Spring 2021

Detecting the Intent of Email Using Embeddings, Deep Learning and Transfer Learning

Zaid Alibadi

Recommended Citation

Alibadi, Z.(2021). *Detecting the Intent of Email Using Embeddings, Deep Learning and Transfer Learning.* (Doctoral dissertation). Retrieved from https://scholarcommons.sc.edu/etd/6384

DETECTING THE INTENT OF EMAIL USING EMBEDDINGS, DEEP LEARNING AND TRANSFER LEARNING

by

Zaid Alibadi

Bachelor of Science
Al-Nahrain University, 2006

Master of Science
University of South Carolina, 2017

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Computer Science

College of Engineering and Computing

University of South Carolina

2021

Accepted by:

Jose M. Vidal, Major Professor

John R. Rose, Committee Member

Csilla Farkas, Committee Member

Jijun Tang, Committee Member

Pelin Pekgun, Committee Member

Tracey L. Weldon, Interim Vice Provost and Dean of the Graduate School

DEDICATION

To my late father; Haider Alibadi.

To my late grandfather; Jaleel Soura Meri.

Gone But Not Forgotten.

ACKNOWLEDGEMENTS

I express a special thanks to my academic advisor, Dr. Jose M. Vidal. For your valuable guidance and ideas to present this dissertation as best as possible, I show my sincere gratitude and appreciation.

A special thanks to Dr. Manpreet Gill, Dr. Ramkumar Janakiraman, Dr. Pelin Pekgun and Dr. Chen Zhou for giving me the opportunity to be part of their research.

Grateful thanks to the Graduate Director Dr. Jijun Tang, the Graduate Program Coordinator Ms. Sri Satti, and the Sponsored Students Coordinator Ms. Joanna Zietara for helping me in all the administrative matters that accompanied the years of study and research.

A sincere gratitude to the Higher Committee for Education Development (HCED) in Iraq for their generous financial support.

Finally, to my beautiful wife, Marwah, wonderful family, and everyone who helped and supported me, I say "thank you".

ABSTRACT

Throughout the years' several strategies and tools were proposed and developed to help the users cope with the problem of email overload, but each of these solutions had its own limitations and, in some cases, contribute to further problems. One major theme that encapsulates many of these solutions is automatically classifying emails into predefined categories (ex: Finance, Sport, Promotion, etc.) then move/tag the incoming email to that particular category. In general, these solutions have two main limitations: 1) they need to adapt to changing user's behavior. 2) they require handcrafted features engineering which in turn need a lot of time, effort, and domain knowledge to produce acceptable performance.

This dissertation aims to explore the email phenomenon and provide a scalable solution that addresses the above limitations. Our proposed system requires no handcrafted features engineering and utilizes the Speech Act Theory to design a classification system that detects whether an email required an action (i.e., to do) or no action (i.e., to read). We can automate both the features extraction and the classification phases by using our own 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, followed by an LSTM layer to consider the meaning of a given feature (trigrams) concerning some "memory" of words that could occur much earlier in the email. Our system detects the email intent with 89% accuracy outperforming other related works.

In developing this system, we followed the concept of Occam's razor (i.e., law of parsimony). It is a problem-solving principle stating that entities should not be multiplied without necessity. Chapter four present our efforts to simplify the above-proposed model by dropping the use of the CNN layer and showing that fine-tuning a pre-trained Language Model on the Enron email dataset can achieve comparable results. To the best of our knowledge, this is the first attempt of using transfer learning to develop a deep learning model in the email domain.

Finally, we showed that we could even drop the LSTM layer by representing each email's sentences using contextual word/sentence embeddings. Our experimental results using three different types of embeddings: context-free word embeddings (word2vec and GloVe), contextual word embeddings (ELMo and BERT), and sentence embeddings (DAN-based Universal Sentence Encoder and Transformer-based Universal Sentence Encoder) suggest that using ELMo embeddings produce the best result. We achieved an accuracy of 90.10%, comparing with word2vec (82.02%), BERT (58.08%), DAN-based USE (86.66%), and Transformer-based USE (88.16%).

TABLE OF CONTENTS

Dedication iii

Acknowledgements iv

Abstract v

List of Tables viii

List of Figures ix

Chapter 1: Introduction 1

Chapter 2: Why Machine Learning 15

Chapter 3: Email Overload 30

Chapter 4: Transfer Learning in the Email Domain 57

Chapter 5: Investigating the Effects of Contextual Representation in Email 69

Chapter 6: Conclusion 80

References 82

LIST OF TABLES

Table 3.1: Evaluation Results 52

Table 3.2: Comparison of our work with related works 56

Table 4.1: Removed email addresses....................63

Table 4.2: Training dataset shape 64

Table 4.3: Baseline model results 65

Table 4.4: SVC model result....................66

Table 4.5: First re-trained model results 66

Table 4.6: Best retrained model results....................67

Table 4.7: Baseline vs. re-trained LM model results 68

Table 5.1: First model using different context-free word embeddings....................76

Table 5.2: Five models using different pre-trained embeddings 78

LIST OF FIGURES

Figure 3.1: Our CNN feature extractor architecture 45

Figure 3.2: The architecture of our model 48

Figure 4.1: Our LM main steps 62

Figure 5.1: Frequency based embeddings 70

Figure 5.2: Prediction based embeddings 71

CHAPTER 1

INTRODUCTION

On his voyage from London to Philadelphia in 1726, Benjamin Franklin conceived the notion of a notebook in which he would record systematically his efforts at self-improvement. In order to achieve this goal, he describes thirteen virtue. The third was Order; "Let all your things have their places; let each part of your business have its time". Years later, in his autobiography (1790), he described that this particular virtue gave him the most trouble: "Order … with regard to places for things, paper, etc., I found extremely [sic] difficult to acquire." Two hundred years later, order continues to be a difficult goal, especially in the emails domain.

In this research, we are aiming at studying the Email Phenomenon. Specifically, we trying to answer the six following questions: How do we study email? How do we use email? How do we process email? How do we reply to email? How do we organize email? and How do we recognize an important email?

Next, we describe the main problem facing email users, in particular, we are focusing on the problem of Email Overload. We also try to cover the available solutions to help the user manage their email inboxes and their limitations. Then, we explore how could machine learning techniques solve this problem and introduce our proposed solution and list its advantages over the current solutions. Later, we introduce the concept of transfer learning, specifically in the domain of Natural Language Processing (NLP), by giving more details into the application of transfer learning in the NLP domain and Language Modeling.

Finally, we present our model's architecture and compare the results with other related works.

1.1. How Do We Study Email?

Since the inception of email as a communication tool, many researchers try to study the phenomena of emails. Overall, we could divide these efforts into two main camps: The first focus on the users, While the second camp focus on email's usage. An example of the first camp would be the Collaborative User Experience research group in IBM which investigating emails through user's observations and interviews, design mockups, prototype implementations, and user evaluations (Rohall et al., 2004).

(Whittaker et al., 2005) have a different classification to how researchers study email. They observe three such classes: The first is "empirical studies of email usage". The second is "novel system designs", where different techniques that had been utilized to tackle different problems, and the third is based on theory; Depending on how users view emails, different theories can be applying. Almost all of these classes suffer from the same limitations: limited number of participants, organizational homogeneousness of participants, reactive methods (surveys, interviews, experiments), and lack of longitudinal perspective (Kalman and Ravid, 2014).

Regardless of the classification of these studies, we found one common goal; to obtains observations regarding the Five Hows; How users utilize emails? How users process their incoming emails? How users reply to emails? How users organize their emails? And finally, How users define important emails? In the following sections we will dive into each "How". But first let review couple of these studies and highlight their main methodologies.

(Dabbish et al., 2004) argue in favor of statistical analysis of surveys rather than trying to analysis email archives using machine learning approach. Their reasons were: 1) machine learning approach give little insight for user behaviors and actions on emails while surveys provide us with general model on user's behaviors and 2) machine learning approach tend to be on small sample size because these approaches are often intrusive while survey would cover much bigger samples, increasing the generalization of the model. (Venolia et al., 2001) tried to take advantage of both approaches by studying both the user's behavior and email's content. They used interviews and survey to study the users, in addition to analysis of message archives.

1.2. How Do We Use Email?

Email is one of the most useful communication tools over the Internet. Since the 2000's, email technology was significantly expanded, and its daily use becomes a necessity to manage people's work, business and interpersonal relations. Some consider email not as a product but as a protocol that lets people send and receive information (Vishnu et al., 2013). Others, (Ducheneaut and Bellotti, 2001), see email more like a habitat than an application. Overall, Email is an effective knowledge management tool which conveniently enables fast and accurate communication. It is used for a wide range of tasks such as information management and for coordination and collaboration in organizations.

(Mackay, 1988) is among the first who study the uses of emails. He identifies three major forms of work management used in emails: information management, time management and task management. (Whittaker and Sidner, 1996) shared some of Mackay's findings and discuss three main functions of email in their studies of 20 user's inboxes: task management, personal achieving, and asynchronous communications. Other researchers, (Dabbish et al., 2004), try to examine the use of email in the work environment, they observed six main purposes that email serves in an organizational context: Action request, status update, reminders, information requests and responses, scheduling requests and responses, and social content.

Rather than studying the purpose of emails, (Muller and Gruen, 2002) try to focus on the situation in which email itself is the object of the collaboration. They studied how teams shared a single mailbox to conduct a work operation (i.e., customer care, educational institution, published office address). Other examples of reuse are comment fields as instructions, mailboxes as identity statements, and folders as action-requests or status indicators.

As for differences between work emails and personal emails, (Grevet et al., 2014) notice that work's 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).

1.3. How Do We Process Emails?

When it's come to how users handling their inboxes, two early studies tried to explain it; (Mackay, 1988) divides users into two different categories: 1) prioritizes who focusing on managing their emails as they arrived and 2) archivers who focusing on archiving information for later user. The second study, (Whittaker and Sidner, 1996), found that users fell into one of three categories: 1) Frequent fliers who constantly cleaning their inbox, 2) spring cleaners who cleaned their inbox every few months and 3) no fliers who keep all their emails in the inbox folder and use a search tool to manage it.

Another classification by (Gwizdka, 2004) focus on how users managed the tasks' emails and he categorize them into: cleaners and keepers. While cleaners tend to set specific times to read their emails so incoming emails wont interrupt other activities, keepers would keep reading emails all the times and let incoming emails interrupt other activities. He concludes that cleaners seem to have more control over their email behavior. In the following two sections, we try to cover the most common activities applied by users on their emails and the strategies they follow to organize their email's inboxes.

1.3.1. Users Activities

According to research conducted by McKinsey in 2012, reading and answering email counts for 28% of the average employee workday (McKinsey Global Institute & International Data Corp, 2012). As emails' users spend more time processing and managing their email, several activates done by users could be observed. A recent quantitative study, done by (Castro et al., 2016) for 110,000 Yahoo active users, found that 98% of users read their incoming emails, 29% reply, 28% forward and 18% delete an email, more specifically, 89.5% of these delete actions are not preceded by any read operation. They make a distinguish between two delete behaviors: delete-with-read and delete-without-read. The first brings some level of interest: its header is intriguing enough for the user to open (and most probably read) it. The second indicates a lesser level of interest: either the header brings sufficient information that the user does not need to read more, or more frequently, the header is sufficient for the user to decide that he or she has no interest whatsoever in that email, which is quite common with bulk mailing.

As for the user's time management for these emails activities, it turns out that they divide their time unequally. Based on video studies done by (Bellotti et al., 2005), they found the percentage of time people spending on: composing emails (54%), reading (23%), filing (10%), scanning mailbox (6%), deleting (2%), searching folders (2%), managing attachments (2%). Another area of emails that users spend considerable time on is email attachment. Few studies have been made to analysis the use of attachments in email. (Hailpern et al., 2014) conducted two studies (813 participants) to understand the state and limitations of attachments. They conclude that most attachments are large sized files, taking up a considerable amount of physical storage on local computers and on exchange servers, too. As for attachment's access, users are opening a large percentage of documents.

1.3.2. User's Strategies

Not all users manage their emails in the same way. Many factors determine the best strategy to match the end goal of the user. (Gwizdka, 2004) study these factors and he proposes four cognitive abilities which have a possible effect on email processing:

1. Flexibility of closure: extracting email or email attributes from a distracting background of other emails
2. Speed of closure: recreation of structure or relationships between a group of email messages
3. Visual memory: ability to remember the configuration and location of an email, and
4. Working memory: a temporary store for recently activated items of information.

These four cognitive abilities were measured using Factor-Referenced Kit of Tests (Ekstrom, 1976). Based on these four factors, Gwizdka observes four strategies utilized by users to deal with emails of future tasks or events:

1. Immediate processing,

2. Limiting (e.g., the active screen should contain all the important emails otherwise, ignore),
3. Encoding additional information (e.g., by adding flags to messages), and
4. Accumulation.

A different approach is to choose either a single pass or a multi-pass strategy (Neustaedter et al., 2005a). In a single pass, user prefers simplicity over the ability to quickly find important emails, while the multi-pass strategy was effective in finding important emails quickly, although this strategy is more time consuming. They also found that users who followed the multi-pass strategy would first try to find emails that could quickly be deleted or got rid of. Because it's easiest for users to handle emails of little importance (they could quickly delete them or file them) and often once the unimportant emails were gone, it was easier to find the important emails. (Bälter and Sidner, 2002) observed that users primarily following the multi-pass strategy, as they scanned the inbox a mean of 2.3 times to decide which email need attention first. Moreover, (Venolia et al., 2001) found that 70% of their interviewees processed emails out of order.

1.4. How Do We Reply To An Email?

On average, user send 31% as many emails as they receive, which implies many emails don't required a reply (Dabbish et al., 2005). A more recent study, (Castro et al., 2016), give much lower percentage, only 2% of incoming emails received a reply. (Kooti et al., 2016) found a correlation between the number of received emails and the frequencies of users' replies. As user receive more emails, he/she tend to send less replies. In their study of more than 2 million users exchanging 16 billion emails over several months, the percentage of emails which received a reply decrease from 25% (on a day with low load of incoming emails) to less than 5% at high load day (about 100 emails a day). They observed other factors affect the likelihood and the promptness of replying to an email:

- The day and time the message was received: Email users are more active during the day than nighttime, and on workdays rather than the weekend. Emails received in the morning get substantially longer replies than those received in the afternoon and evening.
- The device used: Replies sent from phones are the fastest, followed by emails sent from tablets, and finally replies from desktops.
- The number of attachments in the email: Replies to emails with attachments are much slower (median of 56 minutes) than replies to emails without any attachment (median of 32 minutes). Emails with an attachment, get longer replies (median of 47 words) than emails without attachments (median of 33 words).

