# Text Classification: Exploiting the Social Network

Sakhar Alkhereyf

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2021

## ABSTRACT

Text Classification: Exploiting the Social Network

Sakhar Alkhereyf

Within the context of social networks, existing methods for document classification tasks typically only capture textual semantics while ignoring the text's metadata, e.g., the users who exchange emails and the communication networks they form. However, some work has shown that incorporating the social network information in addition to information from language is useful for various NLP applications, including sentiment analysis, inferring user attributes, and predicting interpersonal relations.

In this thesis, we present empirical studies of incorporating social network information from the underlying communication graphs for various text classification tasks. We show different graph representations for different problems. Also, we introduce social network features extracted from these graphs. We use and extend graph embedding models for text classification.

Our contributions are as follows. First, we have annotated large datasets of emails with fine-grained business and personal labels. Second, we propose graph representations for the social networks induced from documents and users and apply them on different text classification tasks. Third, we propose social network features extracted from these structures for documents and users. Fourth, we exploit different methods for modeling the social network of communication for four tasks: email classification into business and personal, overt display of power detection in emails, hierarchical power detection in emails, and Reddit post classification.

Our main findings are: incorporating the social network information using our proposed methods improves the classification performance for all of the four tasks, and we beat the state-of-the-art graph embedding based model on the three tasks on email; additionally, for the fourth task (Reddit post classification), we argue that simple methods with the proper representation for the task can outperform a state-of-the-art generic model.

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

I would like to begin by expressing my deepest gratitude to my research advisor Prof. Owen Rambow, for his invaluable guidance and endless support during my doctoral study at Columbia. I could not have asked for a better advisor than Owen. He has always been available to provide valuable feedback on my work. Even after he left Columbia, I did not feel any difference in his availability to mentor me closely as our weekly meetings did not change. He was not only my Ph.D. advisor but a friend from whom I learned a lot about different subjects. Owen taught me how to define research problems, how to obtain suitable empirical results, and how to write and present my work clearly and precisely. Thanks, Owen, for your constant guidance and for your dedicated time even after leaving Columbia.

I am grateful to my doctoral committee members, Prof. Julia Hirschberg, Prof. Kathleen McKeown, Dr. Smaranda Muresan, and Prof. Dragomir Radev, for finding the time to join my dissertation committee. I appreciate their valuable feedback and comments, as they significantly improved my dissertation. I would specifically like to thank Julia and Smara for their valuable comments on this dissertation during the initial proposal stage. I had the chance to take the *NLP in Context: Computational Models of Social Meaning* seminar class with Smara. Thanks, Smara, for being a great instructor and preparing a course that provided me deep insights into the field. I am also incredibly thankful to Kathy for her guidance and support as my departmental advisor. I would also like to thank Drago for being a great instructor. Not only did I have the chance to take *Natural Language Processing* with him at Columbia in 2015, but he also gave me the opportunity to be a TA twice for the NLP course.

I extend my appreciation to my sponsor, King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia, and to the Saudi Arabian government for providing me with generous scholarships that allowed me to pursue my graduate studies. I am deeply thankful for their generous scholarship, which also included a full scholarship for my wife to pursue her graduate studies. I would also like to thank my former supervisor at KACST, Dr. Abdulqadir Alaqeeli, for his support and guidance during my work at KACST before transferring to Columbia.

Before joining Columbia, I had a chance to work on a research project on Arabic NLP with Prof. Nizar Habash. Thanks, Nizar, for giving me the opportunity to collaborate with you on that project and for encouraging me to apply to the Ph.D. program at Columbia. As a doctoral student, I also had the chance to work with Nizar on another project while he was at NYU Abu Dhabi.

