To Read or To Do? That's The Task

Z. Alibadi and J. Vidal
Computer Science and Engineering Department, University of South Carolina, Columbia, SC, USA

Abstract— *This research studies the problem of email overload and proposes a system that automatically detects whether the email is "to read" or "to do". The goal of our research is to test if we could automate both the features extraction and sentence classification phases by using word embedding and a deep learning model that consist of CNN and LSTM. We use word embeddings, trained on the entire Enron Email dataset, to represent the input. Then, we use a convolutional layer to capture local tri-gram features from the input, followed by a LSTM layer to consider the meaning of a given feature (tri-grams) with respect to some "memory" of words that could occur much earlier in the sentence. We conduct experiments using several variations of model architectures with no handcrafted features. We evaluate the results on a subset of Enron Email Dataset. Our system works on the sentence level and is able to detect whether the sentence contains an intent or a delivery acts with an accuracy of 89%.*

Keywords*:* email overload, emails classification

1. Introduction

In 1982, Peter Denning (then the ACM President) first wrote about the problem of emails overload, calling it "The Receiver's Plight" and he asked the following question, "Who will save the receivers [of an email] from drowning in the rising tide of information so generated?" [1]. How is the situation now, after three decades? [2] expects the number of business emails sent and received per user per day to average 126 messages by the end of 2019. One consequence of the increase in email's usage is that users spend more time processing and managing their email. According to research conducted by McKinsey in 2012, reading and answering email accounts for 28% of the average employee workday. While the average number of unread work-related emails increased from (153) in 2006 [3] to (696) in 2014 [4]. A recent study, by [5] of more than 2 million users exchanging 16 billion emails over several months, confirms [4] results regarding email overload as they conclude that users are generally unable to keep up with rising load. So even after thirty-six years of Peter Denning's question, it seems that his question is still relevant.

In the following sections, we start by exploring the available commercial solutions for the problem of emails overload and their limitations. Then, we present our proposed solution. Finally, we compare our model's results with a baseline model and with other previous research works.

1.1. The available solutions and limitations

To overcome the challenges of managing email, [6] observed two main strategies users would follow to facilitate email's retrieval: organizing their mailbox using folders and/or utilize sorting and searching. Although foldering declutters the inbox, it requires efforts [7] as it requires a commitment from the user to invest considerable amounts of time and efforts to construct the folders' organization for future retrieval. As for searching and sorting, although these methods don't require initial efforts, they could affect productivity and the efficiency of email archive and retrieval, since a successful retrieval is highly dependent on predicting future retrieval requirements [6].

This has, in turn, proved to be a difficult cognitive task [8] since users must to remember where a particular email is located, the name of the targeted folder and the characteristics of the targeted email. All these challenges would explain why [9] found that users sometimes avoid foldering emails that later turn out to be useless or irrelevant. Other researchers argue that users are not foldering much of their emails. Reference [10] showed that 70% of users never created a single folder, while [4] found that inboxes show indication of having a large number of emails, fewer messages are archived, and labels are not as extensively used.

Throughout the years, several commercial solutions from the popular email clients were developed to provide alternative solutions for email's users to manage their emails. Yahoo mail offers Smart views, which provide search facets for messages, such as People, Social, Travel, Shopping and Finance [11]. The model behind this system works on structural emails by structural clustering (X-Clusters) based on the structure of the email's body. It utilizes several Classification features: subject/body words, user actions on emails (open, reply, delete, etc.), overall traffic volume (message distribution within a day/week) and structural feature of the email (how many HTML tag have, personal's emails has fewer). Gmail has been offering various ways for users to scan their inboxes, first with its Priority Inbox which classifies emails into specific tabs: primary, social, promotion, and updates [12]. Then with its Smart Labels and Inbox Tabs and more recently with its Inbox for Gmail, which supports automatic sorting into various Bundles (Travel, Purchases, Finance and Social). The model learns from user's interaction with his emails and request feedback. Also, it utilizes many other features including email content, HTML code, sender IP address. Microsoft offers "Clutter" and then "Focused Inbox"

in its email client, Outlook. Focused Inbox uses a tab system, with "Focused" and "Other" tabs. Low-priority e-mails get placed in the "Other" and what lands in the Focused Inbox is determined by an understanding of the people that the user interact with often, and the content of the email itself (e.g., newsletters, machine-generated mail, etc.).

