Identity and Graph-aware Framework for Email Reply Prediction

UNDERGRADUATE THESIS

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Harsh SHAH
ID No. 2017B3A30650P

Under the supervision of:

Prof. Surekha BHANOT
&
Dr. Kokil JAIDKA

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
December 2021

Declaration of Authorship

I, Harsh SHAH, declare that this Undergraduate Thesis titled, 'Identity and Graph-aware Framework for Email Reply Prediction' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 26/12/2021

Certificate

This is to certify that the thesis entitled, "*Identity and Graph-aware Framework for Email Reply Prediction*" and submitted by Harsh SHAH ID No. 2017B3A30650P in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

Supervisor	*Co-Supervisor*
Prof. Surekha BHANOT	Dr. Kokil JAIDKA
Professor	Assistant Professor
Electrical & Electronics Engineering,	Communications and New Media,
BITS Pilani, Pilani Campus	National University of Singapore
Date:	Date:

"Yesterday is history, Tomorrow is a mystery, and today is a gift. That's why they call it Present."

Master Oogway

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS

Abstract

Bachelor of Engineering (Hons.)

Identity and Graph-aware Framework for Email Reply Prediction

by Harsh SHAH

Email is a critical means for asynchronous communication, especially in the workplace. The way people within an organization communicate with each other depends on many factors, including but not limited to email content, historical context, and workplace relations. Prior work in characterizing email communication has focused mainly on linguistic cues but has ignored the signals of pairwise temporal context and workplace relationships that could offer dynamic signals about future behavior, such as replying to emails. This work examines a new dataset of 0.6 million corporate emails exchanged by employees in 2012-2015 to illustrate the importance of influence and interpersonal relationships in a graph-aware temporal framework for predicting enterprise email replies. Incorporating features encoding the sender's influence and likeability in the organizational network in a multimodal framework improves the performance accuracy compared to linguistic features alone. Apart from individual features, we also leverage information related to the relation between sender-receiver pairs, improving the accuracy of predicting subsequent email replies

Acknowledgements

I want to thank Dr. Kokil Jaidka for giving me the opportunity to work under her guidance. She guided me through the research work, helping me understand new concepts and improve my skills. Also, I would like to thank Prof. Surekha Bhanot, who encouraged me to pursue new ideas and provided valuable feedback.

I would like to express gratitude to the National University of Singapore for providing the resources to conduct the research. Additionally, I would like to thank the Electrical and Electronics Department, BITS Pilani for allowing me to pursue the thesis on a topic of my interest. Lastly, I want to express gratitude towards my parents, family and friends who have always been there to support me.

Contents

Declaration of Authorship i

Certificate ii

Abstract iv

Acknowledgements v

Contents vi

List of Figures viii

List of Tables ix

Abbreviations x

1 Introduction 1
- 1.1 Email Reply Prediction . . . 1
- 1.2 Commercial & Research Applications . . . 2
- 1.3 Literature Review . . . 2
- 1.4 Thesis Outline . . . 3

2 Background Concepts 4
- 2.1 Neural Networks . . . 4
  - 2.1.1 Neurons . . . 5
  - 2.1.2 Structure . . . 5
- 2.2 TF-IDF . . . 6
- 2.3 CNN . . . 7
- 2.4 RNN . . . 8
  - 2.4.1 LSTM . . . 9
  - 2.4.2 GRU . . . 10
- 2.5 Transfomers . . . 10
  - 2.5.1 Attention Mechanism . . . 10
  - 2.5.2 BERT . . . 13

2.5.3 RoBERTa . . . 13

3 Dataset Description **14**
3.1 Luxury-Standard Dataset . . . 14
3.2 Email Data . . . 15
3.3 Extracted Features . . . 15
3.3.1 LIWC . . . 15
3.3.2 Politeness . . . 16
3.3.3 SNA . . . 16
3.3.4 Interpersonal Context . . . 17
3.3.5 Influence Features . . . 18
3.3.6 Topic . . . 18
3.3.7 Number of Features . . . 18

4 Methodology **19**
4.1 Problem Statement . . . 19
4.2 Model Architecture . . . 19
4.3 Training Parameters . . . 21
4.3.1 Baseline Models . . . 21
4.3.2 Transformer Models . . . 22
4.4 Experiments . . . 22

5 Result and Discussion **23**
5.1 Results . . . 23
5.1.1 Full Data . . . 23
5.1.2 MultiModal Data . . . 24
5.2 Discussion . . . 24

6 Conclusion **25**
6.1 Recommendation for Future Research . . . 25
6.2 Conclusion . . . 26

A Code **27**

B Converting dataset from CSV to Graph **30**

Bibliography **32**

List of Figures

2.1 Vanilla Neural Network 5
2.2 Convolutional Neural Network 7
2.3 Recurrent Neural Network 8
2.4 LSTM 9
2.5 Bahdanau Attention 11
2.6 Transformer Architecture 12

4.1 Model Architecture 20

6.1 Data as Heterogeneous Graph 26

List of Tables

3.1 Dataset Description 14
3.2 Table Structure 15
3.3 LIWC 2015 Example 16
3.4 SNA Features 16
3.5 Feature Count 18

4.1 List of baseline models 21

5.1 Baseline Results - Full Data 23
5.2 Model Results - 18K Data 24

Abbreviations

SVC	**S**upport **V**ector **C**lassifier
DNN	**D**eep **N**eural **N**etwork
TF-IDF	**T**erm **F**requency - **I**nverse **D**ocument **F**requency
NLP	**N**atural **L**anguage **P**rocessing
CNN	**C**onvolutional **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
GNN	**G**raph **N**eural **N**etwork
GCN	**G**raph **C**onvolutional **N**etwork
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
RoBERTa	**R**obustly **O**ptimised **BERT** Pretraining **A**pproach
CSV	**C**omma **S**eperated **V**alues
LIWC	**L**inguistic **I**nquiry and **W**ord **C**ount
SNA	**S**ocial **N**etwork **A**nalysis

Dedicated to my Parents

Chapter 1

Introduction

1.1 Email Reply Prediction

Email is a critical means for asynchronous communication, especially in the workplace. In recent times, many corporations have started shifting to remote or hybrid work which involves a lot of communication between the employees over different channels out of which email is often the primary one. Accurately predicting whether a sent mail will receive a reply or not can help the involved parties manage their time and work efficiently.

The way people within an organization communicate with each other depends on many factors, including but not limited to email content, historical context, and workplace relations. Prior work in characterizing email communication has focused mainly on linguistic cues but has ignored the signals of pairwise temporal context and workplace relationships that could offer dynamic signals about future behavior, such as replying to emails. This work examines a new dataset of 0.6 million corporate emails exchanged by employees in 2012-2015 to illustrate the importance of influence and interpersonal relationships in a graph-aware temporal framework for predicting enterprise email replies.

Incorporating features encoding the sender's influence and likeability in the organizational network in a multimodal framework improves the performance accuracy compared to linguistic features alone. Apart from individual features, we also leverage information related to the relation between sender-receiver pairs, improving the accuracy of predicting subsequent email replies.

1.2 Commercial & Research Applications

Commercial products like Google Mail and Outlook are widely used by corporations for a majority of their work related communication. Incorporating the model proposed into these existing email services directly or via a browser add-on is not a very complex task. Based on who the sender and receiver are, the model can predict whether the current email body will result in a reply or not. This also allows the user to tweak the text until a satisfactory number is predicted as the probability of getting a reply.

While this work poses the problem as a binary classification task where a 0 and 1 mean not getting and getting a reply respectively, the model can also be modified to perform a regression task to predict the time it will take for the receiver to reply to that mail. Furthermore, if the replies of the receiver are present, one can also train a generative language model to generate contextual replies to a given mail.

1.3 Literature Review

A lot of prior work exists on the topic of email classification. Sarrafzadeh et al. [26] performed qualitative and quantitative study to find out how often emails are deferred or ignored. The Enron corpus[16] released in 2004 also dives into classifying emails. Apart from classification, there is also email intent extraction and analysis. Sappelli et al. [25] demonstrated that tasks and intents in emails can be extracted with high accuracy. Other work tries to rank all unread emails by ranking them according to their importance[1].

Looking at the specific topic of email reply probability prediction, Liu et al. [31] proposed a model to predict the likeliness and time that a recipient may take to reply to a certain mail. This work provided insights about importance of different features like number of attachments and email body length. Mukherjee et al. [21] decided to approach this problem as content-based recommendation. To achieve this, they compute user representation by looking at their inbox. Using this, they compute similarity between the email and the user's inbox, which helps improve the prediction accuracy.

Taking inspiration from prior research done on this topic, the proposed model incorporates not just the mail text and the user inbox when predicting the probabilty, but also looks at social graph, temporal and workplace relation related features.

1.4 Thesis Outline

The next few chapters cover the theory and concepts used in this work.

Chapter 2 contains the basics of various Deep Learning-based architectures used in Natural Language Processing, starting from Vanilla Neural Networks and building up to the current State-of-the-Art Transformer models like BERT.