Another reason for not replying for higher percentage of incoming emails is 90% of emails are machine-generated, therefore, these emails do not expect a reply (Castro et al., 2016). Other observed factors that affect the probabilities of receiving replies: Emails from people within the same work organization not only have high probability to receive a reply but user typically responded more quickly (Tyler and Tang, 2003). They also observed that users typically responded more quickly to emails from people with whom they have a history of quick communication. Incoming emails belong to a continuing conversation thread increase the probability to receive a reply. While the frequency of communication with the sender has no major effect on the likelihood of a reply (Dabbish et al., 2004).

Another interesting finding by (Dabbish et al., 2004) is perceived importance is only weakly correlated with responding; people are only 8% more likely to respond to important emails. They found the likelihood of replying to an email is affected by sender's characteristics more than email's content except for two cases: first, if the content is social, in this case, its %23 more likely to have respond. Second, emails with information request are more likely to have a reply. They also observe that email with only one recipient more likely to have a reply. A recent study by (Castro et al., 2016) confirmed that as they found the reply's ratio is higher when the number of recipients is smaller (indicating a personal correspondence).

As for how the users handle the emails which received a reply, they utilize two different methods: Rapid-Response; composed quickly in a "fire-and-forget" fashion, and Extended-Response, which requiring extra work, possibly with a need to make notes so that ideas on how to handle the email is not forgotten (Bellotti et al., 2005).

Another important aspect is the reply time, (Tyler and Tang, 2003) found that users gave a special attention to how and when they replied to emails. The time to respond expresses a "responsiveness image" that depends on the social relationship with the sender. (Kooti et al., 2016) found that 90% of the replies happen within a day of receiving the message, and the most likely reply time is just two minutes. Also, half of the replies are within 47 minutes of receiving the message. Replies become faster as the conversation progresses, but the last reply is much slower than the previous replies.

1.5. How Do We Organize Emails?

(Whittaker et al., 2006) observed two main strategies users would follow to facilitate email's retrieval: 1) organizing their mailbox using folders and 2) utilizing sorting and searching. As for which one dominate the usage, (Whittaker et al., 2011) conclude, based on their large-scale quantitative study of how people retrieved email, that sorting and searching dominates. Other researchers, such as (Teevan et al., 2004), argue that emails' users utilize a mix of the above two strategies to retrieve an email.

1.5.1. Foldering

Several reasons in favor of foldering could be observed:

- Declutter the inbox into a relatively small set of folders each containing multiple emails related to same tasks (Whittaker et al., 2007). In this context foldering is used to organize ongoing tasks/activates or archive completed tasks found in emails.
- Another representation of "to-do" item (Minkov et al., 2008).
- For archival purposes (Kushmerick et al., 2006), as users tend to file emails containing attachments, web links, or presentations (Bellotti et al., 2005).
- Preserved the context of the communications and activates history rather than just a method for finding emails later (Ducheneaut and Bellotti, 2001).

For all the above reasons, email's users spend a considerable amount of time and effort to organizing their emails (Gwizdka and Chignell, 2004). (Fisher et al., 2006) comparison of email archives from 1996 and 2006 shows that archive size and number of folders have been increased dramatically but the average inbox size have remained the same.

1.5.2. Threads

Both (Wattenberg et al., 2005) and (Whittaker, 1996) proposed threaded views as a way to help users manage email. While folders require the users to manually create them and assign each email individually to its appropriate folder(s), thread is presented as an automated method to clustering individual emails related to each other by the reply function in email (Kerr, 2003). A thread-based view of emails can use space more efficiently while at the same time giving the user additional context when focused on an individual message (Tang et al., 2008). A good example would be Gmail and how it uses a clean and consistent model of threads, rather than individual messages to organize emails.

On the other hand, there are some limitations for the use of threading, grouping both incoming and outgoing email in the same threads would possibly confuse the user and violate his/her model of the "Sent" folder as a separate folder from the inbox (Tang et al., 2008). (Cselle et al., 2007) identify two other difficulties: Thread drift and Topic drift. Thread drift means that a topic contains several threads while Topic drift means that the same thread contains information about different topics (i.e., the use the "Reply To" button instead of "New Mail").

1.6. How Do We Recognize an Important Email?

(Spira and Goldes, 2007) report that a typical worker receives 200 non-spam email on daily bases, a more recent report by Radicati.com estimates the number of business emails sent and received per user per day to average 126 messages in 2019. As managers' responsibilities broaden, these numbers would increase, e.g., NSF program managers report 500 to 1000 non-spam emails per day (Yoo et al., 2011).

As the user's inbox is flooded with tasks, responsibilities, and deadlines, all compete for user's attention daily, their productivity would negatively be affected. They either keep reading email streams loaded with low-priority announcements and acknowledgments, or alternatively by not reading email frequently and risk the chance of ignoring high-priority urgent emails. Therefore, highlighting important emails would save email's users a lot of time and efforts and decrease the chances of missing out an important task need to be done. (Faulring et al., 2010) argue that the order in which emails are handled can significantly affect the efficiency of the strategy, since performing similar tasks together reduces the overhead of switching between different types of tasks.

Importance ranking for emails is hard problem as users disagree on what is important and requiring a high degree of personalization (Aberdeen et al., 2010). Not only the perceived important of emails differ from user to user, but it differs for the same user based on the time of accessing his inbox or the project that he/she currently working on. One solution to identify the importance of an email by flagging it as important but this solution found to be ignored by most users (Whittaker and Sidner, 1996). The reason could be this field requires to be filled manually by the user which is a tedious work and it reflect the importance from the sender perspective not the receiver (Dabbish et al., 2004).

So, what's make an email important? (Mackay, 1988) see the importance of a message has as much to do with the current state of the user as the content of the email. Several studies have been conducted to observe the factors which affect perceived important of emails. A certain pattern could be observed reviewing the literature:

1. Social based factors: Social information is vital for determining the importance of an email, such as:
   - Sender (Venolia et al., 2001) (Dabbish et al., 2005),
   - The history of the communication between the sender of an email and the receiver. Higher communication frequency with the sender increases the importance (Dabbish et al., 2005),

- Sender within the same organization as the receiver or from outside the organization (Neustaedter et al., 2005b). For example, emails from someone with a close personal relationship (e.g., close colleagues, direct managers) and emails from new social contacts working on similar projects were typically quite important.
- Other social metrics are number of recipients (Venolia et al., 2001) (Balter and Sidner, 2002) (Dabbish et al., 2003) (Dabbish et al., 2005) as higher number of recipients decreased email's importance, the number of times user reply to a person's emails, the number of emails that user receive from someone that get marked read, or the number of times someone replies to user's emails (Neustaedter et al., 2005b).

2. Content based factors: Through collecting data on 121 users using a web-based survey, (Dabbish et al., 2005) found that message content plays a role in user's perception in importance of an email. Emails of action/meeting request/response had more importance than social contents email. (Venolia et al., 2001) confirm that the nature of the email would affect its importance. Also, whether the email is a reply have an effect, too. Replies were important as they often contained a solution to a problem sent by the recipient.
3. Time based factors: (Neustaedter et al., 2005a) argue that the importance of email is largely determined by its time attributes. Emails related to events in user's calendar for the day were judged important, regardless of arrival date (Balter and Sidner, 2002).

Looking for this issue from different angle, it may be useful to see what characteristics of an email make it more likely to be ignore or deleted by the user. Less-recent emails, news-related items, or email from people with a lesser relationship (e.g., someone for whom the user did not typically send replies to) were typically not as important to users (Venolia et al., 2001). Carbon copies were judged as less interesting than other messages, but not always (Balter and Sidner, 2002).

CHAPTER 2

WHY MACHINE LEARNING?

Several researchers propose using machine learning in modeling the change in the user's behavior, personalized filtering emails and email's content summery to help the user in allocating attention and deciding actions. (Mackay,1988) was one of the earliest researchers to suggest that intelligent information retrieval techniques may prove practical for classifying and retrieving emails. He advised that email clients designers should focusing more on tools that accommodate the diverse use of emails rather than searching for an optimal set of functions. As for the problem of allocating important email in the age of email overload, (Whittaker et al., 2007) argue that analyzing email's content and headers might both help users to allocate attention to important email and to decide what action to apply to an email. (Faulring et al., 2010) conclude, based on their work on RADAR, that adding AI technologies to collaborative systems can benefit users. They found that users who received AI assistance performed 37% better compared with users who did not. They emphasis that predictability and understandability are sub goals of the ultimate goal: usable systems.

Although machine learning solutions do not require much work from the users, trust is the main concern for users (Pazzani, 2000) (Whittaker and Sidner, 1996). To overcome this obstacle, more freedom to adjust some of the setup of the tools may help. MailCat (Segal and Kephart, 1999) avoids the risk of the user not knowing where content has gone by presenting its best three guesses about where to place the message and i-ems (Crawford et al., 2002) intelligent classification application allows user intervention to update the suggested folder.

In the following sections, we will start by explaining how the machine read and understand email, then we present some of the previous work in using machine learning techniques in the domain of email systems. We divide them into four main sections: prioritization, categorization, visualization and social network between sender and receiver.

2.1. How Machine Understand Email?

An email consists of two type 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 text data that appears in the subject and body of the emails. Several natural language techniques have been utilized by researchers to represent the structure and unstructured data of emails. Based on the reviewed literature, we observed the below four different representations:

1. Graph: (Minkov et al., 2008) represent email's content, social network (information about senders and receivers), time information and activities, as a structured dataset (a graph). They model email as a heterogeneous graph, where nodes represent the message, person, email address, date and terms. For example, a message can be linked to a person node with a relation of sent-to, sent from etc.
2. Lexical: The bag-of-words (BOW) model is a simplifying representation used in natural language processing and information retrieval (IR). In this representation, a text (i.e. email content) is represented as a bag (multiset) of its words, ignoring grammar and even word order but keeping multiplicity. The final output is a one-

line text file, per email, containing the number of occurrences of each known token (e.g., word). (Qadir et al., 2016) experiment with lexical representation of the email content. They use the bag of words (BOW) from email body and subject, after Penn Tree Bank (PTB) style tokenization. They also experiment with a syntactic representation using heuristics on the output of a PTB constituent parser (Quirk et al., 2012) to identify Nouns (N) and Verb Phrases (VP) in email body and subject. (Graus et al., 2014) also use bag of words, while (Carvalho and Cohen, 2007) used tf-idf vectors to represent the content of an email.

3. Topic modeling: The main assumption behind this approach is that each document (i.e., email) was generated by a single activity-specific distribution over words. Therefore, emails about the same activity will use similar words, while emails about different activities will use different words. For this purpose, Latent concept models would be useful (Dredze et al., 2008). As these models treat documents as having an underlying latent semantic structure, which may be inferred from word-document co-occurrences (relates words to concepts and concepts to documents). Two widely used latent concept models are: Latent Semantic Analysis/Indexing (LSA/LSI) and Latent Dirichlet Allocation (LDA). LSI introduced by (Deerwester et al., 1990), represents words and documents as points in Euclidean space which could be used to determine emails similarity. Unlike traditional bag-of-words models and similarity metrics that are based on how many words two documents share, LSI projects documents onto a reduced-dimensionality subspace, and computes similarity as the distance between two document vectors in the reduced subspace. The intent of subspace projection is to capture concepts inherent in

similar documents, such that each dimension in the subspace corresponds to a different concept in the document corpus (Dredze and Lau, 2006). The second method which used in document's clustering is (LDA). In this approach, concepts are distributions over words and weights are mixing probabilities representing distributions over topics. LDA models learn probability distributions of words as latent topics in a corpus. This method treats each document as a finite mixture over an underlying set of topics, where each topic is characterized as a distribution over words. Each email has a different distribution over these topics: an email about going on vacation might give equal probability to both "hotel rooms" and "flights".

4. Embeddings: for this approach, word representations are typically learned from large collection of documents in a sliding window-fashion by updating the central word in the window such that it is capable of accurately predicting the surrounding words in the same window. Typically, a neural language model learns the probability distribution of next word given a fixed number of preceding words which act as the context. A recently proposed scalable Continuous Bag-of-Words (CBOW) and scalable Continuous Skip-gram (SG) model (Mikolov et al., 2013) for learning word representations have shown promising results in capturing both syntactic and semantic word relationships in large news articles data. Their scalable open-source software is available online (code.google.com/p/word2vec). (Kooti et al., 2015) use a neural language model, known as paragraph2vec (Le and Mikolov, 2014) to represent each email as a real-valued low dimensional vector. Introducing low dimensional embedding of words by neural networks take advantage of word

order in documents and state the assumption that closer words in the word sequence are statistically more dependent.

2.2. Prioritization As a Solution

Machine learning techniques could be utilized to extracted important information automatically from email-header information, such as frequency, longevity of communication, and likelihood of response (Wattenberg et al., 2005) (Whittaker, 2004). Then, such information would help to automatically produce prioritization rules. For this end and based on the reviewed literature, two approaches were observed:

First, analyzing user's triage behavior on previous emails to determine the priority ranking for new emails. Displaying an importance rating for an email beside other information such as subject, sender, etc., would be useful particularly for sorting emails by importance or comparing the relative importance of emails (Alibadi et al., 2014).

The second approach argued by (Franovic and Šnajder, 2012) is the alternative to importance-based filtering is content-based classification, which labels each message based on its content, leaving it to the user to decide on the importance of the message. An example would be analyzing previous emails to predict whether the user will perform an action (i.e., reply, delete, etc.) on a new incoming email. The reason behind this approach is the task of predicting the importance of the email is not only difficult but probably not useful per se, and the users may prefer to find "actionable email" instead (Castro et al., 2016). Several researchers follow this approach; (Carvalho and Cohen, 2005) use machine learning to identify email requiring an action, (Ayodele and Zhou, 2009) propose to solve the problem of email prioritization and overload by determining if email received needs reply, and (Cselle et al., 2007) propose "reply expected" indicator which marks topics in which the newest email was addressed to the user and has not yet been replied to. On the other hand, predicting if new email requires no action (or no immediate action) could save valuable time for the user and let him/her focus on important emails instead. In (Dabbish et al., 2004) study, 64% of the message didn't required a reply. Such a high percentage on no reply emails suggest an intelligent agent could aid the user in prioritizing emails for view.