In general, the task of automatically classifying emails proves to be a difficult task and there are several challenges associated with it. Reference [13] identify that these classification model should recognize the need to adapt to changing behavior. Also, [14] shows that user has very high expectations and being somewhat intolerant of errors. Furthermore, [15] identify two other difficulties: Thread drift and Topic drift. Thread drift means that a topic contains several threads of emails while Topic drift means that the same thread of emails contains information about different topics (i.e. the use the "Reply To" button instead of "New Mail"). Overall, this approach needs feedback from the user, produces better results on structural emails (machine generated emails), and requires time and considerable amounts of emails and its metadata to learn the user's interactions.

2. The proposed solution

The first step in our quest to tackle the problem of "email overload" is to better understand what emails users prefer and request from any email clients. Accessibility and visibility are the two most requested characteristics. Reference [16] found that users want email information to be more available at the surface level, which confirms [17] previous findings that user prefers availability and visibility. Therefore, we conclude that the best solution to serve the users' needs should have these two main characteristics: help the user read and access the email content more efficiently, and it should not add more complexity or require a change in users' behavior.

To achieve this goal, we choose to follow the steps of [18] by utilizing the speech acts theory and working on covering some of the possible speech acts associated with emails. A speech act is an utterance that serves a function in communication, any time a speaker offers a greeting, request, invitation, or refusal, he/she uses speech act. The most popular application of speech act theory is the classification of emails into Email Speech Acts [18]. This taxonomy is based on Speech Act Theory [19] [20] and characteristics of email.

Studying how users use emails will give us a better idea of which email speech acts to utilize for our work. One of the earliest study conducted to understand the uses of emails is [21]. He identifies three major forms of work management used in emails: information management, time management, and task management. Reference [6] shared some of [21] findings and discuss three main functions of email in their studies of 20 users' inboxes: task management, personal achieving, and asynchronous communications. As for differences between work emails and personal emails, [4] notice that work emails tend to be overloaded in email status (to read, to do) while personal emails tend to be overloaded in email type (bills, personal mail, promotional mail). Since we focus on work-related emails in our project, we will select the act of "Intent" (to represent the "to do" emails) and the act of "Delivery" (to represent the "to read" emails).

Another key difference of our approach is that instead of classifying emails into predefined categories (ex: Finance, Sport, Promotion, etc.) and then moving the emails into separate folder/tab/bundle, we will augment the email with either intent if its content is a request, commit or propose, or augment it with "delivery" base on the above act's definition. The idea behind our approach is rather than trying to learn the preference of the user and then classifying and moving the emails out of the inbox folder, we should focus on mining the content of the emails while "sitting" idle inside the user's inbox and then push to the surface relevant information about the nature of the email's content and make it visible to the user to help him/her decide whether to process this particular email or not. As a result, the user would save the time and effort of clicking and opening the email.

2.1. Pre-processing the email dataset

A large corpus of real-world emails subpoenaed from Enron Corporation was placed in the public record and made available to researchers. The data consists of over 500,000 email messages from the email accounts of 150 people. We used the May 7, 2015 version. As a preprocessing step, we read each email and decompose it into its fields (From, To, Subject, Content, etc.). Since the content of each email contains the entire correspondence, including any previous emails, we tried to obtain only the content of the sending message with no forwarded or replied parts. As each email could consist of multiple speeches acts, we choose to focus on the smallest meaningful entity that is a sentence and then tried to predict its speech act. Using NLTK library [21], we convert each email's content into a list of its sentences. After that, we created one list with all the sentences of all the emails in the dataset, excluding any invalid sentences (a "sentence" which consists of only a list of strings of special characters or numbers), resulting in a total of 2,683,615 sentences. Finally, we shuffled the list and select 7497 sentences and use these for labeling. We manually labeled sentences as containing either "intent" or "delivery" acts.

Also, we utilize the Parakweet Lab's Email Intent Dataset. This dataset comes from the same Enron email corpus and contains training and test data for detecting "intent" sentences in email messages. The creators of this dataset, Parakweet Lab, follow the same [18] definitions for a request, propose and commit, and define "intent" as one of these three speech acts. In total, there are 4649 labeled examples but after double checking the sentences manually, we decided that only 3828 are valid sentences (i.e. contain an "intent" or "delivery"). Of the remaining 3828, there are 1796 "intent" and 2026 "delivery" examples. Each labeled example is one sentence from an email. In the end, our dataset contains 11319 labeled sentences, 4124 sentences labeled as intent and 7195 sentences labeled as delivery. Then, using sklearn's (Pedregosa et al., 2011), we split our dataset set into 80% as training set and 20% as a testing set. We further set aside 10%

of the 80% training set as a validation set to tune the hyper-parameters of the model.