Chapter 3 explores the dataset used in this work. A lot of numerical and categorical features apart from the text used in the model(s) have been derived from this data which are also discussed in this chapter.

Chapter 4 contains the details of the baseline and multimodal models which have been used to run the experiments

Chapter 5 has the results and discussion.

Chapter 6 concludes the work presented in the previous chapters and provides a direction for further research in this topic - particularly with Graph Neural Networks.

Chapter 2

Background Concepts

This chapter introduces the basic theory and mathematical concepts behind various Deep Learning model architectures. In particular, models and architectures which were or are State-of-the-Art in NLP have been discussed.

2.1 Neural Networks

Research in the field of Machine Learning has been growing exponentially in the past decade. A majority of this progress can be attributed to a class of models - Deep Neural Networks(DNNs), commonly referred to as Neural Networks.

Neural Networks have been around for a long time, with early work on this subject appearing as early as the late 1950s in the form of the perceptron model[23]. Another influential paper in this field is the 1986 paper, which introduced the backpropagation algorithm to update the weights of the network using partial derivatives[24]. The methods described in this paper are still being used in modern-day Deep Learning.

While this worked well for shallower networks, it was very difficult to train deeper networks due to the computational limits of that time. Because of this, DNNs did not gain that much traction until the early 2010s, when growth of data and increased computational power, resulted in a boom in this field.

2.1.1 Neurons

Neural Networks were developed to replicate the function of learning as a brain does. This is why their structure resembles connected neurons passing information, similar to how neurons fire in the human brain.

A neuron performs the same task as a mathematical function. It takes in a weighted sum of the inputs connected to it, adds a bias term and passes this sum through a non-linear function also called activation function. The equation is given below:

$$h = \underbrace{\sum_{i=1}^{k} W_i X_i}_{\text{Weighted Sum of Input Neurons}} + \overbrace{b}^{\text{Bias Term}} \tag{2.1}$$

$$Y = \underbrace{f}_{\text{Non-Linear Activation}}(h) = \text{Neuron Output} \tag{2.2}$$

2.1.2 Structure

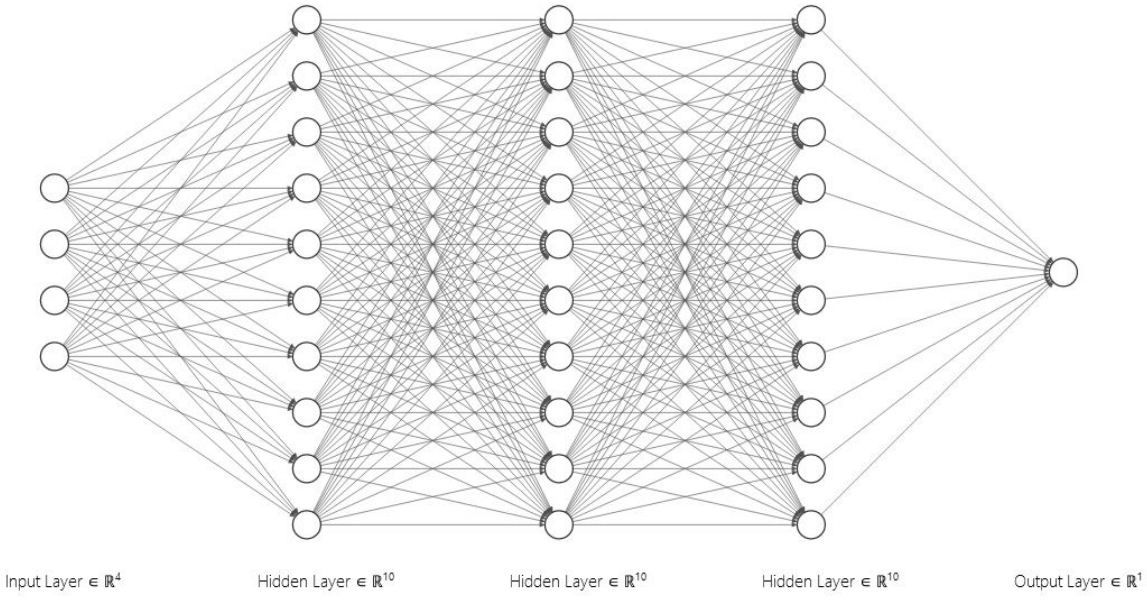


FIGURE 2.1: Basic Neural Network
Source: `https://alexlenail.me/NN-SVG/`

DNNs consists of mainly three components:

- Input Layer
- *n* number of Hidden Layers
- Output Layer

The input layer is the first layer in the network responsible for taking in the input. The information from this input layer gets passed on to the next hidden layer. This keeps on happening until the last hidden layer from which the information gets passed on to the output layer. At the output layer, the model output is used to calculate the loss, which is used to measure how far off the prediction of the network is as compared to the actual output. This loss then gets backpropagated through the neural network, updating the weights between the neurons.

Other models discussed in the following chapters take inspiration from the architecture of Neural Networks and introduce changes resulting in models like RNNs and CNNs. These models perform well on specific data like Images or Text.

2.2 TF-IDF

Term frequency – Inverse document frequency (TF-IDF) is a well known concept in Information Retrieval. Classifying documents, ranking web pages, and extracting relevant words from the corpus are some of the primary uses of TF-IDF. This method was introduced by Karen Spärck Jones in the 1972 article titled ”A Statistical Interpretation of Term Specificity in Retrieval”[13].

Given a corpus with multiple documents, TF-IDF has two parts - Term Frequency(TF), which looks at how often a word appears in a specific document, and Inverse Document Frequency(IDF) which looks at how many times the word appears in different documents from the corpus. The reason is, words like ”a” and ”the” - which appear multiple times in text, won’t be given much importance. But words that appear rarely will have a higher TF-IDF score. If a document contains the word ”artificial” multiple times, but does not appear this often in other documents, then this document is relevant to searches related to the word ”artificial”. It will also have a high TF-IDF score for the particular word.

The formula to compute TF-IDF of words in a corpus of documents is as follows[20]:

$$\mathrm{tf}(w,d) = \frac{f_{w,d}}{\sum_{w' \in d} f_{w',d}} = \text{Term Frequency for word } w \text{ in document } d \tag{2.3}$$

$$\mathrm{idf}(w,D) = \log \frac{N}{|\{d \in D : w \in d\}|} = \text{Inverse Document Frequency for word } w \text{ in corpus } D \tag{2.4}$$

where

$$\begin{aligned} f_{w,d} &= \text{Number of times } w \text{ appears in document } d \\ N &= \text{Total Number of Documents in the corpus} \\ |\{d \in D : w \in d\}| &= \text{Number of documents where } w \text{ appears} \end{aligned}$$

Combining both these equations, we get the formula for calculating the TF-IDF score for a word w in the corpus D:

$$\mathrm{tfidf}(w,d,D) = \mathrm{tf}(w,d) \cdot \mathrm{idf}(w,D) \tag{2.5}$$

2.3 CNN

Convolutional Neural Networks(CNNs) are a class of neural networks that are used primarily for image recognition and classification. They were developed by Yann LeCunn in the 1990s for Optical Character Recognition(OCR) related tasks[17].

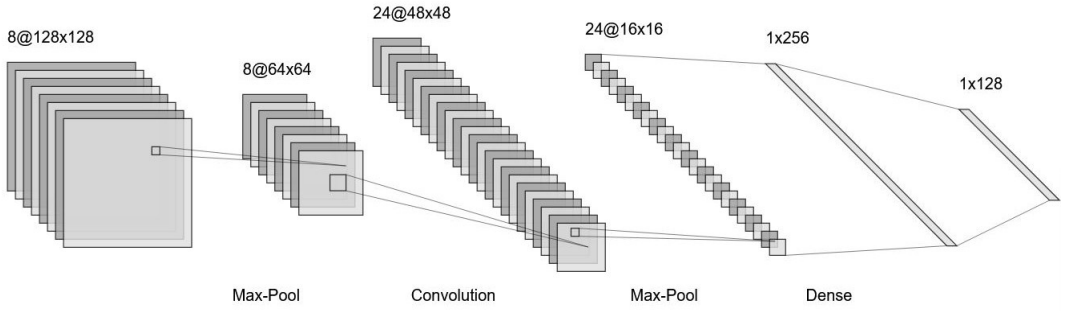


FIGURE 2.2: CNN
Source: https://alexlenail.me/NN-SVG/LeNet.html

Each layer in a CNN has filters which slide on the data from input/previous layer. Taking example of an grayscale image matrix as input to a CNN, the sliding window in this case would be a 2-D matrix of

values which slides over the 2-D image detecting features like edges and lines. As we go towards the latter layers in a CNN, the features being detected go from edges and lines to ears and nose.

2.4 RNN

Recurrent Neural Network(RNN) is a class of Neural Network which is primarily used for sequential data like Text, Audio and Time-Series data. Figure 2.3 shows the architecture of an RNN cell.