2.3. Categorization As a Solution

Another technique to help the user to cope with the email overload is to assist him/her in decluttering the inbox by categorizing the incoming emails. (Dredze and Lau, 2006) state that an email's user would benefit from a system that can identify, given a new message, which activity/task it belongs to. They also noted that the email activity classification problem is an incremental learning problem; the set of class labels can (and will) change over time as new activities are created by the user.

One way to do that is utilizing machine learning techniques to analyze emails headers and content and make recommendations to users about how they might categorize incoming emails into folders. To achieve this goal, the system should satisfy several conditions, (Balter and Sidner, 2002) mention four such conditions:

1. No or little work from the user.
2. All messages should be categorized.
3. Categorization should be scalable, and
4. No messages should be misclassified.

We observed two applications of categorization in the domain of email systems: tagging, and classification. In the following sections, we would review the previous works for each application.

2.3.1. Tagging

Tags are unstructured, one-word/phrase labels that users apply to some object of digital information. Tagging has emerged as a mechanism for impromptu organization of information in social networking (Tang et al., 2008). It offers many desirable properties that make it easy to: assign multiple tags to a single resource, create new tags on the fly with minimal cognitive burden on the user, and represent information indexed by tags through usable tag clouds.

(Millen et al., 2006) (Ames et al., 2007) study the use of tagging and found it demonstrate several benefits in information retrieval. As for using tags in the emails domain, (Sorower et al., 2015) explored the benefits of using tagging in managing emails. They state that tagging email would be an important approach for managing email overload, their results of 14 users study conclude that implicit feedback mechanisms can provide a useful performance boost for email tagging systems. In addition, machine learning methods can help the user with this task by predicting tags for incoming emails. As a result, they proposed three algorithms (and two baselines) for incorporating implicit feedback into the TaskTracer, known as EP2 (Email Predictor 2). TaskTracer Email Predictor 2 (EP2) incorporates automated email prediction into Microsoft Outlook using a multi-label classifier based on the confidence weighted linear classifier.

2.3.2. Classification

One of the main advantages of classifying emails is to declutter the inbox and automatically moves emails into its related folders with no user's intervention required. (Whittaker et al., 2006) argue that classifying emails, by task, would declutter the inbox, increases task salience and reminds the user about ongoing tasks. (Freed et al., 2008) in Reflective Agents with Distributed Adaptive Reasoning (RADAR), considered email task detection as a text classification problem. They used a regularized logistic regression suite of classifiers (Yang et al. 2005) (based on body, headers, links) and combined their results. To adapt in user's email habits (e.g., new people, changing projects), the classifiers were designed to be incrementally adaptive. SCONE (Fahlman, 2006) had been used to improve classification performance; SCONE is RADAR's knowledge base which provides additional ontological information that is not contained in the email's content. Examples include basic facts, such as "the Connan room is in the University Center," and higher-level concepts, such as "a peanut is kind of food that people might be allergic to".

(Sorbo et al., 2015) propose a semi-supervised approach named DECA (Development Emails Content Analyzer) that uses Natural Language Parsing to capture linguistic patterns and classify emails' content according to developers' intentions, such as asking/providing helps, proposing a new feature or reporting/discussing a bug. Other projects such as (Qadir et al., 2016) try to train a logistic regression classifier. The trained classifier is then applied to the email data to identify the Thing of Interest (TOI) phrases. Their results show that syntactic and semantic knowledge such as verb phrases and thing of interests in emails can model the activities much better than bag-of-words.

(Wendt et al., 2016) present a technique for the categorization of machine-generated emails. They developed three new algorithms for classification of machine-generated emails using structural templates: majority label, centroid similarity, and hierarchical label propagation. Also, they explore number of underlying template representations including bag-of-words and topic distributions, different similarity metrics in template graph construction, and varying degrees of graph connectivity.

The task of automatically classifying emails prove to be a difficult task and there are several challenges associated with it. (Brutlag and Meek, 2000) identify some of these challenges, they recognize the need to adapt to changing behavior. In addition, (Pazzani, 2000) show that user have high expectations for text processing techniques used in email and being somewhat intolerant of errors. Other drawback of using supervised machine learning techniques, it requires the user to do additional work by annotation of large numbers of emails for training purposes. For this reason, using unsupervised learning may be preferred since it requires no additional work from the user on the training data. An example of unsupervised learning is clustering. Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

(Whittaker et al., 2011) suggest using clustering techniques to organize the inbox into ‘superthreads’ by combining multiple threads with overlapping topics. While (Surendran et al., 2005) used clustering as a means to derive the topics, not as an end in itself. They argued against email categorization/clustering methods which learn pre-assigned categories by the user, therefore, it would require efforts from the users to create and maintain such categories. Their proposed approach is multi-document key phrase extraction by picking a few characteristic keywords/key phrases for each topic and use those as a characterization of the topic itself.

2.4. Visualization As a Solution

The most popular interface, for displaying emails to the user, is the traditional linear interface which (Whittaker and Sidner, 1996) define it as a tabular format of email folders and once a new email has been read, the user is expected to execute an immediate action on the email. Because of this expectation of "an immediate action" and the time limitations of the user, several challenges arise (emails overload and tasks management), the former we describe in details in the next chapter. Therefore, many researchers try to solve these problems by focusing on the best interface that email clients should present emails to the user. One proposed interface, by (Baecker et al., 1997), is TimeStore; A two-dimensional representation which is based on a time-based email interface. In this interface, emails are automatically organized by time and by sender and displayed on a two-dimensional grid. Another similar example is TimeStore-TaskView, proposed by (Gwizdka, 2002), which is based on TimeStore and uses the same graphical representation. In this interface, email's tasks are represented by small icons on a two-dimensional grid with temporal task information shown on the horizontal and vertical axis, respectively.

Any solution should be based on two principles: First, there should always be a special path for people to get urgent, certified, and personal messages and second, all other paths should be filtered (Crocker, 1982). In alignment with these two principles, (Neustaedter et al., 2005a) states to design email interfaces which support email triage, the interface must reduce the number of items to triage, by grouping, and these groups must be created automatically. These groups may still have many entities to triage within, but it could provide users with additional meta-level attributes of the emails. Therefore, they suggest that email interfaces should also present additional socially noticeable information about the sender, receiver, and time attributes of the emails.

Another approach proposed by (Surendran et al., 2005) is a personalized user interface which can auto-arrange all the email/document according to the discovered topics. They defined personal topic as any cohesive concept that is relevant to the user. It could be an activity they participate in, an event they organized or attended, a person or a group of people they associate with, etc.

As for the problem of task/activities management in emails, (Kushmerick et al., 2006) vision is to provide activity-centric (rather than message-centric) tools that enable users to manage their activities together, rather than as isolated email. The underline key behind this approach is related emails in a task provide a valuable context that can be used for semantic email analysis. At the same time, the activity related metadata in separate email can provide relational clues that can be used to establish links between emails and group them into tasks (Khoussainov and Kushmerick, 2005).

(Gwizdka and Chignell 2004), (Venolia and Neustaedter, 2003), (Wattenberg et al., 2005) proposed several other solutions to visualize inbox tasks, including tree representations and flat representations of information related to specific tasks. These approaches depend on threads to group emails into a common task. Thus, they suffer from the limitation of threads as it’s a weak indicator of tasks because of different email responding practices and some emails include several topics. (Bellotti et al., 2005) try to overcome this limitation by depending on “thrasks”, which is a user-customizable groups based on threads.

2.5. The Social Relationship Between Sender and Receiver

Analyzing the social meta-data of emails may be useful to identify the relationship's strength between senders and recipients, in addition to other meta information such as the average respond time to a new email. An example of such an approach is DriftCatcher by (Lockerd and Selker, 2003), using this meta data, they train an SVM to classify emails by interaction process analysis, which labels emails with eight types of interactions, such as "informing," "inquiring," "planning" and "keepInTouch." Other researchers, such as (Neustaedter et al., 2005b), consider the below social metrics to capture multiple dimensions of the relationship between the user and their correspondents and among the correspondents themselves

- Number of times an author sent email over a time period,
- Number of those messages that were replies,
- Number of those messages that remain unread can be used for supporting email management.

They organize the social metrics of each email into "Sent metrics", which provide social information about email sent by the user to a correspondent, and "Received metrics", which provide information about email received by the user from a correspondent. In the following sections, below we discuss two applications that make benefits of analyzing the social meta data of emails.

2.5.1 Recipient Recommendation

If you are communicating with other using emails, there are chances you had forget to add an email recipient in at a least once. This problem is widespread among emails users, (Carvalho and Cohen, 2008) found, by searching the Enron email corpus, that at least

9.27% of the users have forgotten to add a desired email recipient in at least one sent message. While at least 20.52% of the users were not included as recipients (even though they were intended recipients) in at least one received message. They state that this task can be a valuable addition to email clients, particularly in large corporations, where negotiations are frequently handled via email and the cost of errors in task management is very high. Also, (Sofershtein and Cohen, 2015) stated that prediction of recipients allows for effective "auto-complete" of this field, thereby improving user experience and reducing the overhead of manual typing of the recipient.

Another problem where recipient recommendation would help is "stovepiping". (Pal and McCallum, 2006) define a stovepipe organization as an organization contains members who have narrowly defined responsibilities and information, output and feedback only move along a set path through a management hierarchy. They developed an automated system that suggest a list of additional recipients to the cc field. Using a simple Multinomial Naive Bayes model, they learn the probability distributions of recipients, words in body and subject, and the recipients given so far to predict recipients in email cc lists.

2.5.2. Email Leak

It's in our nature to feel more threatened by malicious, rather than accidental, behavior. Sharks are much scarier than cars but having a car accident is far more likely than getting eaten by Jaws. Research by CompTIA found that human error accounts for 52% of the root cause of security breaches (CompTIA, 2015). A freedom of information request to the ICO reported that 62% of data security incidents reported between January and April 2016 were due to human error (ComputerWeekly, 2016).

In their 2015 report, Verizon found that within ‘Miscellaneous Errors’ (the most common type of data security incident category), ‘misdelivery’ of information accounted for the majority of incidents (Verizon, 2015). (Carvalho and Cohen, 2007a) identify this problem as “email leaks” (i.e., when a message is accidentally addressed to non-desired recipients). The Bank of England, Goldman Sachs and the NHS Trust have all hit the headlines for data breaches due to misaddressed emails. In addition to this, the ICO in the UK recently reported that misaddressed emails were the number one type of data security incident reported to them (Information Commissioner’s Office, 2016).

Conventional solutions to the problem of misaddressed emails would be user should be more careful, email recall, encryption, double-check warnings, or allows IT administrators to apply pre-defined policies to outbound emails. All these approaches suffer from three key problems: highly disruptive to email’s users, create admin overhead for IT teams, and rely on purely rule-based methods to screen outbound emails (in other words, don’t adapt to users’ behavior changes).

More adaptable solutions would utilize the use of machine learning to analyze outgoing email content. Such solutions will be able to detect human error and potential data breaches in real time, before an email is sent, with no change to the way in which employees normally send emails. (Balasubramanyan et al., 2008) propose CutOnce which can be used to provide email leak prevention. CutOnce algorithm works by ranking only the already-specified recipients of an email under composition, with the least likely address on the top. While (Carvalho and Cohen, 2007) leverage content similarity by creating TFIDF centroid vectors and determining k-nearest neighbors of a target email. They try to determine appropriate recipients for an email and alerting the user when an email is sent to an inappropriate recipient (leak detection). They tried to approach this problem by redefining it as an outlier detection task, where the unintended recipients are the outlier.

An example of a commercial solution to the problem of "email leaks" would be CheckRecipient's developed by a team of Imperial College trained engineers, mathematicians and data scientists. They are using machine learning to analyze emails to understand the typical sending patterns and behavior of employees within the organization. This analysis allows to establish the baseline behavior for email communication to detect anomalies that may indicate misaddressing errors on outgoing emails. Every single time an anomaly is detected in an outbound email, CheckRecipient not only prevents the email from being sent but informs the sender why an anomaly was detected and asks them for confirmation on whether they would still like to proceed with sending.

CHAPTER 3

EMAIL OVERLOAD

In 1982, Peter Denning (then the ACM President) first wrote about the pain of working with email, 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?” (Crocker, 1982). How is the situation now, after three decades? (Radicati, 2015) estimates the number of emails sent and received daily in 2015 to be over 205 billion and expected to grow at an average annual rate of 3% over the next four years, reaching over 246 billion by the end of 2019. The number of business emails sent and received per user per day in 2015 to be 122 emails per day and expected to average 126 messages sent and received per business user by the end of 2019. So even after thirty-five years of Peter Denning’s question, it seems that his question still relevant.

Several researchers explore this problem and based on the reviewed literature, two definitions arise: First definition is Information Overload occurs as the volume of information received by the individual surpasses their ability to process it (Schuff et al., 2006). The second by (Dabbish and Kraut, 2006) (Alberts, 2013) define email overload as a perception of email users’ belief in their inability to process, find and handle the amount of email they send and receive. As for the consequences, Email overload would not only hinder the user’s capacity to manage their email inbox, but it could also cause time pressures and can increase working hours, which are reportedly large contributors of stress. (Edmunds and Morris, 2000) discovers that information overload causes stress amongst employees. As a result, (Song et al. 2007) coach readers on regaining control of their lives from the "tyranny of email."

In the following sections, we start by explaining what contributes to the problem of email overload, then exploring the available solutions and their limitations. Later, we present our proposed solution. Finally, we compare our model's results with a baseline model and other previous works.

