       306
   macro avg       0.56      0.56      0.55       306
weighted avg       0.56      0.56      0.55       306
```

A) Validation result of scenario 1



```
              precision    recall  f1-score   support

       other       0.85      0.88      0.86       153
        info       0.87      0.85      0.86       153

    accuracy                           0.86       306
   macro avg       0.86      0.86      0.86       306
weighted avg       0.86      0.86      0.86       306
```

B) Validation result of scenario 2

Fig. 8: Validation result of real-world data

with such small number of data points.

VI. Real World Use Case Scenario

We created a real world email dataset, Dataset_2 by using our email extraction tool which extracts, cleans, and anonymizes the data. We added labels based on subject, content and the sender to classify the emails into two categories: informative and non-informative.

Since Dataset_2 contains only two categories and is quite different from Dataset_1 that we used for model training, we follow two approaches to validate the model as explained below.

1) We use our pre-trained CNN-BiLSTM model on Dataset_1 in Section IV-B1, fine tune it on Dataset_2 and set a threshold on softmax result. If the result is above the threshold we classify the email as informative and otherwise as non-informative.
2) We modify the model architecture and completely retrain the model with Dataset_2 to test the model performance.

For validation 1, we fine-tuned our pre-trained CNN-BiLSTM model for another 50 epochs on our simple Dataset_2 using the same loss function, learning rate, and optimizer as stated in Section V-A on the real-world data. Then we try to find the best threshold by iterating the threshold value from 0 to 1 with an increment of 0.05 for each step. The model gives the best performance when the threshold is set to 0.45, which gives a 0.56 F1 score. The validation result is shown in Fig. 8.A

With validation method 2, We change the shape of the final dense layer to (32,2) and train the model from scratch on Dataset_2 for 50 epochs using the same loss function, learning rate, and optimizer as stated in Section V-A. The model gives 99.56% training accuracy and 86% testing accuracy, 0.86 F1-score. The validation result is shown in Fig.8.B

The huge difference in performance can be attributed to the different vocabulary sizes. The vocabulary size in Section IV-B1 is only 1,441 while Dataset_2 in this section has a vocabulary size of 15,067. With the aid of our extraction algorithms, we validate that our model is capable of handling email classification tasks in real-life use-case situations.

VII. Conclusion

This paper presents an email classification model and an email extraction tool. Our email extraction algorithm can overcome the privacy issue and extract full conversations instead of individual emails to make it easier to collect data for model training. We design, implement, and compare the performance among CNN, BiLSTM, CNN-BiLSTM, and Hierarchical CNN-BiLSTM models using a university departmental email dataset. We found CNN-BiLSTM model and the Hierarchical CNN-BiLSTM models have a similar performance, which is much superior to the CNN and BiLSTM models. We also showcase a real-world application of our email pipeline which includes email extraction and classification. Our model achieves a satisfactory performance of 86% accuracy with the real-life email data.

References

[1] M. Mohsin, “10 email marketing statistics you need to know in 2023,” https://www.oberlo.com/blog/email-marketing-statistics, Jan 2023, unpublished.

[2] I. Alsmadi and I. Alhami, “Clustering and classification of email contents,” *Journal of King Saud University - Computer and Information Sciences*, vol. 27, no. 1, p. 46–57, 2015.

[3] J. Michailoff, “Email classification: An evaluation of deep neural networks with naive bayes,” 2019.

[4] S. Kiritchenko, “Email classification with co-training,” *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, 04 2002.

[5] C. Li, H. Wang, Z. Zhang, A. Sun, and Z. Ma, “Topic modeling for short texts with auxiliary word embeddings,” *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.

[6] A. Vellino and I. Alberts, “Article assisting the appraisal of e-mail records with automatic classification,” *Records Management Journal*, vol. 26, 11 2016.

[7] A. Yasin and A. Abuhasan, “An intelligent classification model for phishing email detection,” *International Journal of Network Security & Its Applications*, vol. 8, pp. 55–72, 07 2016.

[8] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, “A systematic literature review on phishing email detection using natural language processing techniques,” *IEEE Access*, vol. 10, pp. 65 703–65 727, 2022.

[9] M. Du, J. Vidal, and Z. Al-Ibadi, “Using pre-trained embeddings to detect the intent of an email,” 02 2020.

[10] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, “Phishing email detection using natural language processing techniques: A literature survey,” *Procedia Computer Science*, vol. 189, pp. 19–28, 2021, aI in Computational Linguistics.

[11] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” vol. 14, 01 2014, pp. 1532–1543.

[12] C. W. Schmidt, “Improving a tf-idf weighted document vector embedding,” *ArXiv*, vol. abs/1902.09875, 2019.

[13] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” 01 2019, pp. 3973–3983.

[14] X. Zhang, J. Zhao, and Y. Lecun, “Character-level convolutional networks for text classification,” 09 2015.

[15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[16] J. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by gibbs sampling,” 01 2005.

[17] B. Kolisnik, I. Hogan, and F. Zulkernine, “Condition-cnn: A hierarchical multi-label fashion image classification model,” *Expert Systems with Applications*, vol. 182, p. 115195, 2021.

[18] A. Singh, D. Mishra, S. Bansal, V. Agarwal, A. Goyal, and A. Sureka, “Email Dataset for Automatic Response Suggestion within a University,” ”https://figshare.com/articles/dataset/Email_Dataset_for_Automatic_Response_Suggestion_within_a_University/5853057”, 2 2018.

[19] E. L. Bird, Steven and E. Klein. O’Reilly Media Inc., 2009.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, jun 2002.