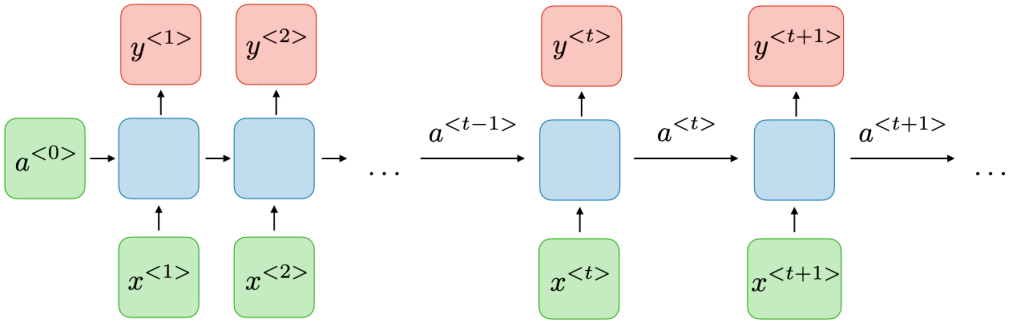


FIGURE 2.3: RNN Architecture
Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

While neurons in a vanilla DNN can look at input data or data from the previous layer, the neurons in an RNN can go one step further. They can look at both - neurons from the last layer and the neuron output from the previous timestep. This small change in RNNs allows them to retain information from the past while calculating the output at a given timestep.

$$\begin{aligned} a^{<t>} &= f(W_{ax}x^{<t>} + W_{ay}y^{<t-1>} + b_a) \\ y^{<t>} &= g(W_{ya}a^{<t>} + b_y) \end{aligned} \tag{2.6}$$

where

$$\begin{aligned} f, g &= \text{Non-Linear Activation} \\ x^{<t>} &= t^{th} \text{ term of the input sequence} \\ a^{<t>} &= \text{Activation output at timestep = } t \\ y^{<t>} &= \text{Neuron output at timestep = } t \\ W_{ax}, W_{ay}, W_{ya} &= \text{Weights} \\ b_a, b_y &= \text{Biases} \end{aligned}$$

Based on the above equations, we can see how the output at a previous timestep, i.e. $y^{<t-1>}$, is used to calculate $y^{<t>}$. Theoretically, this means that past information will be used. But for practical use cases, this results in an issue known as Vanishing Gradient when performing a backward pass. To update weights in RNNs, Backpropagation through Time or BPTT is used. The derivation of the weight update rule in RNN shows that for input sequence length = *N*, the derivative term contains weight raised to the power *N*. This means that if the weight term is either very small or very large, the gradient will tend to zero or explode, respectively.

Architectures like LSTM and GRU addressed and corrected this shortcoming in RNNs.

2.4.1 LSTM

To avoid the problem of Vanishing Gradient, Schmidhuber and Sepp proposed a new kind of architecture[12] called Long-Short Term Memory (LSTM).Apart from the primary recurrence relation present in RNNs, LSTMs also have multiple gating mechanisms which allow it to choose the amount of information from the past that gets carried forward.

There are three different gates in an LSTM cell:

- Forget Gate
- Input Gate
- Output Gate

The forget gate controls the amount of information that gets forgotten, the update gate controls the amount of information that gets carried forward, and the output gate controls the amount of information that gets passed on to the next layer.

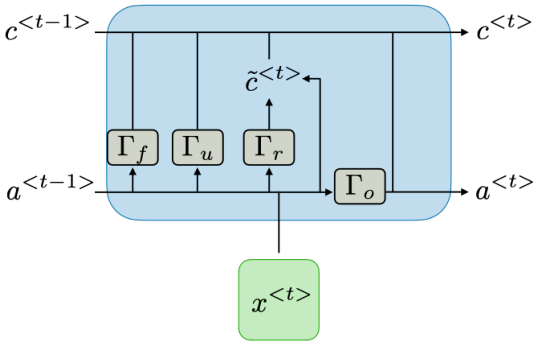


FIGURE 2.4: LSTM Architecture
Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

The following equations show the working of an LSTM cell:

$$\begin{aligned}
\Gamma_r &= \tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + +b_c\right) = \tilde{c}^{<t>} \\
\Gamma_u &= \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + +b_u\right) = \text{Update Gate} \\
\Gamma_f &= \sigma\left(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + +b_f\right) = \text{Forget Gate} \\
\Gamma_o &= \sigma\left(W_{oa}a^{<t-1>} + W_{ox}x^{<t>} + +b_o\right) = \text{Output Gate} \\
c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} = \text{Memory Cell} \\
a^{<t>} &= \Gamma_o * \tanh\left(c^{<t>}\right) = \text{Activation Output}
\end{aligned} \tag{2.7}$$

where

$\sigma =$ Sigmoid Function

$*=$ Hadamard Product (Element-Wise Multiplication)

2.4.2 GRU

Gated Recurrent Units (GRUs) were proposed in 2014[5]. They are similar to LSTMs but with a simpler architecture. Instead of the 3 gates that LSTM has, GRU only has 2 gates: Update Gate and Reset Gate.

This simplification in GRU makes it both, easier and faster to train compared to LSTM. The equations below show how information flows through a GRU cell at any given timestep:

$$\begin{aligned}
\Gamma_r &= \sigma\left(W_{rx}x^{<t>} + W_{ra}a^{<t-1>} + b_r\right) = \text{Reset Gate} \\
\Gamma_u &= \sigma\left(W_{ux}x^{<t>} + W_{ua}a^{<t-1>} + b_u\right) = \text{Update Gate} \\
\tilde{a}^{<t>} &= \tanh\left(W_{ax}x^{<t>} + W_{ac}(\Gamma_r * a^{<t-1>}) + b_c\right) = \text{Candidate Activation} \\
a^{<t>} &= \Gamma_u * \tilde{a}^{<t>} + (1 - \Gamma_u) * a^{<t-1>} = \text{Activation Output}
\end{aligned} \tag{2.8}$$

2.5 Transfomers

2.5.1 Attention Mechanism

The Encoder-Decoder model worked well for machine translation tasks. The intuition behind the model was to use one RNN to encode and the other one to decode. The Encoder would encode the input sequence

to generate meaningful embeddings and the other RNN - called the Decoder would use this embedding to generate translated output sequence.

But for purely RNN-based models, the model performance deteriorated with increasing input length.This changed with attention mechanism. In 2016, a paper titled "Neural Machine Translation by Jointly Learning to Align and Translate" [3] showed the power of attention for machine translation tasks.

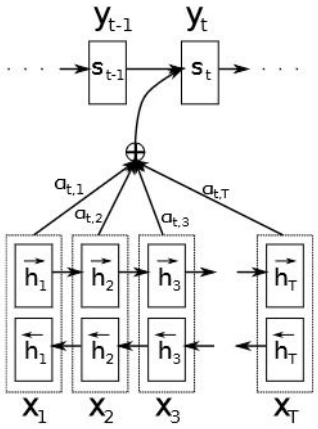


FIGURE 2.5: Bahdanau Attention
Source: Figure 1 from "Neural Machine Translation by Jointly Learning to Align and Translate"[3]

The following equations show how context vector is calculated when decoding the output sequence:

$$
\begin{aligned}
e_{ij} &= v_a^\top \tanh\left(W_{as} s_{i-1} + W_{ah} h_j\right) = \text{Alignment Model} \\
\alpha_{ij} &= \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} = \text{Attention Weight} \\
c_i &= \sum_{j=1}^{T_x} \alpha_{ij} h_j = \text{Context Vector}
\end{aligned}
\tag{2.9}
$$

where

$$
\begin{aligned}
h_j &= j^{th} \text{ Hidden state representation from Encoder} \\
s_i &= i^{th} \text{ Hidden state representation from Decoder} \\
\alpha_{ij} &= \text{Importance of } h_j \text{ when generating } s_i \text{ and } y_i
\end{aligned}
$$

While there are other kinds of attention mechanisms[19], the one being used widely today is Multi-Head Self-Attention used in the Transformer architecture[28]. Self-Attention looks at each word in the input

sequence and computes how important other words in the input are when computing the representation for that particular word. For this the input is first represented as a matrix of shape (n, e) where $n =$ Length of Input and $e =$ Embedding dimension.

From this input, we get three matrices : Key(K), Query(Q) and Value(V) by multiplying the input with weight matrices. The next equation shows how Self-Attention is computed using these matrices:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.10}$$

The dot product between Q and K^T computes the similarity between the current word and all the other words in the input sequence.$\sqrt{d_k}$ (dimension of Key matrix) is used to scale down the dot product so that the gradient does not become unstable for large values of n. Softmax converts this to a probability distribution which adds up to 1. The final multiplication with Value matrix results in those words retaining their embedding whose dot-product score is high, which results in meaningful and contextual embeddings.