3.1. What Contributes To Email Overload?

The average number of unread work-related emails increased from (153) in 2006 (Fisher et al., 2006) to (696) in 2014 (Grevet et al., 2014). Grevet made these estimations by replicate and extend on (Whittaker and Sidner, 1996) and (Fisher et al., 2006) works, with a qualitative analysis of Google's Gmail. Just like in 1996, email overload is still a problem, both in terms of volume and of status. A recent study, by (Kooti et al., 2015) of more than 2 million users exchanging 16 billion emails over several months, confirm Grevet's results regarding email overload as they conclude that users generally unable to keep up with rising load.

(Dabbish et al., 2004) found that 49% of incoming emails kept in the inbox. As for the reasons behind users' mailboxes been so full, an early study conducted by (Whittaker and Sidner, 1996) mentioned two reasons:

1. Email inbox serves as a task manager to remind the user of all his/her tasks. They classify emails, which don't be process immediately, into four types of emails:
    a. "To dos" (Require actions)

    b. “To reads” (requires time and efforts): They found that 21% of emails kept in inbox contained more than 5 screen-full of text. Thus, time limitation may be the reason.
    c. “Indeterminate status” (still evaluating its importance), and
    d. “Ongoing Corresponding” (draft emails pending more time or unavailable information).
2. Filling emails into folders is difficult and have few benefits.

A more recent qualitative study of 34 knowledge workers by (Alberts, 2013), identify five factors that contribute to this problem: Quantity (the amount of emails they received daily), Poor Targeting (ease of writing an email, insecurity of certain employees to CC to their manager, and internal communications), Large Attachments (the presence of many/larges attachments), Discussion Thread Length (take too much time to read), Propagation Effect (senders who are abusing the use of CC field, leading to the messages’ exponential multiplication). Another source, for email overload, is the machine generated emails. (Ailon et al., 2013) (Grbovic et al., 2014) studies shown that more than 90% of non-spam Web email is now machine-generated, with messages of various importance, from e-tickets or invoices, to hotel newsletters.

3.2. Available Solutions

Several strategies and tools were proposed and developed to help the users cope with the problem of email overload. At the same time, each of these solutions had its own limitations and, in some cases, contribute to further problems. In the following sessions, we are reviewing these solutions:

3.2.1. User’s Behavior As A Solution:

Information overload problems can be minimized by increasing user’s information processing capacity, reducing the job’s information load, or doing a combination of both. Studies suggest that employees often increase their information processing capacity by temporarily reading faster, scanning through documents more efficiently, and removing distractions that slow information processing speed, (McShane and Von Glinow, 2015) state that information load can be reduced by buffering, omitting, and summarizing. An example of Buffering would be filtering incoming emails, either by a human assistant or by automatic rules. Another use of automatic rules would be omitting and ignoring some unimportant emails by redirect them from the inbox folder into folders that user never look at. In general, (Whittaker et al., 2006) 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 (Bellotti et al., 2005) as it requires a commitment from the user to invest considerable amounts of time and efforts to construct the folders’ organization for future retrieval. Several studies have shown that email users experience problems in organizing their emails, especially when asked to find already archived emails (Boardman and Sasse 2004). Furthermore, a successful filling is highly depending on predicting future retrieval requirements (Whittaker et al., 2006), which proven to be a cognitively difficult task (Kidd, 1994). Since user need to remember where a particular email is located and the name of the targeted folder. (Elsweiler et al., 2008) observe that participants in their study of memory for email messages correctly recalled over %80 of content, purpose, or task related of particular email in their main inbox, while “frequent fliers” tend to remember less about their email. One reason may be that foldering emails prevent the user from exposing to those emails frequently (Whittaker et al., 2011).

Another problem associate with such method is the efficiency of email archive and retrieval. Since an extra time would be required in searching for the targeted folder when created and when searching for a specific email. Therefore, as the number of folders increase, the efficiency of email’s retrieval would decrease (Bälter, 2000). This on the other hand may lead to “failed folders” (Whittaker and Sidner, 1996); folder contain different emails, or duplicate folders contain similar emails, such folders had been created but not utilized well. All these challenges would explain why (Whittaker et al., 2007) found that users sometimes avoid foldering emails that later turns out to be useless or irrelevant. While (Neustaedter et al., 2007) identify that large proportion of users live with a flat inbox structure, with just a few folders and filter rules for newsletters.

So, why users keep foldering emails despite all these challenges? (Whittaker et al., 2011) argue that foldering is a just reaction to the phenomenon of receiving many emails rather than response to increased demands for refinding emails. Users who receiving large number of emails, tend to folder those emails to decluttering their inbox. Other researchers argue that users are not foldering much of their emails. (Koren et al., 2011) study shown that 70% of Web mail users never defined a single folder. (Grevet et al., 2014) found that inboxes show indication of having a large number of emails, fewer messages are archived, and labels are not as extensively used in Gmail as folders were in previous studies. They also noticed that the volume of emails is much greater in personal accounts than work accounts, and the number of unread emails is much greater in personal email as well.

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 (Whittaker et al., 2006). Another problem with both foldering and searching, it's difficult to find a master method that will accommodate the different needs and habits for all users. Working with emails has come to teach us how diverse and unpredictable people's email reading/storage habits can be. Therefore, a technique which work perfectly for one user may not guarantee to work well for another user.

3.2.2. Email Client As A Solution

Several email clients provide tools to manage emails overload. Users respond by keep scanning email inbox, marking email as unread or flagging it as important for more process later, sorting emails by sender or flags rather than by time, moving emails to folders for later reference, deleting irrelevant emails from the inbox (Bellotti et al., 2005). (Venolia et al., 2001) question the effectiveness of these tools and the adoption rate by users, as they found small number of users (%30 of their samples) utilized such tool to handle their emails. The same conclusion was confirmed by (Bellotti et al., 2005) through their preliminary survey of email tool usage, they found that users use a fraction of the tools provided by an email client such as Microsoft Outlook.

Another shortage of email client, especially in helping the user in emails triage activity, its lack of providing sufficient and relevant information for identifying important new email (Venolia et al., 2001). Since most activities are distributed over multiple email, yet email clients allow users to manipulate just isolated email (Khoussainov and Kushmerick, 2005).

Meanwhile, one of the most successful applications for automation in the email's domain is spam filtering. Although email spam filter control unwanted emails and prove to be an effective solution for the problem of spam emails, (Grevet et al., 2014) state that emails which users sign up for such as discounts, store receipts, or for other reasons, are more difficult to filter through traditional spam detection mechanisms.

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 (Grbovic et al., 2014). 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 have 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 (Aberdeen et al., 2010). 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.).

An issue, with automatically classifying incoming emails into predefined categories, is who to determine which category an email belong to. In the case of Gmail Priority Inbox, to which tab should an email sent by an advocacy groups placed in? the primary inbox tab or the promotion tab? Keeping in mind that these tabs serve another purpose: ad inventory. While Gmail does not sell ads in the primary inbox, advertisers can pay for top placement in the social and promotions tabs in free accounts. So, if an email sent by advocacy groups or political parties to be considered promotions emails, there are legitimate concerns that Gmail's tabs and inbox ads would turn into a Facebook-style news feed where you have to pay for placement.

In general, the task of automatically classifying emails proves to be a difficult task and there are several challenges associated with it. (Brutlag and Meek, 2000) identify that these classification model should recognize the need to adapt to changing behavior. Also, (Pazzani, 2000) shows that user has very high expectations and being somewhat intolerant of errors. Furthermore, there is the problem of Topic drift. 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") (Cselle et al., 2007). 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.

3.3. The Proposed Solution

Studying users' needs and what they want from emails clients help us in our mission to design and develop an intelligent email assistant. For this end, accessibility and visibility are the most two characteristics of an email client that user requested. (Dabbish and Kraut, 2006) found that users want email information to be more available at the surface level, which confirm (Whittaker and Sidner, 1996) previous findings that user prefers availability and visibility. Dabbish and Kraut found, in their survey-study of 484 email users with widely varying job characteristics, that users have a smaller number of folders and keep their inbox small. A behavior that increases the surface level visibility of individual email messages and reduced the feelings of email overload. Therefore, we conclude that the best solution to serve the users' needs should have these two main characteristics: 1) help the user read and access the email content more efficiently, and 2) it should not add more complexity or require a change in users' behavior.

3.4. Speech Act

To achieve the above goal, we choose to follow the steps of (Cohen et al., 2004) by utilizing the speech acts theory to cover 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 an apology, greeting, request, complaint, invitation, compliment, or refusal, he/she uses speech act. Speech act has been used to model conversations for automated classification and retrieval (Twitchell et al., 2004) and it would provide an effective way of summarizing the intended purpose of an email message (Franovic and Šnajder, 2012).

One of the applications of speech act theory is the classification of emails into Email Speech Acts. This taxonomy is based on Speech Act Theory (Searle, 1969) (Austin,

1962) and characteristics of email. (Cohen et al., 2004) utilize speech acts theory and work on ontology of nouns (information and activity) and verbs (request, propose, amend, commit, deliver, refuse, greet and remind) covering some of the possible speech acts associated with emails. They assume that a single email could contain multiple acts, and that each act is described by a verb-noun pair drawn from this ontology (e.g., "deliver data"). As a result, each email may be annotated with several labels, as it may contain several speech acts.

As for the nature of requests and commitments in email, (Lampert et al., 2006) studied those and defined a Verbal Response Modes (VRM) taxonomy of speech acts, which classify emails into two dimensions: literal meaning and pragmatic meaning. In (Lampert et al., 2008), they state that the ontological foundation of their taxonomy is the notion of an action and they define it as:

- Actions are carried out by agents.
- A request is placing of an obligation by one agent on another agent to carry out the requested action.
- A commitment is the taking on, by some agent, of an obligation to carry out an action.
- Both requests and commitments may be conditional

They define requests as sentences carrying an expectation that the recipient of the email should act for it, and commitments as sentences carrying an expectation that the sender is promising future action from themselves or on behalf of another person.

Studying how users use emails will give us a better idea of which email speech acts to utilize for our work. (Grevet et al., 2014) 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 are focusing on work-related emails in this project, we 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 move the emails into separate folder/tab/bundle, we 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. 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.

3.5. The Enron 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 (Bird and Loper, 2004), 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.

In addition to our labeled dataset, 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 (Cohen et al., 2004) 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").

In total, our dataset contains 11319 labeled sentences, 4124 sentences labeled as intent and 7195 sentences labeled as delivery. Then, using sklearn library (Pedregosa et al., 2011), we split our dataset set into 80% as training set and 20% as a testing set. 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.

3.6. 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 (Firth, 1957). There is a linguistic theory behind the approach, namely the "distributional hypothesis" by (Harris 1954). 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) (Pennington et. al., 2014). 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 (Mikolov et. al., 2013) 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 vectors (100 dimensions) trained on 6 billion tokens from Wikipedia 2014 corpus and word2vec pre-trained word vectors 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 (Rehurek and Sojka, 2010) with the following hyper-parameters (windows size = 5 and dimension size = 100), to obtain word vectors for 94,673 unique words.

3.7. 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 Support Vector Classifier 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 (Collobert et. al., 2011). An advantage of using neural networks is that they require no hand-crafted features and enable automatic feature representation learning. (Young et. al., 2017) provide a comprehensive review of most deep learning methods which have been used in NLP research today. In this chapter, we are focusing on two deep learning architectures, Convolutional Neural Networks (CNN) (LeCun et. al., 2015) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

3.7.1. Convolutional Neural Network

Convolutional Neural Networks (CNN) have recently been shown to achieve impressive results on the practically important task of sentence categorization (Kim, 2014; Kalchbrenner et. al., 2014; Goldberg, 2016). 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 (Collobert et. al., 2011). Initially, CNN was designed to be used in image processing tasks. 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.

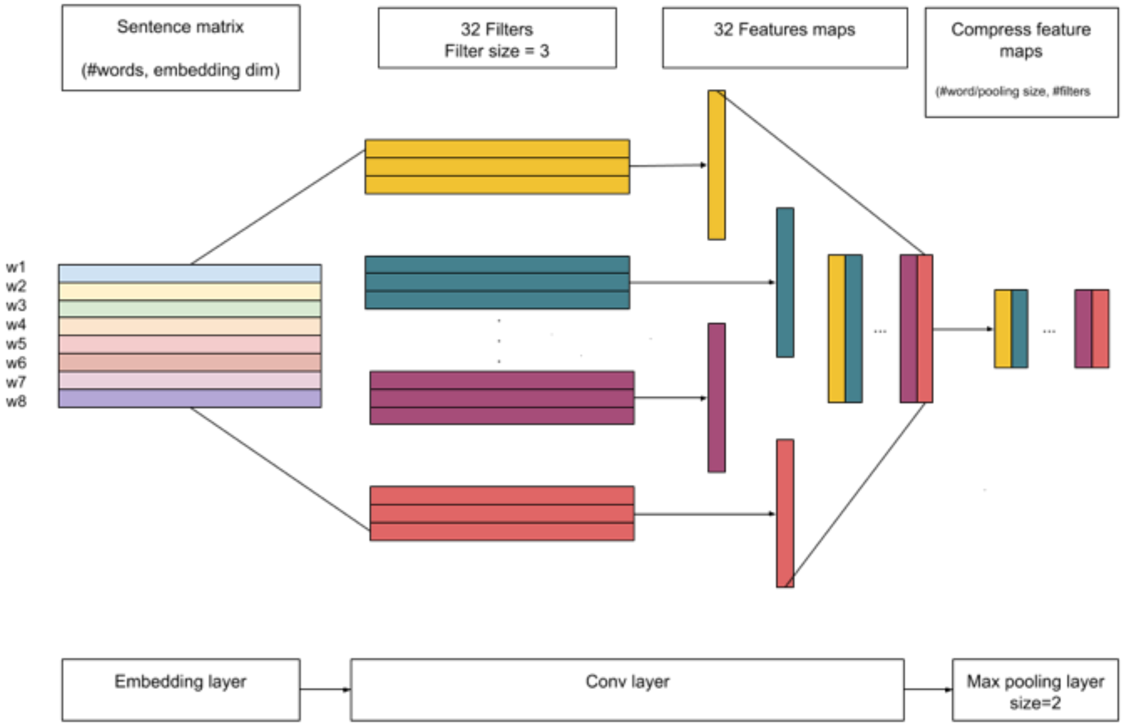

FIGURE 3.1: OUR CNN FEATURE EXTRACTOR ARCHITECTURE

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. Figure 3.1 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.