We followed the below guidance in our labeling efforts:

For the "Intent", we combine the following acts:

1. Request: ask someone else for an action, task, meeting, info or favor. Also, we consider conditional statement (e.g. If someone cannot make it at this time, let me know their names) as a request.
2. Directive: an order or command.
3. Commit: commit self to an action/task/delivery or meeting. Examples are "I'll have it ready by 2 pm" or "I'll review it later".

As for the "Delivery", we defined it as an act of sending something/information, express an opinion, to inform (FYI), or to update. As for the statement, similar to this one "please let me know if you have any questions or need additional information" which is common to end emails with it, we consider it as a "delivery" act.

2.2. Word Embeddings:

An email consists of two parts of data: structured and unstructured data. The former refers to the metadata like participants' emails ids, date/time etc. While the latter corresponds to the raw natural languages text that appears in the subject and body of the emails. In our work, we focus on the second part, more specifically on representing the content of emails using word embedding. Word embedding is a class of approaches for representing words or documents using a dense vector representation that capture something about their meaning. The main idea behind this approach is each word can be represented by means of its neighbors [23]. There is a linguistic theory behind the approach, namely the "distributional hypothesis" by [24]. Comparing with the traditional word representations, word embedding is an improvement over simpler bag-of-word model word encoding schemes like word counts and frequencies that result in large and sparse vectors (mostly 0 values) that describe documents but not the meaning of the words.

There are two main approaches for how to compute these word embeddings: Frequency based embedding and Prediction based embedding. The first approach uses matrix factorization. It starts by going through the text and counting the number of times word couples are seen close to each other (in a given window, e.g. 5 words). This information is stored in a data structure called a "co-occurrence matrix". Words vectors are built and adjusted iteratively, to minimize the (cosine) distance between words having a high probability of co-occurrence. An example of this approach is Glove (Global Vectors for Word Representation). The second approach uses a shallow feed-forward neural network (1 hidden layer). The main idea is to construct a neural network that outputs high scores for windows that occur in a large unlabeled corpus and low scores for windows where one word is replaced by a random word. When such a network is optimized via gradient descent, the derivatives backpropagate into a word embedding matrix. An example is word2vec from Google with its two variants, Continuous Bag-of-Words CBOW (given context words predict a center word) and Continuous Skip-gram SG (given a center word predict the context words).

Learning word representation requires serious computational power, time and big corpus of text. Fortunately, both Stanford and Google offer pre-train word vectors which had been trained on billon of tokens. In our work, we used GloVe pre-train word vectors1 (100 dimensions) trained on 6 billion tokens from Wikipedia 2014 corpus and word2vec pre-trained word vectors2 trained on part of Google News dataset (about 100 billion words), this model contains 300-dimensional vectors for 3 million words and phrases. In addition to the above two pre-train word vectors, we train a word2vec algorithm on the entire Enron email dataset. We used our list of all sentences found in the Enron dataset. In the preprocessing step, we convert all website's link address to the token "[LINK]" and all email addresses into the token "[EMAIL]". As for cleaning step, we followed the advice of Tomas Mikolov, one of the developers of word2vec. He suggests only very minimal text cleaning is required when learning a word embedding model. Therefore, we kept only alphabetical characters and the special char of question mark "?" and we didn't stem or lemmatize the text. After that, we used the word2vec algorithm's implementation provided by genism [26] with the following hyper-parameters (windows size = 5 and dimension size = 100), to obtain word vectors for 94,673 unique words.

3. Neural networks in NLP

Natural language processing (NLP) enables computers to perform a wide range of natural language-related tasks such as parsing, part-of-speech (POS) tagging, machine translation, dialog systems, and sentiments classification. The traditional methods which have been utilized to solve these NLP problems were traditional machine learning models such as SVM and logistic regression trained on very high dimensional and sparse features. These traditional machines learning based NLP systems relied heavily on hand-crafted features which in turn are time- consuming and often incomplete.