This shows the working of a single Self-Attention head. Transformers have multiple such heads, the outputs of which are concatenated and transformed into a compatible dimension. Another feature of transformer models is that they use multiple encoders and decoders stacked on top of each other as their Encoder and Decoder blocks. Figure 2.6 shows the model architecture of a Transformer as provided by the authors in the original paper.

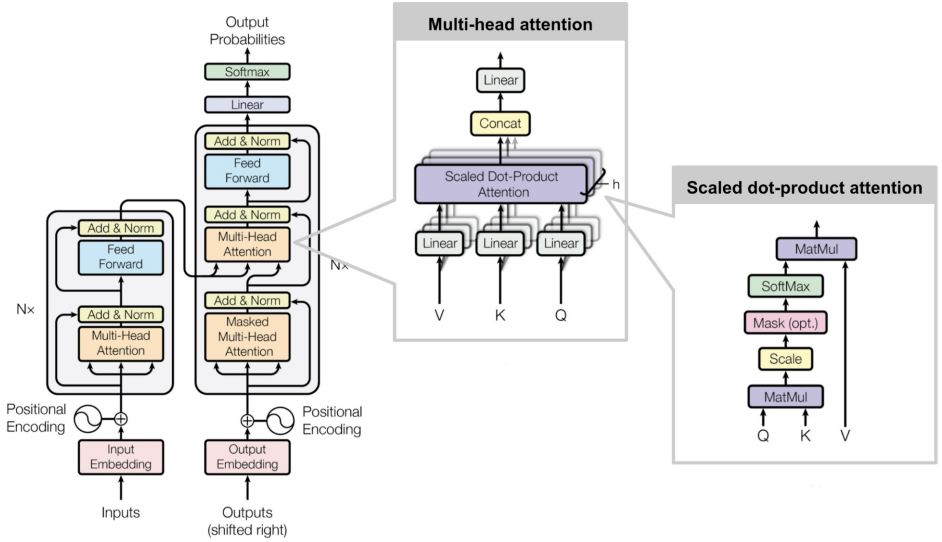


FIGURE 2.6: Transformer Architecture
Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

2.5.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) was originally published by Google AI Language in 2018 where they used the transformer architecture for the task of language modeling[8]. One of the key features of BERT is that it stacks only Encoder blocks on top of each other.

BERT is deeply bidirectional i.e. the model learns from both left-to-right and right-to-left while going over input sentences. Also the self-attention combined with multi-head attention in each transformer encoder block helps to generate more contextual embeddings by taking into account other words in the sentence when encoding a particular word. Multiple attention heads, deep bidirectional nature coupled with the combined training task of Masked Language Modelling and Next Sentence Prediction are some of the reasons why the BERT language model has a really good understanding of the structure and semantics of English language. Due to these reasons, a single dense layer on top of the output of the BERT language model performs exceptionally well on many NLP tasks.

2.5.3 RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) is an alternate approach to training BERT which not only improves it performance but also results in easier training[18].

BERT masks token randomly during preprocessing stage resulting in a static mask. And to avoid using the same masked sequence every epoch, the training data for BERT was duplicated multiple times so that each sequence can be masked in multiple ways. But this becomes difficult to implement with large datasets.To addresss this, authors of RoBERTa propose a dynamic masking scheme which means that masking takes place before a sentence gets selected into a minibatch to be fed into the model.

Chapter 3

Dataset Description

3.1 Luxury-Standard Dataset

The data used in this work is the Luxury-Standard Dataset[27]. Luxury Inc. and Standard Inc. were two rival good manufacturing firms in the U.S. that merged to form the new Luxury-Standard Inc. The original data was collected post-merger to understand how the behavioral changes in the employees could influence the chances of success of the newly formed Luxury-Standard.

The data was collected between 2012 and 2015 by surveying the employees on various aspects like job security, organizational attachment, relation with other employees, and so on. Apart from this, the dataset also contains information on emails exchanged by the employees in this period. All the data which can be used to identify an employee or has sensitive information has been de-identified.

TABLE 3.1: Dataset Description

Aspect	**Value**
Emails exchanged	665120
Employees	4320
Time period in days	252
Mails per Employee	153
Mails per day	2639

3.2 Email Data

The dataset contains around 0.6 million emails exchanged by the employees. This information is present as a CSV with each row corresponding to an email exchanged. Each row also contains metadata related to that email, including but not limited to the email body, timestamp, sender, receiver, thread_id and reply. Table 3.2 contains the details of columns present in the CSV.

TABLE 3.2: Table Structure

Column Name	Data Type	Description
pstid	String	Unique Identifier for each mail
date_	DateTime	Timestamp
thread_id	Integer	Thread Id to which the mail belongs to
id	Integer	ID of an email within a specific thread
message	String	Text in the mail body
sender	String	Unique identifier for the sender
receiver	String	Unique identifier for the receiver
reply	Bool	0 if the mail has no reply, otherwise 1

A small subset of the employees was surveyed about their relations with other colleagues. Employees rated their coworkers on traits like friendliness, communication frequency, and whether they would avoid a particular coworker or not. Merging this data with the original data results in a CSV with approximately 18,000 emails where apart from the columns mentioned above, details regarding the interpersonal relations of the sender and the receiver are also present.

3.3 Extracted Features

Apart from the information present already in the dataset, the message column containing the email body can be used to derive many more features. The content of the message contains many psycho-linguistic and politeness-related markers which can help improve the accuracy of the models.

3.3.1 LIWC

Linguistic Inquiry and Word Count (LIWC)[22] is a dictionary containing words and word stems. Each word/word stem is associated with a number of psychological, emotional and linguistic categories.

This helps to break down a block of text into its semantic components, which can be used to analyze psycho-linguistic markers present in it.

TABLE 3.3: LIWC 2015 Example

Index	WC	Analytic	Clout	Authentic	Tone	WPS	Sixltr	Dic	function	pronoun	ppron	i	we	you	shehe	they	ipron	article	prep
0	1	93.26	99.00	1.00	25.77	1.0	100.00	100.00	0.00	0.00	0.00	0.00	0.00	0.0	0.0	0.0	0.00	0.00	0.00
1	48	77.41	58.24	35.37	25.77	24.0	16.67	56.25	31.25	6.25	6.25	6.25	0.00	0.0	0.0	0.0	0.00	2.08	10.42
2	11	57.84	50.00	2.40	25.77	11.0	9.09	45.45	36.36	9.09	0.00	0.00	0.00	0.0	0.0	0.0	9.09	9.09	0.00
3	7	93.26	50.00	1.00	99.00	3.5	0.00	28.57	0.00	0.00	0.00	0.00	0.00	0.0	0.0	0.0	0.00	0.00	0.00
4	11	4.01	99.00	1.00	25.77	11.0	9.09	54.55	45.45	27.27	9.09	0.00	9.09	0.0	0.0	0.0	18.18	0.00	0.00

The updated 2015 version of the LIWC dictionary has been used to generate 93 features for experiments carried out in this work. Table 3.3 shows some of the features generated using LIWC. Each row in the table corresponds to an email in the dataset.

3.3.2 Politeness

Similar to LIWC, Convokit[4] helps to extract conversational linguistic features from text. For this work in particular, politeness features[6] are extracted using Convokit. This helps generate 21 categorical features like Gratitude, Sentiment and Please which capture the politeness quotient in the mail.

3.3.3 SNA

Social Network Analysis (SNA) is used to study agents and ties in a social network. Since this dataset is not in a graph format, it needs to be converted to a compatible format before SNA can be applied to it. To achieve this, a social graph is constructed using employees as the nodes. From this graph, various features related to the node's degree and connections are extracted.

TABLE 3.4: SNA Features

Index	name	overalldegree	nodes	edges	density	bw	pagerank	closeness	authscore	hubscore	eigen
0	00c1bde836c715286ed888dfa808703967eb335f	86	1107	4718	0.007707	2463.515145	0.000217	0.000001	0.003719	0.002913	0.003934
1	02ab1ab9d8e1175006d696e533e129fc44d1abf0	114	1382	5815	0.006094	18105.003140	0.000639	0.000001	0.011138	0.003741	0.008938
2	03f155e1147d239917b905e50bf8d3f19d2a8148	133	1157	6375	0.009533	5607.304214	0.000431	0.000001	0.019390	0.015877	0.020778
3	043072a101e299a6de3993e8bc6fcf43573e6161	132	733	4962	0.018496	10217.306025	0.000320	0.000001	0.011817	0.014012	0.015356
4	0489a242d084104452045fece4038cd45fba6d7a	170	893	6671	0.016750	14330.551100	0.000326	0.000001	0.016419	0.019228	0.021476

3.3.4 Interpersonal Context

While the previous features help to capture information from the mail content or a social graph, in interpersonal context, the primary aim is to capture the linguistic style of the sender and receiver. The equations below depict the procedure followed to generate these features.