3.7.2. 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. In vanishing gradient problem, the weight's update is minor and results in slower convergence; This makes the optimization of the loss function slow and in the worst case, may completely stop the neural network from learning. As for the exploding gradient, this is the exact opposite of vanishing gradient. Consider you have non-negative and large weights and small activations A. When these weights are multiplied along the layers, they cause a large change in the cost. Thus, the gradients are also going to be large. This means that the changes in weights will be in huge steps, the downward moment will increase. This may result in oscillating around the minima or even overshooting the optimum again and again and the model will never learn.

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.

3.8. The Proposed Model Architecture

Figure 3.2 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 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.

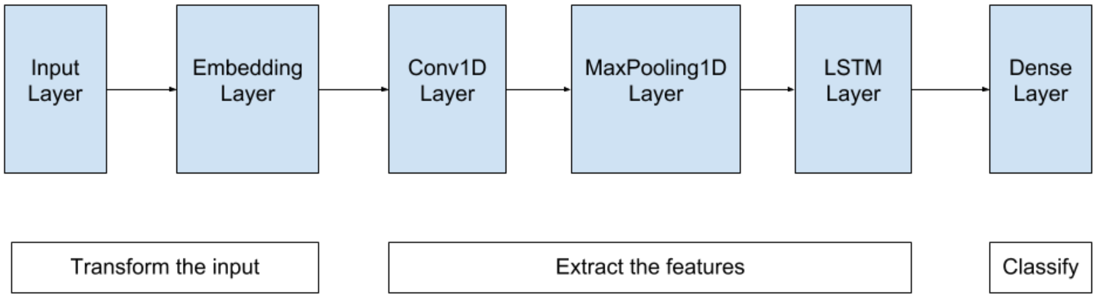

FIGURE 3.2: THE ARCHITECTURE OF OUR MODEL

3.9. The Experiment and Evaluation Results

To evaluate our model's performance, we compared our model's result with 1) Traditional approach, which consists of using TF-IDF to represent the input and a Support Vector Classifier to classify the input sentence into "to do" or "to read". 2) Deep Learning approach: We reproduced the models described in (Kim, 2014) and (Kudugunta and Ferrara, 2018), then we trained these models on our own labeled dataset and compare the result with our own (CNN-LSTM) model.

The traditional approach consists of using TF-IDF to represent the input and a Support Vector Classifier 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 below model’s hyper-parameters using 3 folds:

- 'vect__ngram_range': [(1, 1), (1, 2), (1, 3), (1, 4)],
- 'tfidf__use_idf': (True, False),
- 'clf__kernel': ("linear", "poly", "rbf", "sigmoid"),
- 'clf__C': [0.001, 0.01, 0.1, 1, 10],
- 'clf__gamma': [0.001, 0.01, 0.1, 1],
- 'clf__degree': [3,4,5]

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.

As for comparing our CNN-LSTM model performance with a Deep learning approach, we reproduced the models described in (Kim, 2014) and (Kudugunta and Ferrara, 2018). (Kim, 2014) reports on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. The classification's tasks include both sentiment analysis and multi questions classification. While (Kudugunta and Ferrara, 2018) used a deep neural network based on long short-term memory (LSTM) architecture to detect bots at the tweet level. They used Glove word embeddings to represent the input. In the end, we compared our (CNN-LSTM) model with the following models:

1. CNN Model: one-layer CNN followed by a densely connected NN layer with a "sigmoid" function. These are the hyper-parameters of this model: CNN filters numbers: 32, filter size: 3, pool size: 2, batch size: 16 and epochs: 10. We tried different versions of this model:
   - CNN on top of randomly initialized embeddings and then modified during training.
   - CNN on top of pre-trained word2vec embeddings
   - CNN on top of pre-trained Glove word embeddings
   - CNN on top of Enron Embeddings
2. Multichannel CNN: 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 trigram. We tried the following different versions of this model:
   - Multichannel CNN on top of randomly initialized embeddings and then modified during training.

   - Multichannel CNN on top of pre-trained word2vec embeddings
   - Multichannel CNN on top of Glove word embeddings
   - Multichannel CNN on top of Enron Embeddings

3. Three layers of CNN with randomly initialized embeddings and then modified during training. We used the following hyperparameters: filters numbers =[64,32,16], kernel size = [3,3,3], batch size= 16 and trained the model for 10 epochs.
4. LSTM: one-layer LSTM followed by a densely connected NN layer with a “sigmoid” function. These are the hyper-parameters of this model: cell numbers= 100, dropout = 0.5, rec dropout = 0.2, batch size= 16 and trained the model for 10 epochs. We tried the following different versions of this model:
   - LSTM on top of randomly initialized embeddings and then modified during training.
   - LSTM on top of pre-trained word2vec embeddings
   - LSTM on top of Glove word embeddings
   - LSTM on top of Enron Embeddings
5. Our proposed Model CNN + LSTM: One layer of CNN followed by one-layer LSTM. Then, a densely connected NN layer with a “sigmoid” function. The hyper-parameters values are the following: CNN filters numbers: 32, filter size: 3, pool size: 2, activation function is “relu”, and LSTM size = 100. 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 other models.

Table 3.1 shows the evaluation results. We are using F1 score to report our results, F1 score is the harmonic average of the precision and recall.

TABLE 3.1: EVALUATION RESULTS

Model	**Accuracy**	**Loss**	**Precision**	**Recall**	**F$_1$**
Traditional approach: TF-IDF + SVC	85.07%	-	85%	85%	0.85
CNN + rand. initialized embeddings	81%	1.03	81%	81%	0.81
CNN + Enron embeddings	80%	0.82	82%	89%	0.81
CNN + Glove embeddings	82%	0.60	82%	83%	0.82
CNN + Word2vec embedding	82%	0.74	83%	83%	0.83
Multichannel CNN + rand. initialized embeddings	80.30%	0.93	81%	80%	0.80
Multichannel CNN + Enron embeddings	86.70%	0.37	87%	87%	0.87
Multichannel CNN + Glove embeddings	84.36%	0.37	84%	84%	0.84
Multichannel CNN + Word2vec embedding	85.07%	0.42	85%	85%	0.85
Multichannel CNN + Enron, Glove, word2vec embeddings	84.93%	0.38	85%	85%	0.85
3 layers of CNN + rand. initialized embeddings	79.01%	0.87	79%	79%	79%
LSTM + rand. initialized embeddings	81.75%	0.73	82%	82%	82%
LSTM + Enron embeddings	86.57%	0.31	87%	87%	87%
LSTM + Glove embeddings	84.36%	0.34	85%	84%	84%
LSTM + Word2vec embeddings	85.60%	0.33	86%	86%	86%
Our Model CNN + LSTM + rand. initialized embeddings	80%	0.90	81%	81%	81%
Our Model CNN + LSTM + Enron embeddings	**89%**	**0.29**	**89%**	**89%**	**89%**
Our Model CNN + LSTM + Glove embeddings	86%	0.30	87%	87%	87%
Our Model CNN + LSTM + word2vec embeddings	86%	0.33	86%	86%	86%

3.9.2. Results Comparison To Related Works

(Cohen et. al., 2004) 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, (Kraut et. al., 2005) 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) (Schapire and Singer, 1999) produce their best results: above 80% precision and above 50% recall. In a follow-up work by (Carvalho and Cohen, 2006), 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.

(Lampert et. al., 2010) focused only on Request speech act. Their request classifier works on the message level. Their approach consists of 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". Another SVM classifier, implemented using Weka (Hall et. al., 2009), 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 words, 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.

(Qadir and Riloff, 2011) 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%.

(Franovic and Snajder, 2012) 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.

All 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 3.2, 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 users intents in emails.

3.10. Analysis

The aim of our research, in this chapter, 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. The main goal 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 3.2: COMPARISON OF OUR WORK WITH RELATED WORKS

Model	Level	Hand Crafted features	Speech acts	Precision	Recall	F_1
Cohen et al., 2004	message	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	message	Yes	Request	84.90%	83.7%	0.84
			Non-request	82.50%	83.9%	0.83
Qadir & Riloff, 2011	Sentence	Yes	Commissive	63%	39%	0.48
			Directive	87%	85%	0.86
			Expressive	97%	91%	0.94
			Represent	32%	16%	0.21
Franovic & Šnajder, 2012	Sentence	Yes	Deliver	-	-	0.88
			Amend	-	-	0.72
			Commit	-	-	0.78
			Remind	-	-	0.69
			Suggest	-	-	0.69
			Request	-	-	0.72
Ours	Sentence	No	Intent-To do	90%	93%	0.91
			Delivery-To read	87%	82%	0.85

CHAPTER 4

TRANSFER LEARNING IN THE EMAIL DOMAIN

Neural networks are networks that information flows through. In the forward pass the input flows and transforms, hopefully becoming a representation that is more amenable to the targeted task. During the back phase we propagate a signal, the gradient, back through the network. Its standard that a neural network consists of multiple layers of non-linearity functions stacked together. The actual role of the non-linearity is to twist and turn the feature space so that the boundary turns out to be linear. With each layer, the network transforms the data, creating a new representation. These representations make the data “nicer” for the network to classify. We can look at the data in each of these representations and how the network classifies them. When we get to the final representation, the network will just draw a line through the data (or, in higher dimensions, a hyperplane).

In order to reach the “best” final representation, the network need to be trained on large amount of data. The main objective of training a neural network is to identify the correct weights for the network by multiple forward and backward iterations. In most cases these weights would be initialized randomly. More recently, pre-trained models, which have been previously trained on larger datasets, are been utilized to initialize these weights and then retrain the model. This approach is known as transfer learning. We “transfer the learning” of the pre-trained model to our specific problem statement. With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. Since a neural network is trained on data, the network gains knowledge from this data, which is compiled as “weights” of the network. These weights can be extracted and then transferred to any other neural network. Instead of training the other neural network from scratch, we “transfer” the learned features and such a model would be called a pre-trained model.

There are different situations when using a pre-trained model may be useful as it saves time and processing power required to train the model from scratch:

- Use the architecture of the pre-trained model: We use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.
- Feature extraction: We can use a pre-trained model as a feature extraction mechanism. We can remove the output layer (the one which gives the final predicted probabilities) and then use the entire network as a fixed feature extractor for the new data set. An example of this usage would be using a ConvNet that has been pre-trained on ImageNet, remove the last fully connected layer, then treat the rest of the ConvNet as a feature extractor for the new dataset. Once you extract the features for all images, train a classifier for the new dataset.
- Train some layers while freeze others: Another way to use a pre-trained model is to train it partially. The main idea is to keep the weights of initial layers of the model frozen while retrain only the higher layers.

Another situation, where using transfer learning would be preferable, is when there is not much of labeled training data. The general idea is to use knowledge learned from tasks for which a lot of labelled data is available in settings where only little labelled data is available. Creating labelled data is expensive, so optimally leveraging existing datasets is a key. Transfer Learning is mostly used in Computer Vision and Natural Language Processing tasks like sentiment analysis, because of the huge amount of computational power that is needed for them.

In the following sections, we start by giving more details into the application of transfer learning in both the vision domain as well as the NLP domain, what is Language Modeling and its role in transfer learning in text. Finally, we will present our approach, compare our model's results with a baseline and other previous works.

4.1. Transfer Learning in Vision

Deep learning methods have led to significant successes in computer vision. Typically, deep learning techniques have been invented and applied in research settings on enormous datasets, such as ImageNet (Deng et al., 2009) or MS Coco (Lin et al., 2014). To increase performance on these large datasets, researchers have come up with network architectures with increasing depth and complexity (Simonyan and Zisserman, 2014) (He et al., 2016) (Chollet, 2017). All these network architectures required gathering considerable amounts of images which have been annotated so it can be feed into the network in order to train it. That's in turn would require a considerable time and processing power.

As it turns out, deep learning networks learn hierarchical feature representations (Olah et al., 2017). This means neural Networks usually try to learn low level features, such as edges in their earlier layers, shapes in their middle layer and some task-specific features in the later layers. With transfer learning, we can use the early and middle layers and only re-train the latter layers. It helps us to leverage the labeled data of the task it was initially trained on. Transfer learning is mostly used in Computer Vision because it can reduce the size of the dataset, which decreases computation time and makes it more suitable for traditional algorithms as well.

4.2. Transfer learning in NLP

While deep learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Usually, we need a lot of data to train a Neural Network from scratch, but we don't always have access to enough data. That is where Transfer Learning comes into play because, with it, we can build a solid machine Learning model with comparatively little training data because the model is already pre-trained. This is especially valuable in Natural Language Processing (NLP) because there is mostly expert knowledge required to created large labeled datasets. A simple example of transfer learning would be using just a single layer of weights (known as embeddings) which has been extremely popular for some years, such as the word2vec embeddings from Google.

Amore computer-vision-like transfer learning example, where an entire pre-trained model used, is (Howard and Ruder, 2018). They proposed Universal Language Model Fine-tuning (ULMFiT), a transfer learning method that can be used to achieve computer-vision-like transfer learning for any task for NLP. They show that their method outperforms the state-of-the-art on six text classification tasks. In short, their method looks like this: Train a Language Model (LM) on a huge dataset or download pre-trained one, fine-tune this LM on the targeted dataset, then add few layers and fine-tune it to solve the task at hand.

In this work, we are building on their work. Our Hypothesis is that training a Language Model on the entire Enron email data (around 500,000 emails) and then using transfer learning to adapt this pre-trained Language Model on our labeled intent emails dataset would outperform both the traditional hand-crafted features approach and the use of an LSTM model trained on the intent email dataset.

4.2.1. Language Model

A statistical language model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P\ (w_1, w_2, w_3, \dots, w_m)$ to the whole sequence. For each word in the language's vocabulary, the Language Model (LM) computes the probability that it will be the next word. A language model learns these probabilities based on examples of text. Simpler models may look at a context of a short sequence of words, whereas larger models may work at the level of sentences or paragraphs. Most commonly, language models operate at the level of words.