In the last few years, neural networks based on dense vector representations have been producing superior results on various NLP tasks [27]. An advantage of using neural networks is that they require no hand-crafted features and enable automatic feature representation learning. Reference [28] provide a comprehensive review of most deep learning methods which have been used in NLP research today. For our project, we are focusing on two deep learning architectures, convolutional neural networks (CNN) [29] and Long Short-Term Memory (LSTM) [30].

3.1. Convolutional Neural Network

Convolutional Neural Networks (CNN) have recently been shown to achieve impressive results on the practically important task of sentence categorization [31], [32], [33]. For most NLP task, CNN plays the role of feature extractor by extracting higher-level features from constituting words or n-

grams to create a useful latent semantic representation of the sentence [27]. Initially, CNN was designed to be used in image processing tasks [34]. Therefore, the input is expected to be a "2-D matrix" representing image pixels. For NLP task, instead of image pixels, the input are sentences or documents as sequences of tokens. In order to apply CNN, the input needs to be represented as a matrix where each row of the matrix corresponds to one token, typically a word. That is, each row is a vector that represents a word. Typically, these vectors are word embedding.

Instead of hand engineering our features to classify whether a sentence is an intent or a delivery, we are using a CNN as a features' extractor so it would capture a phrase such as "send me" regardless of where it happens in the sentence. Fig. 1 illustrates our CNN model architecture. The model comprises of filters layer and pooling layer. The filters layer consists of 32 filters of size 3 and their width would be equal to the embedding dimensions (100 dimensions) used to represent the sentence.

This layer performs convolutions on the input sentence matrix and generates 32 feature maps. Then the largest number from each two neighboring cells in the feature map is selected using a 1D-max pooling to produce the compress feature map. Then, these 32 compress feature maps are concatenated to form one feature matrix. This feature matrix would represent the higher-level features of the input sentence and would be passed into the LSTM layer.

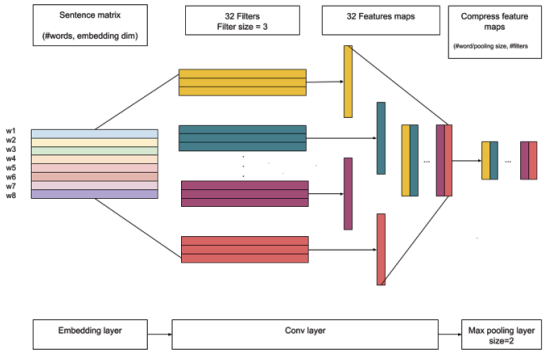

Fig. 1. Our CNN feature extractor architecure

3.2. Long Short-Term Memory network

Long Short-Term Memory network (LSTM) is a special type of Recurrent Neural Network (RNN). A recurrent neural network is a neural network that attempts to model time or any other sequence, such as language. One problem of standard RNN, as the distance between words or sequences values increase, i.e. they are separated by a large number of other words or values, modeling such dependencies will lead into the problem of vanishing gradient problem (or exploding gradient problem). LSTM is capable of overcome such shortcoming of standard RNN by learning long-term dependencies using a new structure called a memory cell. A memory cell is composed of three main gates: an input gate, a forget gate and an output gate. The weights of these gates will model the interactions between the memory cell itself and its environment. As our CNN layer learn and output the most important features of the words, the LSTM consider the meaning of a given word and remember what the previous word was. So, the model would have some "memory" of words that could occur much earlier in the sentence.

4. The proposed model architecture

Fig. 2 shows all layers of the model. The first layer is the input layer, where we are converting the input sentence from a list of tokens (i.e. words) into a list of the word index, the word index is just the location number of that word in our dictionary of unique words occurs in the corpus. Then, to solve the problem of invariant sentence length, we are padding each input sentence with zero to make all sentences length equal to the longest sentence in our corpus. The output of this layer would be a vector with the length equal to the longest sentence. We used the keras library [27] to obtain the vocabulary of word indexes and to pad the input sentences to get the same fixed length sentences.

The output of this layer is a 2d matrix, each row is a 100-D word vector represents each word in the input sentence. The third layer is a conv1D, convolutional layer. This layer applies 32 convolutional filters (filter size = 3 and activation = 'relu') on the embedding matrix input. Each filter output a feature map vector, the length of these features map vectors is equal to the length of the input sentence (i.e. the number of words). All these 32 features map vectors are concatenated to form a feature map matrix, the dimension of this matrix is (number of words in the input sentence, number of features maps). We are using a Maxpooling layer with size = 2 to compress the features matrix.