$$\begin{gathered} \text{For } e_t : s \rightarrow r, \text{ where} \\ e_t = \text{Email exchanged at time} = t, s = \text{Sender}, r = \text{Receiver} \end{gathered} \tag{3.1}$$

To capture the linguistic style of the sender, the recent most 4 emails sent by the sender to anyone else in the organization are concatenated with a [SEP] token (special token used by BERT tokenizers) in between. The BERT embedding of this concatenated text is used as a representation of the linguistic style of the sender at time = t.

$$\begin{aligned} L_t^s &= \phi_{BERT}\left(\left[e_{t-4}^s : [SEP] : e_{t-3}^s : [SEP] : ... : e_{t-1}^s\right]\right) \\ &= \text{Linguistic Style of } s \text{ at time} = t \\ \text{where } \phi_{BERT}(x) &\text{ returns a 768 dimension BERT embedding of } x \end{aligned} \tag{3.2}$$

Similarly we can capture the receiver's style by concatenating the recent most 4 mails sent by the receiver r to anyone else in the organisation.

$$\begin{aligned} L_t^r &= \phi_{BERT}\left(\left[e_{t-4}^r : [SEP] : e_{t-3}^r : [SEP] : ... : e_{t-1}^r\right]\right) \\ &= \text{Linguistic Style of } r \text{ at time} = t \end{aligned} \tag{3.3}$$

Now, to see how much similar the content of current email e_t is to the writing style of sender s and receiver r, we compute the cosine similarity between the BERT embedding of e_t with L_t^s and L_t^r.

$$\begin{aligned} S_{es} &= CosSim\left(\phi_{BERT}\left(e_t\right), L_t^s\right) \\ S_{er} &= CosSim\left(\phi_{BERT}\left(e_t\right), L_t^r\right) \\ &\text{where} \\ CosSim(a, b) &= \text{Cosine Similarity between vectors } a \text{ and } b \end{aligned} \tag{3.4}$$

S_{es} and S_{er} obtained above are the interpersonal context features we use in the proposed model.

3.3.5 Influence Features

In addition to SNA features which are being used to implicitly capture the workplace relations between employees by modelling the data into a social graph, the surveys conducted on employees where they were asked to rate their fellow coworkers on aspects like Friendship, Avoid, Communication Frequency and others are also present. These features are referred to as Influence features.

3.3.6 Topic

Topic is a categorical feature which indicates whether the email e_t is a Business related email or a Personal one. The classifier used to extract this feature has been released in the paper titled "Work Hard, Play Hard"[2].

3.3.7 Number of Features

Table 3.5 summarizes the dimension and data type of each of the features discussed in the last few sections.

TABLE 3.5: Feature Count

Feature	**Dimension**	**Type**
LIWC	93	Numerical + Categorical
Politeness	21	Categorical
SNA	10	Numerical
Influence	9	Numerical + Categorical
Interpersonal	2	Numerical
Total	**135**	

Chapter 4

Methodology

4.1 Problem Statement

$$\begin{aligned} &\text{Given } e_t : s \rightarrow r, \text{ where} \\ &\quad e_t = \text{Email exchanged at time} = t \\ &\quad s = \text{Sender} \\ &\quad r = \text{Receiver} \end{aligned} \tag{4.1}$$

The task is binary classification, where a 0 implies not getting a reply and 1 means getting a reply to the mail e_t, respectively.

While classification can be done using the mail text solely, the text alone cannot capture interpersonal information i.e. the work place relation between coworkers and SNA features. Thus, the features introduced in Chapter 3 provide crucial information which may help improve the accuracy if incorporated into the model.

4.2 Model Architecture

Using BERT transformer as the backbone, a MultiModal architecture is proposed which can incorporate both - the mail text and other non-textual features while learning to predict the probability of getting a reply to a particular mail.

BERT can handle the text part and generate contextual embeddings for the mail body. To handle the extra non-textual features, some architectural changes need to be made to the model rather than simply adding a few layers after BERT and finetuning it. This is implemented by concatenating the numerical and categorical features with the 768 dimension embedding[11], output by the BERT model. Figure 4.1 shows the model architecture.

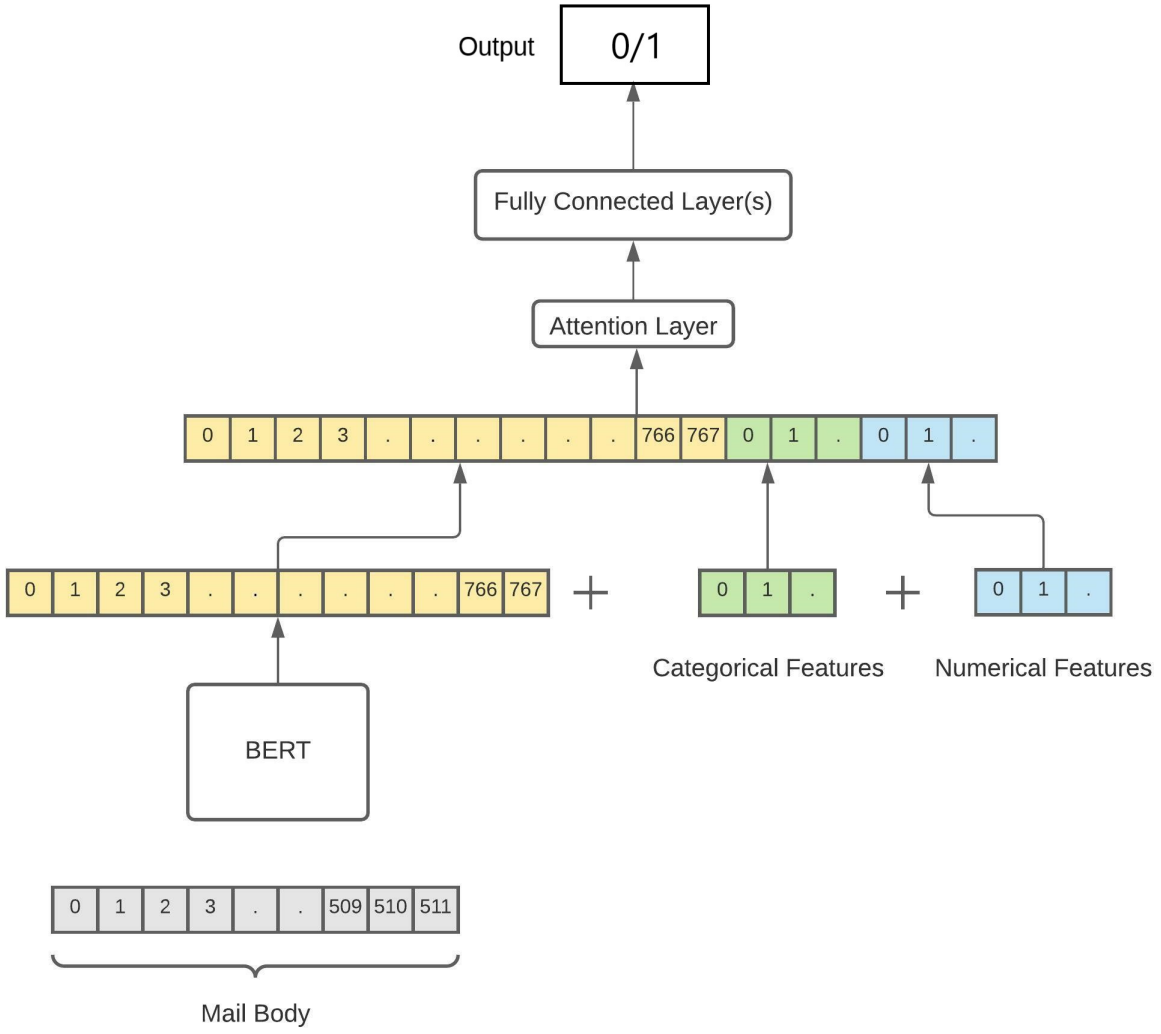


FIGURE 4.1: Model Architecture

The model also conatins an attention layer between the final classifier layers and the concatenated email representation. This helps the model to focus on the features which are important when calculating the final probability.

4.3 Training Parameters

4.3.1 Baseline Models

Experiments were conducted with various architectures, ranging from TF-IDF and CNN to BERT and RoBERTa. Results from these models help us understand the metrics of similar models for this data and task. Table 4.1 shows the complete list of baseline models used in the experiments.

TABLE 4.1: List of baseline models

Category	**Model**
Basic	SVM + TF-IDF
DL Based	CNN
	LSTM
	BiLSTM
	LSTM + Attention
Transformers	BERT
	RoBERTa

All the DL-based baseline models were trained with following configuration:

$$\begin{aligned}
&\text{Batch Size} = 1024 && dotted\\
&\text{Vocabulary Size} = 2000 && dotted\\
&\text{Input Length} = 200 && dotted\\
&\text{Word Embedding Dimension} = 50 && dotted\\
&\text{Number of Epochs} = 15 && dotted\\
&\text{Train : Test Split} = 0.8 : 0.2 && dotted
\end{aligned}$$

Apart from this, during each epoch the training data was split into training and validation set with a 0.8:0.2 split. This was done to monitor the validation loss. Early stopping was enabled which would stop the training if the validation loss did not improve for 3 consecutive epochs.