Recently, neural-network-based language models have demonstrated better performance than classical methods both standalone and as part of more challenging natural language processing tasks. Currently, all state-of-the-art language models are neural networks (Merity et al., 2017). (Jozefowicz et al., 2016) found that LSTM-based neural language models out-perform the classical methods. The use of neural networks in language modeling is often called Neural Language Modeling, or NLM for short. The main reason for this improved performance may be the method's ability to generalize. Specifically, a word embedding is adopted that uses a real-valued vector to represent each word in a project vector space. This learned representation of words based on their usage allows words with a similar meaning to have a similar representation (Kim et al., 2016).

A language model can be developed and used standalone, such as to generate new sequences of text that appear to have come from the corpus. Also, it could be a fundamental part of many systems that attempt to solve natural language processing tasks such as machine translation and speech recognition (Goldberg, 2017).

4.3. The Proposed Model Architecture

To apply transfer learning into our problem, we are following this approach: first, train a Language Model on the entire Enron Email Dataset (around 2m sentences), remove last dense layer and add new dense layer for our binary classification task (i.e., classify to do or to read), retrain this model on our training subset of the labeled intent dataset. Finally, test this model on the testing subset of our labeled intent dataset. The below Figure 4.1 illustrates the main steps of our approach.

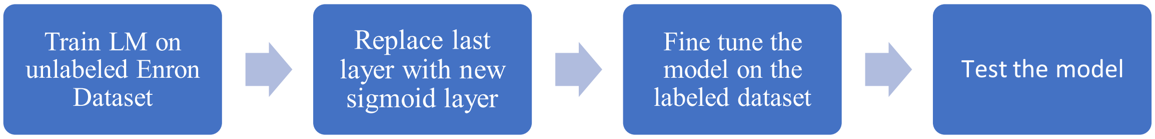
Train LM on unlabeled Enron Dataset
Replace last layer with new sigmoid layer
Fine tune the model on the labeled dataset
Test the model

FIGURE 4.1: OUR LM DEVELOPMENT MAIN STEPS

4.4. The Experiment and Results

We used the same Enron emails dataset, which consists of 495,546 real-world written emails. Before performing any preprocessing step, we decided to filter out any email that was not send from an enron.com domain or the word “Enron” is not mentioned in the “From” field. The logic behind this decision is that we found that many emails sent from not an Enron domain, are either newsletters or machine generated emails. We, also, eliminated any machine generated emails from the email addresses mentioned in the following Table 4.1.

TABLE 4.1: REMOVED EMAIL ADDRESSES

Email address removed
outlook.team@enron.com
arsystem@mailman.enron.com
enron.announcements@enron.com
announcements.enron@enron.com
enron_update@concureworkplace.co
exchange.administrator@enron.com

The first pre-processing step was separating the body section (i.e., the actual content of the email composed by the sender) from other sections (i.e., forwarded previous email or the signature). Then, we replaced website's link or email address mentioned in the body of an email by the tokens [LINK] and [EMAIL], respectively. After that, we replaced any duplicated special characters with one copy of that character. Next, we got rid of all emails which are empty or contain only numbers of special characters (i.e., no actual written sentences), 45,921 emails were deleted in this step. Finally, we used NLTK package to extract all the sentences from all emails and then saved it into a text file. In the end, we have one text file which contain all the written sentences in the entire Enron emails dataset.

To transform the entire Enron emails text, we have first to read the entire text file, mentioned above, as a one long sequence of strings. Then, we encode the text as integers. Each word in the source text is assigned a unique integer so we can convert the sequences of words to sequences of integers. Keras provides the Tokenizer class that can be used to perform this encoding. First, the Tokenizer is fit on the source text to develop the mapping from words to unique integers. Then sequences of text can be converted to sequences of integers by calling the texts_to_sequences() function.

To train the language model, we need pairs of input and target output words. To generate (input, output) pairs for the model to learn from, we read the Enron's emails 30 tokens a time as the input and the following 31st token would be the target. The below Table 4.2 shows the shape of our dataset after this step.

TABLE 4.2: TRAINING DATASET SHAPE

Input				Label
W1	W2	…	W30	W31
W2	W3	…	W31	W32
W3	W4	…	W32	W33
W4	W5	…	W33	W34

Finally, since, the language model will be statistical and will predict the probability of each word in the vocabulary given an input sequence of text, we need to fit the language model to predict a probability distribution across all words in the vocabulary. That means that we need to turn the output element from a single integer into a one hot encoding with a 0 for every word in the vocabulary and a 1 for the actual word that the value. This gives the network a ground truth to aim for, from which we can calculate error and update the model. Keras provides the to_categorical() function that we use to convert the integer to a one hot encoding while specifying the number of classes as the vocabulary size.

4.4.1. Training The Language Modeling

After we finished pre-processing and transformed the input and output. We defined the neural language model. The model uses a learned word embedding in the input layer. This has one real-valued vector for each word in the vocabulary, where each word vector has a specified length. In this case we used a 100-dimensional projection. The model has a two hidden LSTM layer with 256 units. For regularization, we added a dropout layer of (0.2) after each LSTM layer. The output layer is comprised of one neuron for each word in the vocabulary and uses a Softmax activation function to ensure the output is normalized to look like a probability.

Next, we compile and fit the model on the encoded text data. Since we are modeling a multi-class classification problem (predict the word in the vocabulary), we used the categorical cross entropy loss function. We used the efficient Adam implementation of gradient descent and track accuracy at the end of each epoch. The model was trained for 2 epochs.

4.4.2. Fine Tuning The Pre-trained Language Model

We used the same labeled dataset which has been described previously, also we followed the same pre-processing steps and labeling guidance. We experienced with three different scenarios where we hypothesis that applying transfer learning may be useful, but first we will describe our base model that we used to evaluate each model against.

TABLE 4.3: BASELINE MODEL RESULT

	Precision	**Recall**	**F1**
Intent Class	87%	90%	0.89
Delivery Class	82%	77%	0.8
Overall	85%	85%	0.85

The baseline model uses a learned word embedding in the input layer. This has one real-valued vector for each word in the vocabulary, where each word vector has a specified length. In this case we used a 100-dimensional projection. The model has two hidden LSTM layer, each layer with 256 units. For regularization, we added a dropout layer of (0.5) after each LSTM layer. Finally, to make the prediction, we are using a regular densely connected layer with a “sigmoid” function to squeeze the output feature vector from the LSTM. We trained this model on %80 of our intent email dataset for 2 epochs with batch size of 16. Then, we tested the model on the remaining %20. The above Table 4.3 shows this model results:

In the first scenario, we used the pre-trained Language model as a features extractor by removing the last dense layer, then using Support Vector Classifier (SVC) to classify the intents of the emails. We used the following parameters for the SVC: C=1, kernel= "rbf", gamma=0.1, degree = 3. Using the pretrained Language Model as features extractor proved to be a failed experiment. The below Table 4.4 shows this model's results:

TABLE 4.4: SVC MODEL RESULT

	Precision	**Recall**	**F1**
Intent Class	64%	100%	0.78
Delivery Class	98%	6%	0.11
Overall	77%	65%	0.54

In the 2nd scenario, we replaced the last dense layer of the language Model, by a new dense layer and retrain the entire model on our intent data subset. We trained this model on %80 of our intent email dataset for 8 epochs with batch size of 16. Also, we decrease the learning rate to (0.0001). Then, we tested the model on the remaining %20. Following this approach proved to be a success. Comparing the baseline model's result, we improved the model's precision and recall for both the Intent act and Deliver act. The below Table 4.5 shows this model results:

TABLE 4.5: FIRST RE-TRAINED MODEL RESULT

	Precision	**Recall**	**F1**
Intent Class	88%	91%	0.90
Delivery Class	84%	80%	0.8
Overall	87%	87%	0.87

Retraining all layers of the Language Model at the same time may be an aggressive way to let the pre-trained Language Model adapt to the knowledge obtained from our intent email's dataset. So, we decided to retrain the entire model gradually. We start by freezing all the layers' weights and then unfreeze and retrain them one layer a time, starting from the last (i.e., the dense layer). After freezing all the Language Model's layer, except the last dense layer, we trained this model on %80 of our intent email dataset for 70 epochs with batch size of 16. Also, we decrease the learning rate to (0.0001). We tested the model on the remaining %20. Next, we unfreeze the second LSTM layer's weights and the following dropout layer. We retrain the model again with the same parameters as the above for another 10 epochs. Finally, we unfreeze all the layer's weights and retrain the entire model for another 3 epochs. The below Table 4.6 show the results after each step:

TABLE 4.6: BEST RE-TRAINED MODEL RESULT

	Intent		**Deliver**		**Overall Precision**	**Overall Recall**	**Overal l F1**
	Precision	**Recall**	**Precision**	**Recall**			
Retrain last layer only	0.79	0.91	0.79	0.59	0.79	0.79	0.78
Retrain last 3 layers	0.86	0.93	0.87	0.75	0.87	0.87	0.86
Retrain all layers	**0.91**	0.91	0.84	**0.84**	**0.88**	**0.88**	**0.88**

4.5. Analysis

We found that our model is able to detect the intent of the sentence with an accuracy of 88.43% and a loss rate of 0.32. The precession and recall, of the "intent" act, are 91% and 91% respectively. While the precession and recall, of the "delivery" act, are 84% and 84% respectively. These results confirm our hypothesis that using transfer learning on Language Model, outperforms using only a two LSTM layers model. As shown in the below Table 4.7, using a simple pretrain Language Model and then gradually retrain it on our intent email's dataset, we were able to improve the accuracy of the model and decrease its loss, comparing with the baseline model.

TABLE 4.7: BASELINE VS. RE-TRAINED LM MODEL RESULTS

Model	**Precision**	**Recall**	**F_1**
2 LSTM Baseline model	85%	85%	0.85
Transfer learning on pretrained Language Model	**88%**	**88%**	**0.88**

CHAPTER 5

INVESTIGATING THE EFFECT OF CONTEXUAL REPRESENTATION IN EMAIL

Up to this point in our research, we utilized two embeddings: word2vec and Glove to represent the textual data of emails. Both these embeddings are belonging into the same type of embeddings, context free embedding. In this type of embedding, each word has one embedding without any consideration to the context where it's uses. In recent years, other types of embeddings have been developed. In the next phase in our work, we compare different word embeddings and sentence encodings with the goal of understanding which embeddings/encodings are more suitable for use in the task of detecting the intent of an email. We focus our experiments on three different types of embeddings:

- Pre-trained context-free word embeddings: word2vec (Mikolov et. al., 2013), GloVe (Pennington et. al., 2014), and context-free word embeddings trained on the entire Enron emails dataset using the word2vec algorithm.
- Pre-trained contextual word embeddings: Embeddings from Language Models (ELMo) (Peters et. al., 2018) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et. al., 2018)
- Pre-trained sentence embeddings: Universal Sentence Encoder (USE) and Transformer (Cer et. al., 2018)

Then, using a simple Neural Network model consists of one dense layer of 256 nodes followed by one node dense layer to perform the classification.

5.1. Context-Free Word Embeddings: Word2vec and GloVe

As we discussed in chapter 3, there are two approaches for computing word embeddings: Frequency based embedding, explained in Figure 5.1 and Prediction based embedding explained in Figure 5.2. 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.

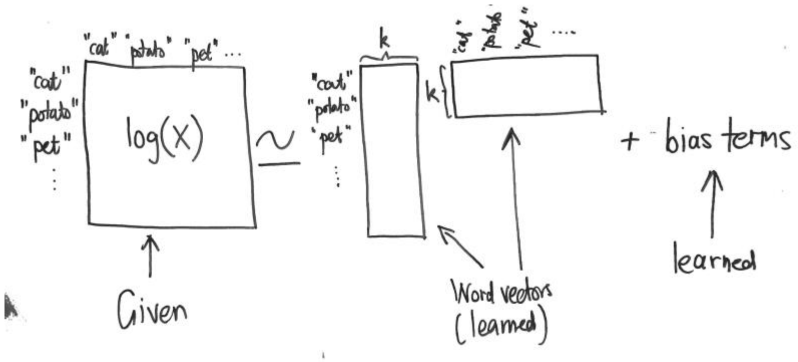

FIGURE 5.1: FREQUENCY BASED EMBEDDINGS

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). A pre-trained word2vec embeddings available to download from (https://code.google.com/archive/p/word2vec/), while a pre-trained GloVe embedding is available to download from (https://nlp.stanford.edu/projects/glove/).

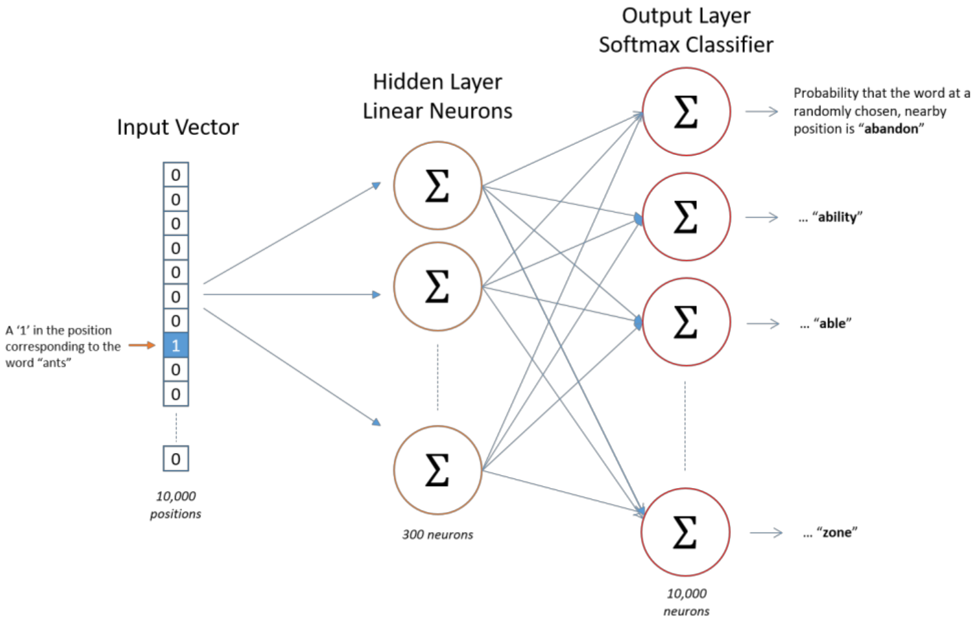

FIGURE 5.2: PREDICTION BASED EMBEDDINGS

5.2. Contextual Word Embeddings: ELMo

One disadvantage of using the above word embeddings (word2vec or GloVe) is that single word would be represented by one vector no matter what the context was. For example, bank (a financial establishment) and bank (the land alongside a river) both would have the same word vector representation. Several NLP researchers (Peters et. al., 2017; McCann et. al., 2017) argue why not give it an embedding based on the context it's used in? to both capture the word meaning in that context as well as other contextual information. Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it, an embedding. Rather than a dictionary of words and their corresponding vectors, ELMo analyses words within the context that they are used. It is also character based, allowing the model to form representations of out-of-vocabulary words. This means that the way ELMo is used is quite different to word2vec. Rather than having a dictionary 'look-up' of words and their corresponding vectors, ELMo instead creates vectors on-the-fly by passing text through the deep learning model. It uses three layers bi-directional LSTM trained as a language model to be able to create those embeddings. ELMo gained its language understanding from being trained to predict the next word in a sequence of words. This is convenient because we have vast amounts of text data that such a model can learn from without needing labels. ELMo comes up with the contextualized embedding through grouping together the hidden states (and initial embedding). First, all the hidden layers for the forward and backward language model are concatenated together, then multiply each concatenated vector by a weight based on the mask. Finally, summing up all the weighted vector to generate the contextualized embedding. The pre-trained model is available to download from TF-Hub at (https://tfhub.dev/google/elmo/2). This pre-trained ELMo was trained on the 1 Billion Word Benchmark.