The fourth layer is an LSTM with hidden state of 100 units. In this layer, there is only one LSTM cell that is reused for as many rows in the compress feature matrix. The LSTM cell maintains a hidden state and a cell state (i.e. memory cell) within it that passes forward to the next step. But there is only 1 set of parameters being learned. Those parameters need to be able to handle all steps, conditional on the current input, hidden state, and cell state. The cell state is not an output; however, it is passed forward as an input to the next step. The hidden state is passed to the output as well as to the next step. Finally, to make the prediction, we are using a regular densely-connected NN layer with a "sigmoid" function to squeeze the output feature vector from the LSTM.

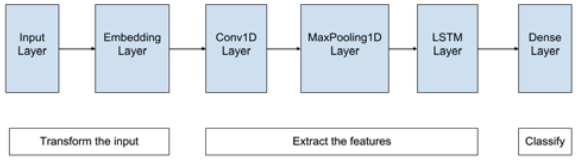

Fig. 2. The architecure of our model

5. The experiment and results

Since our approach is using a Neural network and word embeddings to classify the speech act of a sentence, we decided to consider a traditional approach to be our baseline.

The traditional approach consists of using TF-IDF to represent the input and a Support Vector Machine (SVM) to classify the input sentence into "intent" or "delivery". We used sklearn CountVectorizer and TfidfTransformor to represent the input and sklearn implementation of C-Support Vector Classification to make the classification, their implementation is based on libsvm. To obtain the best results, we used sklearn GridSearchCV to tune the model's hyper-parameters using 3 folds. Using the parallel computing option provided by sklearn, it took our Machine (MacBook Pro with 2.9 GHz Intel Core i7 processor) 278.1 mins to process the 5760 different models. The following parameters gave us the best results: Ngrm_range: (1,2), Use_idf: True, Kernel: linear, C: 1, gamma: 0.001) and Table I shows the results of three different versions of models and the baseline model. We are using F1 score to report our results, F1 score is the harmonic average of the precision and recall.

TABLE I. OUR MODELS RESULTS

Model	F_1 Intent act	F_1 Delivery act	F_1 overall
Baseline: SVC	0.88	0.79	0.85
Model1: CNN+w2v embeddings	0.86	0.77	0.83
Model2: 3channel CNN+Enron embds.	0.90	0.81	0.89
Model3: CNN+LSTM+Enron embds.	**0.91**	**0.85**	**0.89**

5.1. Our Results

Throughout our work, we tried several model architectures. The first model was a one-layer CNN followed by a densely-connected NN layer with a "sigmoid" function. In this model, we utilized several word embeddings to represent the input sentences (word2vec, Glove, and our own word embedding trained on the Enron email dataset). We also tried to let the model learn these embedding by considering them as another hyper-parameter that the model needs to learn. These are the hyper-parameters of this model: CNN filters numbers: 32, filter size: 3, pool size: 2, batch size: 16 and epochs: 10. Table II shows the results of the different variant of this model. This type of model produces the least results (accuracy and loss rate) and its performance was less than our baseline.

For the second model, we implemented a multichannel version of the first model with a 50% dropout layer between the CNN layer and the dense layer. We set a different filter size (1, 2, 3) for each channel. The intuition behind this model's architecture is that each channel could capture different features of the input. So, a channel with a filter size of one would read the sentence as 1-grams, a channel with a filter size of two would read it as bi-grams and the third channel as a tri-grams. Table III shows the results of model 2 using different word embeddings. Comparing with the model 1, the performance of this model was better, and it relatively matches the baseline performance of using traditional machine learning method. Then, we got the idea to setup a new model using different word embedding for each channel and see if we would have an improvement in the result, which turn to be useless since no improvement had been gained.

TABLE II. MODEL 1 RESULTS

Embeddings	Accuracy	Loss	Precision	Recall	F_1
Trained embeddings	81%	1.03	81%	81%	0.81
Enron embeddings	80%	0.82	82%	89%	0.81
Glove embeddings	82%	0.60	82%	83%	0.82
Word2vec embedding	82%	0.74	83%	83%	0.83

TABLE III. MODEL 3 RESULTS

Embeddings	Accuracy	Loss	Precision	Recall	F_1
Trained embeddings	80.30%	0.93	81%	80%	0.80
Enron embeddings	86.70%	0.37	87%	87%	0.87
Glove embeddings	84.36%	0.37	84%	84%	0.84
Word2vec embedding	85.07%	0.42	85%	85%	0.85
Enron, Glove, word2vec	84.93%	0.38	85%	85%	0.85