4.3.2 Transformer Models

For BERT and RoBERTa, following training arguments were used:

$$\begin{aligned}
&\text{Batch Size} = 150 && dotted\\
&\text{Input Length} = 200 && dotted\\
&\text{Number of Epochs} = 10 && dotted\\
&\text{Train : Test Split} = 0.8 : 0.2 && dotted\\
&\text{Initial Learning Rate} = 5e^{-5} && dotted\\
&\text{Warmup Ratio} = 0.07 && dotted\\
&\text{Weight Decay} = 0.5 && dotted
\end{aligned}$$

The loss function used to train all these models is Binary Cross Entropy.

$$L(y, p) = -\big(y \log(p) + (1 - y) \log(1 - p)\big) \text{ = Loss Function}$$
$$y = \text{Actual Class}$$
$$p = \text{Predicted Class}$$

For optimizer, ADAM optimizer[14] was used for all baseline DL models and ADAM with Weight decaay was used for Transformer baseline models.

4.4 Experiments

Experiments were conducted by varying the data size. Since the Influence features were only available for a small set of employees, the resulting dataset contains 18k rows as opposed to 600k rows for the whole dataset. Similarly, the features appended to the BERT embedding were also varied. The findings from all these variations have been listed in the next chapter.

Chapter 5

Result and Discussion

5.1 Results

The results from all the models that have been introduced previously are discussed in this section.

5.1.1 Full Data

All the baseline models were run on the full dataset (600k rows). They do not use anything other than the mail text as input.

TABLE 5.1: Baseline Results - Full Data

Model	Majority F1	Minority F1	Macro F1	Accuracy
SVM + tfidf	0.819	0.19	0.505	0.705
CNN	0.827	0.163	0.495	0.713
LSTM	0.827	0.178	0.503	0.714
BiLSTM	0.827	0.183	0.505	0.715
LSTM + Attention	0.828	0.179	0.503	0.715
BERT	0.827	0.215	0.521	0.716
RoBERTa	0.826	**0.232**	**0.529**	**0.717**

RoBERTa outperforms all the other baseline models on all metrics except Majority F1. Since the other features are not available for the full data, the MultiModal models have not been included in Table 5.1.

5.1.2 MultiModal Data

Using 18K rows from the full data, this subset is created which contains 135 other non-text features apart from the email body. Results from the models that were run on this data are presented in Table 5.2

TABLE 5.2: Model Results - 18K Data

Model	**Minority F1**	**Macro F1**	**Accuracy**
SVM + tfidf	0.263	0.528	0.677
CNN	0.240	0.524	0.693
LSTM	0.133	0.481	0.713
BiLSTM	0.186	0.507	0.716
LSTM + Attention	0.034	0.433	0.713
BERT	0.272	0.546	0.712
RoBERTa	0.272	0.548	0.716
BERT + LIWC + Politeness + Topic	0.085	0.461	0.724
BERT + Interpersonal Context	0.257	0.555	0.754
BERT + Interpersonal Context + Influence	**0.264**	**0.559**	**0.757**

It is evident from the results that these non-text features do have the power to improve the model by providing more context about the mail, the sender and the receiver. While the model with LIWC, Politeness and Topic features are only $\sim$1% more accurate than RoBERTa, the other MultiModal models are upto $\sim$4% more accurate.

5.2 Discussion

As seen in Table 5.1 and 5.2, the MultiModal models perform better than all the baseline models, beating the most capable model (RoBERTa) by more than $\sim$4% in terms of accuracy. Comparing the different MultiModal models, we see that adding Influence features to BERT with Interpersonal Context features results in a small improvement.

Another observation from the above results is that while features like LIWC and Politeness, which capture different stylistic properties about the text, do increase the accuracy of the model but not nearly as much as features like Interpersonal Context, which provide information about the linguistic style of the sender and receiver.

Further work needs to be done to figure out the kind of features which lead to this increase in the model performance.

Chapter 6

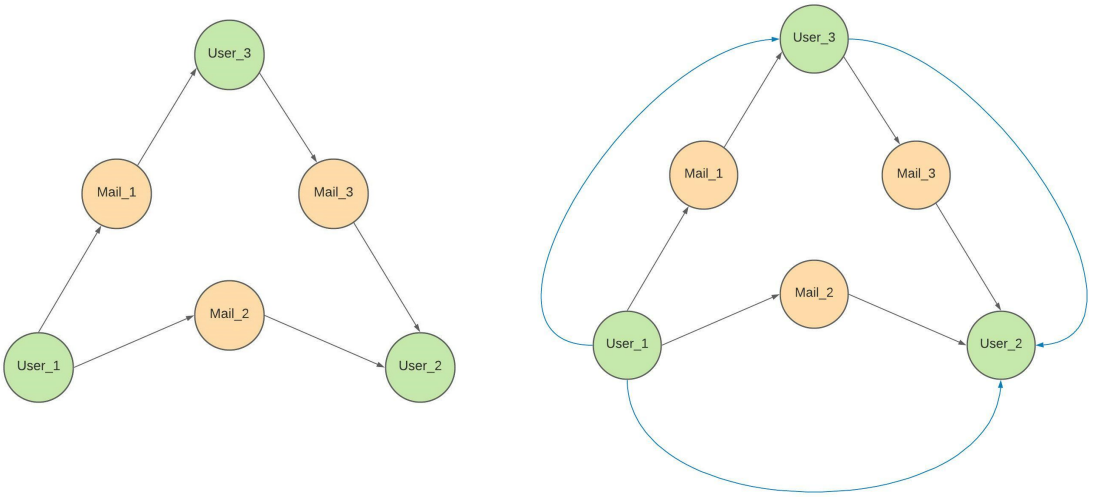
Conclusion

6.1 Recommendation for Future Research

In the past few years, Graph Neural Networks (GNNs) have gained a lot of popularity. Traditional Deep Learning based models like CNNs and RNNs work on data like Images and Text, which can be regarded as Euclidean data. GNNs extend the power of these models to graph data which is non-Euclidean in nature. Molecular structure of proteins and people in a social graph can be considered as examples of non-Euclidean data. GNNs have also achieved great success in tasks on similar datasets in the recent years[30, 10]. This has made GNNs an active research topic in Artificial Intelligence today leading to new and improved graph based models like Graph Convolutional Network (GCN)[15] and Graph Attention Network[29].

The Luxury-Standard dataset contains information about both - the emails sent by the employees and the mail content. This means that the current data can be modelled as a social network graph after which a GNN can be applied to it. The advantage of GNNs is that they are able to leverage information from the neighbour nodes in the social graph similar to how a CNN exploits the spatial data in an image by looking at the adjacent pixels when sliding a window over the input.

The data can be represented as a graph by converting each employee and mail as a node in a heterogeneous graph. For each $e_t : s \rightarrow r$, an edge is constructed between s-e_t and r-e_t. A design choice over here can give rise to two different kind of graphs - one containing an edge between s-r and another one without any s-r edge.



(A) Without Sender-Receiver Edge (B) With Sender-Receiver Edge

FIGURE 6.1: Data as Heterogeneous Graph

Figure 6.1 shows an example of how the graph would look if created using either of the above mentioned methods. Once the data is in a compatible format, graph based machine learning libraries like PyTorch-Geometric[9] and StellarGraph[7] can be used to apply State-of-the-Art graph based methods on it.

6.2 Conclusion

Predicting the probability of getting a reply to a corporate mail would help the employees plan better. It can also help them write emails which are more likely to get a reply. The email body contains a lot of information that could help predict the chances of getting a reply back. But looking beyond this, we can see that there are many more factors involved like preferences of the sender and receiver, their workplace relations with each other, style of the message, and so on.

Incorporating these various features along with the email body cannot be done in a simple Transformer model like BERT which works solely on textual input. The proposed MultiModal architecture addresses this issue by using BERT to generate embeddings for the email body which is then concatenated with other numerical and categorical features to form the final email representation. These changes improve the accuracy by more than $\sim$4% as compared to purely text based methods like BERT and RoBERTa.