5.3. Contextual Word Embeddings: BERT

BERT follow the same approach as ELMo. Its start by training a general-purpose "language modeling" on a large text corpus (ex: Wikipedia), and then use that model for downstream NLP tasks that we care about (ex: question answering). BERT's key technical innovation is applying the bidirectional training of Transformer (Vaswani et. al., 2017), an attention model, to language modelling. In its vanilla form, Transformer includes two separate mechanisms: an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder is necessary. The input of the transformer's encoder is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index. When training language models, there is a challenge of defining a prediction goal. Many models predict the next word in a sequence, a directional approach which inherently limits context learning. To overcome this challenge, BERT uses two training strategies: 1) Masked LM (MLM): before feeding word sequences into BERT, some of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. 2) Next Sentence Prediction (NSP): In the training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. BERT's authors offer two pretrained model size for BERT: BERT base and BERT large). Both model sizes have a large number of transformer's encoder layers (i.e., Transformer Blocks); 12 for the base version, and 24 for the large version. These also have larger feedforward-networks (768 and 1024 hidden units respectively), and 12 and 16 attention heads respectively. Each encoder layer would have an embedding representation for each token in the text sequence. Like ELMo, we use all these embedding layers of the pre-trained BERT to create contextualized word embeddings. Pre-trained models for BERT are available to download from (https://github.com/google-research/bert#pre-trained-models).

5.4. Sentence Embeddings: DAN-Based USE

While word embeddings such as word2vec or GloVe try to embed a single word into a high dimensional vector, Universal Sentence Encoder (USE) (Cer et. al., 2018) try to embed not only words but phrases, sentences, and short paragraphs. USE takes variable length English text as input and outputs a 512-dimensional vector. Two different approaches had been utilized for encoding sentences into embedding vectors. One makes use of a deep averaging network (DAN) (Iyyer et. al., 2015) while the other uses Transformer (Vaswani et. al., 2017) architecture, which we will be discussed in the next section. Both variations of this model were trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The main difference between USE and word embeddings is USE was trained on a number of natural language prediction tasks that require modeling the meaning of word sequences rather than just individual words. The DAN based USE makes use of a deep averaging network (DAN) whereby input embeddings for words and bi grams are first averaged together and then passed through a feedforward deep neural network (DNN) to produce sentence embeddings. The main advantage of the DAN encoder, over the Transformer based USE, is that compute time is linear in the length of the input sequence. The pre-trained model is available to download from TF-Hub at (https://tfhub.dev/google/universal-sentence-encoder/2).

5.5. Sentence Embeddings: Transformer-Based USE

Just like the DAN-based USE, the Transformer based USE take as an input English strings and produce as output a fixed dimensional embedding representation of the string. This Model is based on the transformer architecture with the aim of high accuracy at the cost of greater model complexity and resource consumption. The original Transformer model constitutes an encoder and decoder, but this model uses its encoder part only. The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. They also employed a residual connection around each of the two sub-layers, followed by layer normalization. Since the model contains no recurrence and no convolution, for the model to make use of the order of the sequence, it must inject some information about the relative or absolute position of the tokens in the sequence, that is what the "positional encodings" does. The pre-trained model is available to download from TF-Hub at (https://tfhub.dev/google/universal-sentence-encoder-large/3). Both models (Dan-based and Transformer-based USE) were trained with the Stanford Natural Language Inference (SNLI) corpus. The SNLI corpus is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference (NLI), also known as recognizing textual entailment (RTE). Essentially, the models were trained to learn the semantic similarity between the sentence pairs.

5.6. Experiment and Results

The first experiment was to evaluate the context-free word embeddings (i.e., word2vec and GloVe). For representing the input, we used word embeddings to represent each word and then concatenate all the words' vectors in a sentence to constrict one vector for each sentence. We experimented using two different pretrained word embeddings (word2vec and GloVe), in addition to word embeddings trained on the entire Enron Email datasets using the word2vec algorithm. We used the word2vec algorithm's implementation provided by genism [39] with the following hyper-parameters (windows size = 5 and dimension size = 100), to obtain word vectors for 94,673 unique words. As shown in Table 5.1, the model using Google's pre-trained word2vec gave the best result. We are using F1 score to report our results, F1 score is the harmonic average of the precision and recall.

TABLE 5.1: FIRST MODEL USING DIFFERENT CONTEXT-FREE EMBEDDINGS

Embeddings	**Accuracy**	**Precision**	**Recall**	**F_1**
Enron embeddings	80.91%	82%	81%	0.81
Glove embeddings	77.82%	78%	78%	0.78
Word2vec embedding	**82.02%**	**82%**	**82%**	**0.82**

For the second model, we used pre-trained ELMo to represent each sentence into a 1024-dimensional vector. Then, we used a neural network consist of 256-relu nodes dense layer followed by a one-sigmoid node dense layer to classify the sentence's intent into "to do" or "to read." We trained the model for 10 epochs on 9055 labeled sentences using the following hyperparameters (optimizers: Adam, learning rate: 0.001, batch size: 32, and loss: binary cross entropy). Then, we tested the model on the remaining 2264 labeled sentences.

For the third model, we used the pre-trained BERT model to represent each sentence into a 768-dimensional vector. Then, we used a neural network consist of 256-relu nodes dense layer followed by a one-sigmoid node dense layer to classify the sentence's intent into "to do" or "to read." We trained the model for 10 epochs on 9055 labeled sentences using the following hyperparameters (optimizers: Adam, learning rate: 0.001, batch size: 32, and loss: binary cross entropy). Then, we tested the model on the remaining 2264 labeled sentences. As shown in Table 5.2, the model using pre-trained ELMo embeddings produce the best results as measured by the overall model's accuracy, precision, recall, and F1 for both classes.

For the fourth model, we used pre-trained DAN-based USE to represent each sentence into a 512-dimensional vector. Then, we used a neural network consist of 256-nodes dense layer followed by a one-sigmoid node dense layer to classify the sentence's intent into "to do" or "to read." We trained the model for 10 epochs on 9055 labeled sentences using the following hyperparameters (optimizers: Adam, learning rate: 0.001, batch size: 32, and loss: binary cross entropy). Then, we tested the model on the remaining 2264 labeled sentences.

For the last model, we used pre-trained Transformer-based USE to represent each sentence into a 512-dimensional vector. Then, we used a neural network consist of 256-nodes dense layer followed by a one-sigmoid node dense layer to classify the sentence's intent into "to do" or "to read." We trained the model for 20 epochs on 9055 labeled sentences using the following hyperparameters (optimizers: Adam, learning rate: 0.001, batch size: 32, and loss: binary cross entropy). Then, we tested the model on the remaining 2264 labeled sentences.

TABLE 5.2: FIVE MODELS USING DIFFERENT PRE-TRAINED EMBEDDINGS

	Intent		**Deliver**		**Accuracy**	**F1**
	Precision	**Recall**	**Precision**	**Recall**		
Word2vec	0.86	0.86	0.75	0.75	82.02	0.82
ELMo	**0.91**	**0.94**	**0.88**	**0.84**	**90.1**	**0.90**
BERT	0.66	0.73	0.39	0.31	58.08	0.57
DAN USE	0.87	0.93	0.85	0.76	86.66	0.86
Transformer USE	**0.91**	0.9	0.83	**0.84**	88.16	0.88

5.7. Analysis

The main goal of our research, in this chapter, is to compare three types of embeddings: context-free word embeddings, contextual word embeddings, and sentence embeddings. For context-free word embeddings, we used pre-trained word2vec and GloVe embeddings. We also, experimenting with pre-trained word embeddings trained on the entire Enron Email Dataset using the word2vec algorithm. Among those three context-free word embeddings, the model using pre-trained word2vec from Google trained on the Google News dataset, gave the best result in our tasking of classing the intent of emails. We also used two different type of contextual word embeddings. One using bi-directional LSTM trained on the task of Language Modeling (ELMo) and the second utilize a bidirectional training of Transformer, an attention model, to language modelling (BERT). The first model, ELMo, performed exceptionally well comparing with BERT. We were able to obtain an accuracy of 90.10% comparing with 58.08% for BERT. Lastly, we experimented with using pretrained sentence embeddings to represent the input. We used two different version of Google Universal Sentence Encoder (DAN-based and Transformer-Based. The Transformer-Based USE performed better than DAN-based sentence embeddings. As shown in Table 5.2, among all the different types of word and sentence embeddings, the modeling using the ELMo embeddings gained the largest benefits of using pre-trained embeddings in our task of classing the intent of emails. We got an accuracy of 90.10%.

CHAPTER 6

CONCLUSIONS

The aim of this dissertation is to explore the email phenomenon and provide a scalable solution that addresses the problem of email overload. Our proposed classification model requires no handcrafted features engineering and utilize the Speech Act Theory to design a classification system that detect whether an email required an action (i.e., to do) or no action (i.e., to read). We were able to automate both the features extraction and the classification phases by using word embeddings, trained on the entire Enron Email dataset, to represent the input, in addition to a convolutional layer to capture local tri-gram features, followed by a LSTM layer to consider the meaning of a given feature (trigrams) with respect to some "memory" of words that could occur much earlier in the email.

Furthermore, by applying the principle of Occam's razor (i.e., law of parsimony), we were able to simplify the above proposed model by dropping the use of the CNN layer and showing that fine tuning a pre-trained Language Model on the Enron email dataset can achieve a comparable result. To the best of our knowledge this is the first attempt of using transfer learning to develop a deep learning model in the email domain.

Lastly, by experimenting with three different types of embeddings: context-free word embeddings (word2vec and GloVe), contextual word embeddings (ELMo and BERT), and sentence embeddings (DAN-based Universal Sentence Encoder and Transformer-based Universal Sentence Encoder), we found that using a contextual word embedding (i.e. ELMo) is sufficient to represent both the semantic and contextual meaning of the sentence to classify it into its appropriate class. This final version of the our model achieved the best so far with an accuracy of 90.10%.

A second application of the embeddings, we found, is to encode the user's log data into a character embedding and then using it to detect malicious insider activities (Saaudi et. al, 2018). The raw users' log data represent five user activities: "Log on/ Log off," "Connect / Disconnect," "website link," "Emails," and "local files". These users' activities were pre-processed to generate new textual session-based sequences which were in turn encoded into its char embeddings.

A third and final use of embeddings is in the domain of recommender system. We found that just like you shall know a word by the company it keeps, you shall know a product by the company it keeps. In another words, while a sentence is a group of words, a purchase order is a group of products. Building on this assumption, we used the same algorithm we used to train our Enron embedding but this time we train it on millions of customers orders by replacing each product with its product ID. After training, we obtained an embedding for each product ID. Applying Cosine similarity on pair of products embeddings, result in a score of how similar these two products are.

REFERENCES

Aberdeen, D., Pacovsky, O., & Slater, A. (2010). The learning behind gmail priority inbox.

Ailon, N., Karnin, Z. S., Liberty, E., & Maarek, Y. (2013, February). Threading machine generated email. In *Proceedings of the sixth ACM international conference on Web search and data mining* (pp. 405-414).

Alberts, I. (2013). Challenges of information system use by knowledge workers: The email productivity paradox. *Proceedings of the American Society for Information Science and Technology*, *50*(1), 1-10.

Alibadi, Z. H., Thamer, S. K., & Goerge, L. E. (2014). Design and Implementation of Email Agent System. *Journal of University of Babylon*, *22*(3), 966-974.

Ames, M., & Naaman, M. (2007, April). Why we tag: motivations for annotation in mobile and online media. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 971-980).

Austin, J. L. (1970). How to do things with words: the William James lectures delivered at Harvard University in 1955.

Ayodele, T., & Zhou, S. (2009). Applying machine learning techniques for e-mail management: solution with intelligent e-mail reply prediction. *Journal of Engineering and Technology Research*, *1*(7), 143-151.

Balasubramanyan, R., Carvalho, V. R., & Cohen, W. (2008). Cutonce-recipient recommendation and leak detection in action. In *AAAI-2008, Workshop on Enhanced Messaging*.

Bälter, O., & Sidner, C. L. (2002, October). Bifrost Inbox Organizer: Giving users control over the inbox. In *Proceedings of the second Nordic conference on Human-computer interaction* (pp. 111-118).

Bälter, O. (2000, April). Keystroke level analysis of email message organization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 105-112).

Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. and Grinter, R.E., 2005. Quality versus quantity: E-mail-centric task management and its relation with overload. Human–Computer Interaction, 20(1-2), pp.89-138.

Bird, S., & Loper, E. (2004). NLTK: The Natural Language Toolkit In Proceedings 42nd. In *Meeting of the Association for Computational Linguistics (Demonstration Track) pp* (pp. 214-17).