TABLE IV. MODEL 4 RESULTS

Embeddings	Accuracy	Loss	Precision	Recall	F_1
Trained embeddings	80%	0.90	81%	81%	0.81
Enron embeddings	89%	0.29	89%	89%	0.89
Glove embeddings	86%	0.30	87%	87%	0.87
Word2vec embedding	86%	0.33	86%	86%	0.86

A third and final version of our model is using a combination of CNN and LSTM layers together with the Enron word embeddings. The hyper-parameters values are the following: CNN filters numbers: 32, filter size: 3, pool size: 2 and activation function is "relu". After training the model using "Adam" optimizer for only three epochs with a batch size of 16, we got the best performance compared with the baseline model and the other two models. Not only we get better accuracy and lower the loss rate, but this model produces better precession and recall for both labels (intent and delivery). Table IV shows the results of this model.

5.2. Results Comparison to related work

Reference [18] presented an ontology of "email speech acts". Their ontology is pairs of nouns and verbs covering some of the possible speech acts associated with emails. In their work, they focus on the message level and assumed that a single email may contain several acts and each act is described by a verb-noun pair from their ontology and it's up to the annotators to determine the overall intent of the email. They also propose a system that automatically classifies emails based on its intention. The system was trained and tested on four email datasets totally (1,357) emails. The first three are subsets from the CSpace email corpus, [35] while the fourth dataset is PW CALO corpus. They used bigrams with an unweighted bag of words representation to represent the emails. They also add hand-crafted features, a total of 9602

features such as (times, POS tags and POS counts). SVM (Support Vector Machine with a linear kernel) and DT (a simple decision tree learning system) [36] produce their best results: above 80% precision and above 50% recall. In a follow-up work by [37], they show that combination of n-gram sequence features with more work on message preprocessing could reduce the classification error rates by 26.4% on average.

Reference [38] focused only on Request speech act. Their request classifier works on the message level. Their approach consists of: first, using SVM-based automated email zone classifier configured with graphic, orthographic and lexical features to classify the email content into different functional zones "email zones" then, another SVM classifier, implemented using Weka [39], would consider only small number of zones to classify whether the message contains a request or not. They used a subset of the Enron email dataset (505 email messages), released by Andrew Fiore and Jeff Heer, to train their system. Hand-crafted features such as message's length, number of capitalized word, and number of non-alpha-numeric characters, were used to represent the email messages. Their system achieves an accuracy of 83.76% and weighted F1-measure of 0.838.

Reference [40] trained several classifiers to identify speech act sentences using a variety of lexical, syntactic, and semantic features divided into three groups: "Lexical and Syntactic (LexSyn) Features", "Speech Act Clue Features", and "Semantic Features"; they also utilized speech act word lists from external resources and domain-specific semantic class features. Their system consists of four SVM classifiers, one for each speech acts (Directive, Expressive, Representative and Commisive). The classifiers were trained on 150 message board posts contained a total of 1,956 sentences, these messages were obtained from Veterinary Information Network (VIN), which is a web site (www.vin.com) for professionals in veterinary medicine. Their system performance was for sentences containing speech act: 86% Precision, 83% recall and 0.84 F-measurement, and for sentences with no speech act: 93% Precision, 95% recall, and 0.94 F1-measurement. As for each speech acts, the F-measurement were: Commisive: 48%, Directive: 86%, Expressive: 94% and Representative: 21%.

Reference [41] proposed a multilabel classification system of email messages in the Croatian language based on the following speech acts: Deliver, Amend, Commit, Remind, Suggest and Request. They used both TF (Term Frequency) and TF-IDF (Term Frequency – Inverted Document Frequency) to represent the input. As for learning, they used six different models: SVMs (Support Vector Machines), naive Bayes (NB), k-NN (k-Nearest Neighbors), Decision Stump (DS), AdaBoost (with Decision Stump as the weaker learner), and RDR (Ripple Down Rule) on three types of features extracted at three levels (message, paragraph and sentence level). Their system was trained on 1337 email messages. SVM classifier on sentence level gave the best overall performance of 0.8816 F1- measurement.