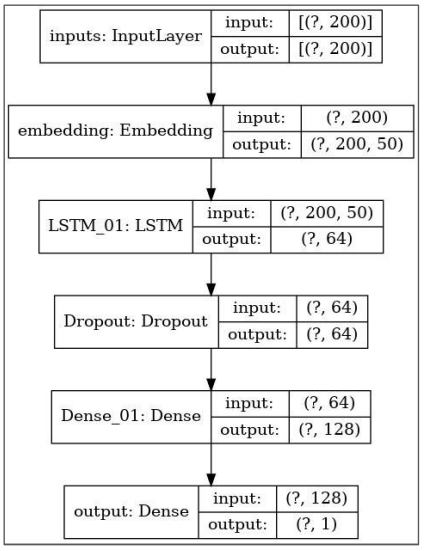
Appendix A

Code

LSTM

```
11 model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['acc',f1_m,precision_m, recall_m])
12
13 model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          [(None, 200)]             0
_________________________________________________________________
embedding (Embedding)        (None, 200, 50)           100000
_________________________________________________________________
LSTM_01 (LSTM)               (None, 64)                29440
_________________________________________________________________
Dropout (Dropout)            (None, 64)                0
_________________________________________________________________
Dense_01 (Dense)             (None, 128)               8320
_________________________________________________________________
output (Dense)               (None, 1)                 129
=================================================================
Total params: 137,889
Trainable params: 137,889
Non-trainable params: 0
_________________________________________________________________
```

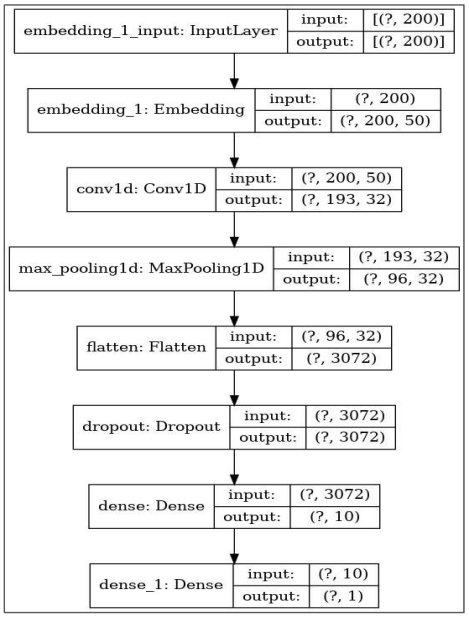


CNN

```
model_CNN.compile(loss='binary_crossentropy',
              optimizer= 'adam',
              metrics=['acc',f1_m,precision_m, recall_m])

model_CNN.summary()
```

```
Model: "CNN_with_embeddings"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 200, 50)           100000
_________________________________________________________________
conv1d (Conv1D)              (None, 193, 32)           12832
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 96, 32)            0
_________________________________________________________________
flatten (Flatten)            (None, 3072)              0
_________________________________________________________________
dropout (Dropout)            (None, 3072)              0
_________________________________________________________________
dense (Dense)                (None, 10)                30730
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 143,573
Trainable params: 143,573
```

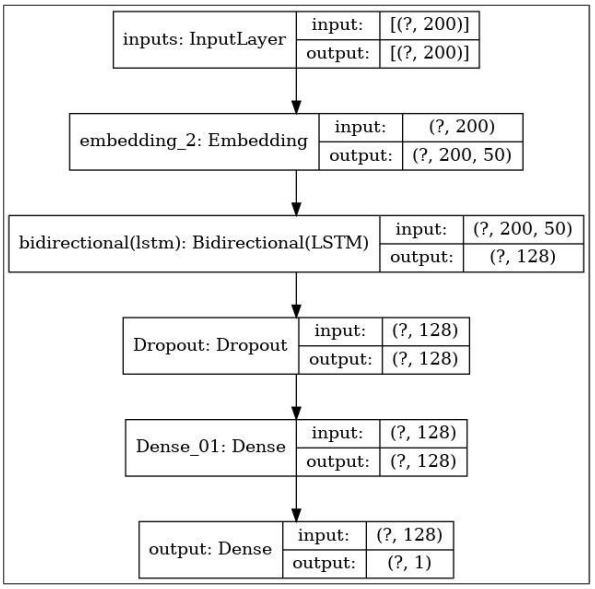


BiLSTM

```
bi_model.compile(loss='binary_crossentropy',
                 optimizer='adam',
                 metrics=['acc',f1_m,precision_m, recall_m])

bi_model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          [(None, 200)]             0
_________________________________________________________________
embedding_2 (Embedding)      (None, 200, 50)           100000
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               58880
_________________________________________________________________
Dropout (Dropout)            (None, 128)               0
_________________________________________________________________
Dense_01 (Dense)             (None, 128)               16512
_________________________________________________________________
output (Dense)               (None, 1)                 129
=================================================================
Total params: 175,521
Trainable params: 175,521
Non-trainable params: 0
```



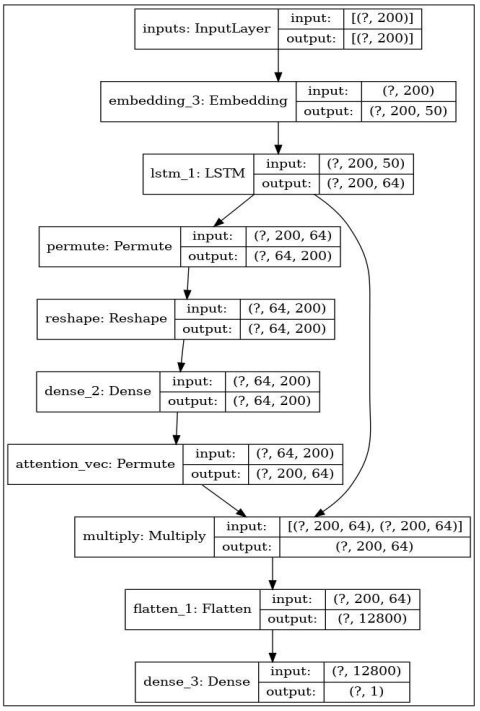
LSTM + Attention

```
m.summary()
m.compile(loss='binary_crossentropy',optimizer='adam',metrics=['acc',f1_m,precision_m, recall_m])
```

Model: "model_2"

Layer (type)	Output Shape	Param #	Connected to
inputs (InputLayer)	[(None, 200)]	0	
embedding_3 (Embedding)	(None, 200, 50)	100000	inputs[0][0]
lstm_1 (LSTM)	(None, 200, 64)	29440	embedding_3[0][0]
permute (Permute)	(None, 64, 200)	0	lstm_1[0][0]
reshape (Reshape)	(None, 64, 200)	0	permute[0][0]
dense_2 (Dense)	(None, 64, 200)	40200	reshape[0][0]
attention_vec (Permute)	(None, 200, 64)	0	dense_2[0][0]
multiply (Multiply)	(None, 200, 64)	0	lstm_1[0][0] attention_vec[0][0]
flatten_1 (Flatten)	(None, 12800)	0	multiply[0][0]
dense_3 (Dense)	(None, 1)	12801	flatten_1[0][0]

Total params: 182,441
Trainable params: 182,441



Baseline Transformer Models

BERT model

```
model_args = ClassificationArgs()
model_args.num_train_epochs = 10
model_args.train_batch_size = 160
model_args.eval_batch_size = 160
model_args.dataloader_num_workers = 4
model_args.overwrite_output_dir = True
model_args.max_seq_length = 200
model_args.output_dir = 'baseline_bert_model/'
model_args.learning_rate = 5e-05
model_args.manual_seed = 42
model_args.warmup_ratio = 0.07
model_args.weight_decay = 0.05

model = ClassificationModel("bert","bert-base-cased",num_labels=2,args=model_args,cuda_device=12)

# Train the model
model.train_model(train_df,acc = sklearn.metrics.accuracy_score)
```

BERT - Code Snippet

Roberta model

```
model_args = ClassificationArgs()
model_args.num_train_epochs = 10
model_args.train_batch_size = 150
model_args.eval_batch_size = 150
model_args.dataloader_num_workers = 4
model_args.overwrite_output_dir = True
model_args.max_seq_length = 200
model_args.output_dir = 'baseline_roberta_model/'
model_args.learning_rate = 5e-05
model_args.manual_seed = 42
model_args.warmup_ratio = 0.07
model_args.weight_decay = 0.05

model = ClassificationModel("roberta","roberta-base",num_labels=2,args=model_args,cuda_device=13)

# Train the model
model.train_model(train_df,acc = sklearn.metrics.accuracy_score)
```

RoBERTa - Code Snippet

Appendix B

Converting dataset from CSV to Graph

Currently, the dataset is in a CSV format which needs to be converted into a Graph format for which we can then feed into the GNN model. To keep things simple for the baseline model, we model all the employees and EACH mail as a node in a heterogeneous graph containing two different node types. After this we create an edge between the following pairs:

- (sender, message)
- (receiver, message)
- (sender,receiver)