Boardman, R., & Sasse, M. A. (2004, April). " Stuff goes into the computer and doesn't come out" a cross-tool study of personal information management. In *Proceedings of the SIGCHI conference on Human factors in computing systems*(pp. 583-590).

Brutlag, J. D., & Meek, C. (2000, June). Challenges of the email domain for text classification. In *ICML* (Vol. 2000, pp. 103-110).

Carvalho, V. R., & Cohen, W. (2007). Recommending recipients in the enron email corpus. *Machine Learning*.

Carvalho, V. R., & Cohen, W. W. (2005, August). On the collective classification of email" speech acts". In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 345-352).

Carvalho, V. R., & Cohen, W. W. (2007, April). Preventing information leaks in email. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (pp. 68-77). Society for Industrial and Applied Mathematics.

Carvalho, V. R., & Cohen, W. W. (2008, March). Ranking users for intelligent message addressing. In *European Conference on Information Retrieval* (pp. 321-333). Springer, Berlin, Heidelberg.

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Sung, Y. H. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

Cohen, W., Carvalho, V., & Mitchell, T. (2004, July). Learning to classify email into “speech acts”. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing* (pp. 309-316).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), 2493-2537.

CompTIA, 2015. Organizations changing strategies and tactics as security environment gets more complex, new CompTIA study finds. [Online] Available at: https://www.comptia.org/about- us/newsroom/press-r eleases/2015/03/31/organizations- changing-strategies-and-tactics-as-security-environment- gets-more-complex-new-comptia-study-finds.

ComputerWeekly, 2016. Human error causes more data loss than malicious attacks. [Online] Available at: http://www.computerweekly.com/news/450297535/Hu man-error-causes-more-data-loss-than-malicious-attacks

Crawford, E., Kay, J., & McCreath, E. (2002, January). An intelligent interface for sorting electronic mail. In *Proceedings of the 7th international conference on Intelligent user interfaces* (pp. 182-183).

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

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

Dabbish, L., Kraut, R., Fussell, S., & Kiesler, S. (2004). To reply or not to reply: Predicting action on an email message. In *ACM 2004 Conference. Citeseer*.

Dabbish, L., Venolia, G., & Cadiz, J. J. (2003, April). Marked for deletion: An analysis of email data. In *CHI'03 Extended Abstracts on Human Factors in Computing Systems* (pp. 924-925).

Dabbish, L. A., & Kraut, R. E. (2006, November). 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).

Dabbish, L. A., Kraut, R. E., Fussell, S., & Kiesler, S. (2005, April). Understanding email use: predicting action on a message. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 691-700).

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391-407.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Di Castro, D., Karnin, Z., Lewin-Eytan, L., & Maarek, Y. (2016, February). You've got mail, and here is what you could do with it! analyzing and predicting actions on email messages. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (pp. 307-316).

Di Sorbo, A., Panichella, S., Visaggio, C. A., Di Penta, M., Canfora, G., & Gall, H. C. (2015, November). Development emails content analyzer: Intention mining in developer discussions (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 12-23). IEEE.

Dredze, M., Lau, T., & Kushmerick, N. (2006, January). Automatically classifying emails into activities. In *Proceedings of the 11th international conference on Intelligent user interfaces* (pp. 70-77).

Dredze, M., Wallach, H. M., Puller, D., & Pereira, F. (2008, January). Generating summary keywords for emails using topics. In *Proceedings of the 13th international conference on Intelligent user interfaces* (pp. 199-206).

Ducheneaut, N., & Bellotti, V. (2001). E-mail as habitat: an exploration of embedded personal information management. *interactions*, *8*(5), 30-38.

Edmunds, A., & Morris, A. (2000). The problem of information overload in business organisations: a review of the literature. *International journal of information management*, *20*(1), 17-28.

Ekstrom, R. B., Dermen, D., & Harman, H. H. (1976). *Manual for kit of factor-referenced cognitive tests* (Vol. 102). Princeton, NJ: Educational testing service.

Elsweiler, D., Baillie, M., & Ruthven, I. (2008). Exploring memory in email refinding. *ACM Transactions on Information Systems (TOIS)*, *26*(4), 1-36.

Fahlman, S. E. (2006, August). Marker-passing inference in the scone knowledge-base system. In *International Conference on Knowledge Science, Engineering and Management* (pp. 114-126). Springer, Berlin, Heidelberg.

Faulring, A., Myers, B., Mohnkern, K., Schmerl, B., Steinfeld, A., Zimmerman, J., ... & Siewiorek, D. (2010, February). Agent-assisted task management that reduces email overload. In *Proceedings of the 15th international conference on Intelligent user interfaces* (pp. 61-70).

Fisher, D., Brush, A. J., Gleave, E., & Smith, M. A. (2006, November). 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).

Franovic, T., & Šnajder, J. (2012). Speech Act Based Classification of Email Messages in Croatian Language.

Freed, M., Carbonell, J. G., Gordon, G. J., Hayes, J., Myers, B. A., Siewiorek, D. P., ... & Tomasic, A. (2008, July). RADAR: A Personal Assistant that Learns to Reduce Email Overload. In *AAAI* (Vol. 8, pp. 1287-1293).

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*, 345-420.

Goldberg, Y., 2017. Neural network methods for natural language processing. Synthesis Lectures on Human Language Technologies, 10(1), pp.1-309.

Graus, D., Van Dijk, D., Tsagkias, M., Weerkamp, W., & De Rijke, M. (2014, July). Recipient recommendation in enterprises using communication graphs and email content. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (pp. 1079-1082).

Grbovic, M., Halawi, G., Karnin, Z., & Maarek, Y. (2014, November). 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).

Grevet, C., Choi, D., Kumar, D., & Gilbert, E. (2014, April). Overload is overloaded: email in the age of Gmail. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 793-802).

Gwizdka, J., & Chignell, M. (2004). Individual differences and task-based user interface evaluation: a case study of pending tasks in email. *Interacting with computers*, *16*(4), 769-797.

Gwizdka, J. (2002, September). TaskView: design and evaluation of a task-based email interface. In *Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research* (p. 4). IBM Press.

Gwizdka, J. (2004, April). Email task management styles: the cleaners and the keepers. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 1235-1238).

Hailpern, J., Asur, S., & Rector, K. (2014, October). AttachMate: Highlight extraction from email attachments. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (pp. 107-116).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

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

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016). Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kalman, Y. M., & Ravid, G. (2014, February). Email inbox management by information overloaded users. In *Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing* (pp. 185-188).

Kerr, B. (2003, October). Thread arcs: An email thread visualization. In *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714)* (pp. 211-218). IEEE.

Khoussainov, R., & Kushmerick, N. (2005, July). Email Task Management: An Iterative Relational Learning Approach. In *CEAS*.

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

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-aware neural language models, AAAI, 2741–2749. *arXiv preprint arxiv:1508.06615*.

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

Koren, Y., Liberty, E., Maarek, Y., & Sandler, R. (2011, August). 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).

Kushmerick, N., Lau, T., Dredze, M., & Khoussainov, R. (2006, July). Activity-centric email: A machine learning approach. In *AAAI* (pp. 1634-1637).

Kudugunta, S., & Ferrara, E. (2018). Deep neural networks for bot detection. *Information Sciences*, *467*, 312-322.

Lampert, A., Dale, R., & Paris, C. (2006, November). Classifying speech acts using verbal response modes. In *Proceedings of the Australasian Language Technology Workshop 2006* (pp. 34-41).

Lampert, A., Dale, R., & Paris, C. (2008). The nature of requests and commitments in email messages. In *Proceedings of the AAAI Workshop on Enhanced Messaging*(pp. 42-47).

Lampert, A., Dale, R., & Paris, C. (2010, June). 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).

Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196).

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), p.436.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Lockerd, A., & Selker, T. (2003, September). DriftCatcher: The Implicit Social Context of Email. In *INTERACT*.

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

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems* (pp. 6294-6305).

McKinsey Global Institute & International Data Corp, 2012. The social economy: Unlocking value and productivity through social technologies. [Online] Available at: http://www.mckinsey.com/industries/high- tech/our-insights/the-social-economy McShane, S. and Von Glinow, M. 2015. Organizational Behavior, 7th edition. McGraw-Hill. ISBN: 978-0-07-786258-9.

Merity, S., Keskar, N. S., & Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*, 3111-3119.

Millen, D., Feinberg, J., & Kerr, B. (2005). Social bookmarking in the enterprise. *Queue*, *3*(9), 28-35.

Minkov, E., Balasubramanyan, R., Cohen, W. W., & Dep, M. L. (2008). Activity-centric search in email. In *Enhanced Messaging Workshop, AAAI*.

Muller, M. J., & Gruen, D. M. (2002). Collaborating within–not through–email: Users reinvent a familiar technology. *Poster at CSCW*.

Neustaedter, C., Brush, A. B., Smith, M. A., & Fisher, D. (2005, July). The Social Network and Relationship Finder: Social Sorting for Email Triage. In *CEAS*.

Neustaedter, C., Brush, A. B., & Smith, M. A. (2005, April). Beyond" from" and" received" exploring the dynamics of email triage. In *CHI'05 extended abstracts on Human factors in computing systems* (pp. 1977-1980).

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning*, *39*(2-3), 103-134.

Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature visualization. *Distill*, *2*(11), e7.

Pal, C., & McCallum, A. (2006, July). CC Prediction with Graphical Models. In *CEAS*.

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

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Peters, M. E., Ammar, W., Bhagavatula, C., & Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Qadir, A., & Riloff, E. (2011, July). Classifying sentences as speech acts in message board posts. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (pp. 748-758).

Qadir, A., Gamon, M., Pantel, P., & Hassan, A. (2016, June). Activity modeling in email. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*(pp. 1452-1462).

Quirk, C., Choudhury, P., Gao, J., Suzuki, H., Toutanova, K., Gamon, M., ... & Cherry, C. (2019). MSR SPLAT, a language analysis toolkit.

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

Rohall, S. L., Gruen, D., Moody, P., Wattenberg, M., Stern, M., Kerr, B., ... & Wilcox, E. (2004, April). ReMail: a reinvented email prototype. In *CHI'04 extended abstracts on Human factors in computing systems* (pp. 791-792).

Saaudi, A., Al-Ibadi, Z., Tong, Y., & Farkas, C. (2018, December). Insider Threats Detection Using CNN-LSTM Model. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 94-99). IEEE.

Schuff, D., Turetken, O., & D'Arcy, J. (2006). A multi-attribute, multi-weight clustering approach to managing “e-mail overload”. *Decision Support Systems*, *42*(3), 1350-1365.

Segal, R. B., & Kephart, J. O. (1999, April). MailCat: An intelligent assistant for organizing e-mail. In *Proceedings of the third annual conference on Autonomous Agents* (pp. 276-282).

Sevinc, G., & D'Ambra, J. (2010). The influence of self-esteem and locus of control on perceived email overload.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sofershtein, Z., & Cohen, S. (2015, August). Predicting email recipients. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*(pp. 761-764). IEEE.

Song, M., Halsey, V., & Burress, T. (2008). *The hamster revolution: How to manage your email before it manages you*. Berrett-koehler publishers.

Sorower, M. S., Slater, M., & Dietterich, T. G. (2015, November). Improving Automated Email Tagging with Implicit Feedback. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (pp. 201-211).

Spira, J. B., & Goldes, D. M. (2007). Information overload: We have met the enemy and he is us. *Basex Inc*.

Surendran, A. C., Platt, J. C., & Renshaw, E. (2005, July). Automatic Discovery of Personal Topics to Organize Email. In *CEAS*.

Tang, J. C., Wilcox, E., Cerruti, J. A., Badenes, H., Nusser, S., & Schoudt, J. (2008). Tag-it, snag-it, or bag-it: combining tags, threads, and folders in e-mail. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems* (pp. 2179-2194).

Teevan, J., Alvarado, C., Ackerman, M. S., & Karger, D. R. (2004, April). The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 415-422).

Twitchell, D. P., Adkins, M., Nunamaker, J. F., & Burgoon, J. K. (2004). Using speech act theory to model conversations for automated classification and retrieval. In *Proceedings of the International Working Conference Language Action Perspective Communication Modelling (LAP 2004)* (pp. 121-130).

Tyler, J. R., & Tang, J. C. (2003). When can I expect an email response? A study of rhythms in email usage. In *ECSCW 2003* (pp. 239-258). Springer, Dordrecht.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*, 5998-6008.

Venolia, G. D., & Neustaedter, C. (2003, April). Understanding sequence and reply relationships within email conversations: a mixed-model visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 361-368).

Team, V. R. (2015). 2015 data breach investigations report.

Vishnu, M. S., Damle, D., Bhaumik, D., & Sahu, D. (2013, November). Semantic emails: agent technology in email systems. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration* (pp. 419-420).

Wattenberg, M., Rohall, S. L., Gruen, D., & Kerr, B. (2005). E-mail research: Targeting the enterprise. *Human–Computer Interaction*, *20*(1-2), 139-162.

Wendt, J. B., Bendersky, M., Garcia-Pueyo, L., Josifovski, V., Miklos, B., Krka, I., ... & Ravi, S. (2016, February). Hierarchical label propagation and discovery for machine generated email. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (pp. 317-326).

Whittaker, S., Bellotti, V., & Moody, P. (2005). Introduction to this special issue on revisiting and reinventing e-mail. *Human–Computer Interaction*, *20*(1-2), 1-9.

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

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

Whittaker, S., Bellotti, V., Gwizdka, J., Jones, W., & Teevan, J. (2007). Personal information management. *American: university of cambright*.

Whittaker, S., Jones, Q., Nardi, B., Creech, M., Terveen, L., Isaacs, E., & Hainsworth, J. (2004). ContactMap: Organizing communication in a social desktop. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *11*(4), 445-471.

Whittaker, S., Matthews, T., Cerruti, J., Badenes, H., & Tang, J. (2011, May). Am I wasting my time organizing email? A study of email refinding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 3449-3458).

Whittaker, S., Swanson, J., Kucan, J., & Sidner, C. (1997). TeleNotes: managing lightweight interactions in the desktop. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *4*(2), 137-168.

Yoo, S., Yang, Y., & Carbonell, J. (2011, October). Modeling personalized email prioritization: classification-based and regression-based approaches. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 729-738).

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, *13*(3), 55-7.