The related works are using traditional machine learning approach. This approach requires a considerable time and efforts on features engineering. Comparing that with our approach, see Table V, which required almost no time and efforts on handcrafting features. Utilizing word embedding and Neural networks, we were able to automate the entire process of feature engineering and to our knowledge, no previous research work had explored that to detect intents in emails.

6. Conclusion

Our research was to detect the intent of a sentence in email and classify it into "intent", representing a "to-do" sentence or "Delivery", representing a "to-read" sentence, according to the Email Speech Act taxonomy [18]. The main goal of our research is to design a system which require no handcrafted features and still achieve good results. In this work, we were able to achieve that using word embeddings to represent the input sentences and then using a model consists of Convolutional Neural Network (CNN) and Long Short-Term Memory network (LSTM) to extract the features to classify the intent of the sentence. We found that our model is able to detect the intent of the sentence with an accuracy of 89% and a loss rate of 0.29. The precession and recall, of the "intent" act, are 90% and 93% respectively. While the precession and recall, of the "delivery" act, are 87% and 82% respectively. These results confirm our hypothesis that using word embeddings and neural network model, outperforms the traditional approach of handcrafting features.

TABLE V. COMPARISON OF OUR WORK WITH RELATED WORKS

Model	Level	Hand Crafted features	Speech acts	Precision	Recall	F_1
Cohen et al., 2004	msg	Yes	Request	Above 80%	Above 50%	0.69
			Proposal			0.44
			Delivery			0.80
			Commit			0.47
			Directive			0.78
			Commissive			0.85
			Meet			0.72
Lampert et al., 2010	msg	Yes	Request	84.90 %	83.7%	0.843
			Non-request	82.50 %	83.9%	0.832
Qadir & Riloff, 2011	Sent.	Yes	Commissive	63%	39%	0.48
			Directive	87%	85%	0.86
			Expressive	97%	91%	0.94
			Represent	32%	16%	0.21
Franovic & Šnajder, 2012	Sent.	Yes	Deliver	-	-	0.885
			Amend	-	-	0.723
			Commit	-	-	0.789
			Remind	-	-	0.695
			Suggest	-	-	0.697
			Request	-	-	0.721
Ours	Sent.	No	Intent-To do	90%	93%	0.91
			Delivery-To read	87%	82%	0.85

7. References

[1] D. Crocker, (1982). Standard for the format of ARPA Internet text messages.

[2] [Online]. Available:http://www.radicati.com.

[3] D. Fisher, A. J. Brush, E. Gleave, & M. A. Smith, "Revisiting Whittaker & Sidner's email overload ten years later", in Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (pp. 309-312), ACM, 2006.

[4] C. Grevet, D. Choi, D. Kumar, & E. Gilbert, «Overload is overloaded: email in the age of Gmail", in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 793-802), ACM, (2014, April).

[5] F. Kooti, L. M. Aiello, M. Grbovic, K. Lerman, & A. Mantrach, "Evolution of conversations in the age of email overload", in Proceedings of the 24th International Conference on World Wide Web (pp. 603-613). International World Wide Web Conferences Steering Committee, (2015, May).

[6] S. Whittaker, V. Bellotti, & J. Gwizdka, "Email in personal information management". Communications of the ACM, 49(1), 68-73, (2006).

[7] V. Bellotti, N. Ducheneaut, M. Howard, I. Smith, & R. E. Grinter, "Quality versus quantity: E-mail-centric task management and its relation with overload", Human-computer interaction, 20(1), 89-138, (2005).

[8] A. Kidd, "The marks are on the knowledge worker", in Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 186-191). ACM, (1994, April).

[9] S. Whittaker, V. Bellotti, & J. Cwizdka, "CD Everything through Email", Personal information management, 167, (2007).

[10] Y. Koren, E. Liberty, Y. Maarek, & R. Sandler, "Automatically tagging email by leveraging other users' folders", in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 913-921). ACM, (2011, August).

[11] M. Grbovic, G. Halawi, Z. Karnin, & Y. Maarek, "How many folders do you really need?: Classifying email into a handful of categories", in Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (pp. 869-878). ACM, (2014, November).

[12] D. Aberdeen, O. Pacovsky, & A. Slater, "The learning behind Gmail priority inbox", in LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds, (2010, December).

[13] J. D. Brutlag & C. Meek, "Challenges of the email domain for text classification", in ICML (pp. 103-110), (2000, June).