Mail node contains an attribute called reply which contains 0 or 1 based on whether this particular message received a reply or not. The final graph created is an undirected heterogeneous multi-graph since the nodes are connected to multiple nodes.

```
samp_df.head(2)
```

	message	sender	to_field	reply
255140	NA\n\nsee the email message from the customer....	c87d6b35c48bee8ce2dbb8af7df7d96b5378d2b5	836b7f26aac0bd032565cc3bccff62d9ac18e7e1	0
195353	NA\n\nthanks\n\n	b79034db2bc923128d46fadfc6144ce16a8cf0d1	0fd2e11f386174c476f13e6a6469cd89c5877187	1

```
df_for_nx.head(6)
```

	node1	node2
0	NA\n\nsee the email message from the customer....	c87d6b35c48bee8ce2dbb8af7df7d96b5378d2b5
1	c87d6b35c48bee8ce2dbb8af7df7d96b5378d2b5	836b7f26aac0bd032565cc3bccff62d9ac18e7e1
2	NA\n\nsee the email message from the customer....	836b7f26aac0bd032565cc3bccff62d9ac18e7e1
3	NA\n\nthanks\n\n	b79034db2bc923128d46fadfc6144ce16a8cf0d1
4	b79034db2bc923128d46fadfc6144ce16a8cf0d1	0fd2e11f386174c476f13e6a6469cd89c5877187
5	NA\n\nthanks\n\n	0fd2e11f386174c476f13e6a6469cd89c5877187

To create the graph from CSV, the node pairs mentioned earlier need to be created. The code above shows the original dataframe and the new dataframe used to create the graph. They both contain the same data, just in different format

```
samp_df = new_df.sample(10,random_state=42)
node1 = []
node2 = []

users = []
mails = []
mail_reply = []
for index,row in samp_df.iterrows():
    node1.append(row[['message']].values[0])
    node2.append(row[['sender']].values[0])



    node1.append(row[['sender']].values[0])
    node2.append(row[['to_field']].values[0])

    node1.append(row[['message']].values[0])
    node2.append(row[['to_field']].values[0])

    #_____
    users.append(row[['sender']].values[0])
    users.append(row[['to_field']].values[0])
    mails.append(row[['message']].values[0])
    mail_reply.append(row[['reply']].values[0])
```

```
df_for_nx = pd.DataFrame(columns = ['node1','node2'])
```

```
df_for_nx['node1'] , df_for_nx['node2'] = node1 , node2
```

```
df_for_nx
```

	node1	node2
0	NA\n\nsee the email message from the customer....	c87d6b35c48bee8ce2dbb8af7df7d96b5378d2b5
1	c87d6b35c48bee8ce2dbb8af7df7d96b5378d2b5	836b7f26aac0bd032565cc3bccff62d9ac18e7e1
2	NA\n\nsee the email message from the customer....	836b7f26aac0bd032565cc3bccff62d9ac18e7e1
3	NA\n\nthanks\n\n	b79034db2bc923128d46fadfc6144ce16a8cf0d1
4	b79034db2bc923128d46fadfc6144ce16a8cf0d1	0fd2e11f386174c476f13e6a6469cd89c5877187
5	NA\n\nthanks\n\n	0fd2e11f386174c476f13e6a6469cd89c5877187
6	NA\n\nhi harriet,\n\ni think the text is too l...	89d58d04354d0308c7038d177dfbe82ef54972af
7	89d58d04354d0308c7038d177dfbe82ef54972af	7ffad702b05c761466398cab7a0b238316410069
8	NA\n\nhi harriet,\n\ni think the text is too l...	7ffad702b05c761466398cab7a0b238316410069
9	NA\n\nare you in the ORGANIZATION_144600055315...	af4636d558f3e4624aebcc939b4278c7c4352ce6
10	af4636d558f3e4624aebcc939b4278c7c4352ce6	66ea625781efeb100accfd05e72aed7d2da68c63
11	NA\n\nare you in the ORGANIZATION_144600055315...	66ea625781efeb100accfd05e72aed7d2da68c63
12	NA\n\ndon't have any details but the last i he...	eeda2898bce68397c0e670460d90c3c2e9cc72ef
13	eeda2898bce68397c0e670460d90c3c2e9cc72ef	174c682cda0a1f51dce981ed58a8f6dd525ebd32
14	NA\n\ndon't have any details but the last i he...	174c682cda0a1f51dce981ed58a8f6dd525ebd32
15	NA\n\nPERSON 1453669423176 5\n\nbest,\n\n	182ca7b7cedbc10b451c2b0d098455eda469d14f

```
G=nx.from_pandas_edgelist(df_for_nx, 'node1', 'node2')
```

Using this code, a networkx graph is created which can be used with a Graph Machine Learning Library.

```
graph_data = StellarGraph.from_networkx(G,node_type_attr='type',edge_type_attr='edge_weight')
print(graph_data.info())
```

```
StellarGraph: Undirected multigraph
 Nodes: 669440, Edges: 1410478

 Node types:
  mail: [665120]
    Features: none
    Edge types: mail-default->user
  user: [4320]
    Features: none
    Edge types: user-default->mail, user-default->user

 Edge types:
    mail-default->user: [1330240]
        Weights: all 1 (default)
        Features: none
    user-default->user: [80238]
        Weights: all 1 (default)
        Features: none
```

Bibliography

[1] Douglas Aberdeen, Ondrey Pacovsky, and Andrew Slater. "The Learning Behind Gmail Priority Inbox". In: *LCCC : NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*. 2010.

[2] Sakhar Alkhereyf and Owen Rambow. "Work Hard, Play Hard: Email Classification on the Avocado and Enron Corpora". In: *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 57–65. DOI: 10.18653/v1/W17-2408. URL: https://aclanthology.org/W17-2408.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2015).

[4] Jonathan P. Chang et al. "ConvoKit: A Toolkit for the Analysis of Conversations". In: *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 1st virtual meeting: Association for Computational Linguistics, July 2020, pp. 57–60. URL: https://aclanthology.org/2020.sigdial-1.8.

[5] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

[6] Cristian Danescu-Niculescu-Mizil et al. "A computational approach to politeness with application to social factors". In: *ACL*. 2013.

[7] CSIRO's Data61. *StellarGraph Machine Learning Library*. https://github.com/stellargraph/stellargraph. 2018.

[8] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[9] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[10] Alex M Fout. "Protein interface prediction using graph convolutional networks". PhD thesis. Colorado State University, 2017.

[11] Ken Gu and Akshay Budhkar. "A Package for Learning on Tabular and Text Data with Transformers". In: *Proceedings of the Third Workshop on Multimodal Artificial Intelligence*. Mexico City, Mexico: Association for Computational Linguistics, June 2021, pp. 69–73. DOI: 10.18653/v1/2021.maiworkshop-1.10. URL: https://www.aclweb.org/anthology/2021.maiworkshop-1.10.

[12] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[13] K. Jones. "A Statistical Interpretation of Term Specificity in Retrieval". In: *Journal of Documentation* 60 (Jan. 2004), pp. 493–502. DOI: 10.1108/00220410410560573.

[14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[15] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[16] Bryan Klimt and Yiming Yang. "The Enron Corpus: A New Dataset for Email Classification Research". In: *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings*. Ed. by Jean-François Boulicaut et al. Vol. 3201. Lecture Notes in Computer Science. Springer, 2004, pp. 217–226. DOI: 10.1007/978-3-540-30115-8_22. URL: https://doi.org/10.1007/978-3-540-30115-8_22.

[17] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[18] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[19] Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166.

[20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. “Scoring, term weighting, and the vector space model”. In: *Introduction to Information Retrieval*. Cambridge University Press, 2008, 100–123. DOI: 10.1017/CBO9780511809071.007.

[21] Sudipto Mukherjee and Ke Jiang. *A Content-Based Approach to Email Triage Action Prediction: Exploration and Evaluation*. 2019. arXiv: 1905.01991 [cs.IR].

[22] James W Pennebaker, Roger J Booth, and Martha E Francis. “Linguistic inquiry and word count: LIWC”. In: (2007).

[23] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65 6 (1958), pp. 386–408.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[25] Maya Sappelli et al. “Assessing e-mail intent and tasks in e-mail messages”. In: *Information Sciences* 358 (Mar. 2016). DOI: 10.1016/j.ins.2016.03.002.

[26] Bahareh Sarrafzadeh et al. “Characterizing and Predicting Email Deferral Behavior”. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM ’19. Melbourne VIC, Australia: Association for Computing Machinery, 2019, 627–635. ISBN: 9781450359405. DOI: 10.1145/3289600.3291028. URL: https://doi.org/10.1145/3289600.3291028.

[27] Wookje Sung et al. “Employees’ Responses to an Organizational Merger: Intraindividual Change in Organizational Identification, Attachment, and Turnover”. In: *Journal of Applied Psychology* 102 (2017), 910–934.

[28] Ashish Vaswani et al. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, 6000–6010. ISBN: 9781510860964.

[29] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations* (2018). URL: https://openreview.net/forum?id=rJXMpikCZ.

[30] Yongji Wu et al. “Graph Convolutional Networks with Markov Random Field Reasoning for Social Spammer Detection”. In: *AAAI*. 2020.

[31] Liu Yang et al. “Characterizing and Predicting Enterprise Email Reply Behavior”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’17. Shinjuku, Tokyo, Japan: Association for Computing Machinery, 2017, 235–244. ISBN: 9781450350228. DOI: 10.1145/3077136.3080782. URL: https://doi.org/10.1145/3077136.3080782.