[14] M. J. Pazzani, "Representation of electronic mail filtering profiles: a user study", in Proceedings of the 5th international conference on Intelligent user interfaces (pp. 202-206). ACM, (2000, January).

[15] G. Cselle, K. Albrecht & R. Wattenhofer, "BuzzTrack: topic detection and tracking in email", in Proceedings of the 12th international conference on Intelligent user interfaces (pp. 190-197). ACM, (2007, January).

[16] L. A. Dabbish & R. E. Kraut, "Email overload at work: an analysis of factors associated with email strain", in Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (pp. 431-440). ACM, (2006, November).

[17] S. Whittaker & C. Sidner, "Email overload: exploring personal information management of email", in *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 276-283). ACM, (1996, April).

[18] W. W. Cohen, V. R. Carvalho & T. M. Mitchell, "Learning to Classify Email into``Speech Acts"", in Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, (2004).

[19] J. R. Searle, "Speech acts", Cambridge, UK: Cambridge University, (1969).

[20] L. John, Austin, "How to do things with words" (1962).

[21] W. E. Mackay, "Diversity in the use of electronic mail: A preliminary inquiry", ACM Transactions on Information Systems (TOIS), 6(4), 380-397, (1988).

[22] S. Bird & E. Loper, "NLTK: the natural language toolkit" in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions* (p. 31). Association for Computational Linguistics, (2004, July).

[23] J. R. Firth, "Applications of General Linguistics. Transactions of the Philological Society", 56: 1–14. doi:10.1111/j.1467-968X.1957.tb00568.x, (1957),

[24] Z. S. Harris, "Distributional structure. Word", 10 (2-3): 146–162. Reprinted in Fodor, J. A and Katz, JJ (eds.), Readings in the Philosophy of Language, (1954).

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado & J. Dean, J, "Distributed representations of words and phrases and their compositionality", in Advances in neural information processing systems (pp. 3111-3119), (2013).

[26] R. Rehurek & P. Sojka, "Software framework for topic modelling with large corpora", in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, (2010).

[27] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu & P. Kuksa, "Natural language processing (almost) from scratch", Journal of Machine Learning Research, 12(Aug), 2493-2537, (2011).

[28] T. Young, D. Hazarika, S. Poria, & E. Cambria, "Recent trends in deep learning based natural language processing", arXiv preprint arXiv:1708.02709, (2017).

[29] Y. LeCun, Y. Bengio & G. Hinton, "Deep learning", nature, 521(7553), 436, (2015).

[30] S. Hochreiter & J. Schmidhuber, "Long short-term memory", Neural computation, 9(8), 1735-1780, (1997).

[31] Y. Kim, "Convolutional neural networks for sentence classification", arXiv preprint arXiv:1408.5882, (2014).

[32] N. Kalchbrenner, E. Grefenstette, & P. Blunsom, 'A convolutional neural network for modelling sentences", arXiv preprint arXiv:1404.2188, (2014).

[33] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing.", J. Artif. Intell. Res.(JAIR), 57, 345-420, (2016).

[34] A. Krizhevsky, I. Sutskever & G. E. Hinton, «Imagenet classification with deep convolutional neural networks", in Advances in neural information processing systems (pp. 1097-1105), (2012).

[35] R. Kraut, S. Fussell, F. Lerch & A. Espinosa, "Coordination in teams: Evidence from a simulated management game", (2005).

[36] R. E. Schapire & Y. Singer, 'Improved boosting algorithms using confidence-rated predictions", Machine learning, 37(3), 297-336, (1999).

[37] V. R. Carvalho & W. W. Cohen, "Improving email speech acts analysis via n-gram selection", in Proceedings of the HLT-NAACL 2006 Workshop on Analyzing Conversations in Text and Speech (pp. 35-41). Association for Computational Linguistics, (2006, June).

[38] A. Lampert, R. Dale & C. Paris, C "Detecting emails containing requests for action", in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (pp. 984-992). Association for Computational Linguistics, (2010, June).

[39] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann & I. H. Witten, «The WEKA data mining software: an update", ACM SIGKDD explorations newsletter, 11(1), 10-18, (2009).

[40] A. Qadir & E. Riloff, "Classifying sentences as speech acts in message board posts", in Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 748-758). Association for Computational Linguistics, (2011, July).

[41] T. Franović & J. Šnajder, "Speech Act Based Classification of Email Messages in Croatian Language", in Proceedings of the Eighth Language Technologies Conference (p. 69), (2012, December).