My years at Columbia and CCLS would not have been possible without these amazing people who played the roles of mentor and friend. I would like to thank my friends and colleagues at Columbia and CCLS (in no particular order): Ramy Eskander, Faisal AlShargi, Mohammed Rasooli, Ahmed ElKholy, Axinia Radeva, Kathy Hickey, Idrija Ibrahimagic, Vinodkumar Prabhakaran, Apoorv Agarwal, Jessica Rosa, Hooshmand Razaghi, Naser AlDuaij, Tariq Alhendi, Edward Liu, Robert Dadashi, Mohamed Elbadrashiny, Hatim Diab, Saikat Chakraborty, Chris Hidey, Chris Kedzie, Giannis Karamanolakis, Oscar Chang, Olivia Winn, Daniel Bauer, Anant Sharma, Avery Wu, Tianxiao Ye, and Elias Tesfaye. I would like to give special thanks to my dear friend, Antonio Moretti, for his support and collaboration during my Ph.D. Although we have been in different labs at Columbia, we have given each other feedback on our work.

I would also like to thank my dear friends: Ibrahim Almosallam, Hani Altwaijri, Azzam Alsuhaibani, Mazen Alhwaimel, and Khalid Alfoaim. Ibrahim has been a mentor to me since my days at KACST. Hani was always ready to share his knowledge of machine learning with me, and he has been a supportive friend in New York City. Azzam, Khalid, and Mazen have also always been there to support me during my Ph.D.

Finally, to my parents, siblings (Shihab, Wehad, and Laith), and my wife, Aljohara, my words cannot express my profound gratitude for you. I am forever indebted to them. My mother, Sara, has always been my guardian angel, watching over me every step of my life. My father, Badr, has always been my hero, being there for me and supporting everything I have done. My siblings have always been my best friends, standing by me and caring and sharing the sorrows and joys of life. To my wife and best friend, Aljohara, thank you for being supportive and patient throughout my journey. Thanks for your encouragement during the hard times, especially when writing this dissertation during the COVID-19 pandemic. You have always been there to help, and you have made everything possible for me to complete my studies. Without you, I would never have been able to complete this dissertation.

To my beloved parents, siblings, and wife.

# Chapter 1

# Introduction

## 1.1 Motivation

A large body of work uses social networks to predict user characteristics. This work exploits *homophily*, i.e., the tendency for similar individuals to engage with one another. For example, young people are more likely to communicate with other young people. In contrast, there has been far less work that uses the communication network (the network induced by conversations) to improve document classification on the communications themselves. This is a more challenging problem since homophily does not determine when characterizing the communications themselves. In some document classification tasks, the document category might not be directly inferred from the relationship of the participants when the same participants exchange different types of documents. For instance, the same people might exchange both personal and business emails, or urgent and nonessential emails.

In this thesis, we investigate using textual content of documents and the underlying social network of document exchange in the context of written conversation. As a case study, we manually annotated two e-mail datasets: Enron and Avocado, for classifying email into two categories: "business" and "personal". There are several reasons for this choice:

1. We are interested in how personal relationships affect communication, taking into account that the same pair of people may have multiple types of relationships.

2. Email remains a crucial communication medium for both individuals and organizations for both personal and business communications. Kiritchenko and Matwin (2011) show that a typical user daily receives 40-50 emails.

3. Two large datasets are available, the Enron corpus and a dataset of emails from an anonymous defunct information technology company referred to as Avocado.

4. Despite the massive growth of other social media over the past decade, enterprise email is still used, not only for business communication but also for personal purposes, as the recent Avocado corpus shows that it has a reasonable proportion of personal emails.

5. Furthermore, unlike spam filtering, email classification into business and personal is a challenging task (as shown in the human inter-annotator agreement reported in (Jabbari *et al.*, 2006) as well as in our annotation in section 4.4) and remains an unsolved task.

6. We are interested in how people communicate in conversations. In fact, email communications are real conversations as there are senders and intended recipients, and this what distinguishes email from other genres such as blogs and Twitter, which are readily available, but typically used for broadcasting to a large group of followers rather than engaging in real conversations.

As for any document classification task, the language used (reflecting both content and language style) is highly predictive of the class. For instance, when a student speaks with her friends, she often uses relatively less formal language than when she speaks with her professor, and she will talk about different topics. As we will see, word embeddings provides a strong baseline for this task.

In this thesis, we develop methods to use the social network features in addition to lexical features, and we show that the social network features improve classification performance when compared to using lexical features alone. Our main task is to use the textual content of the email and the underlying social network of email exchange for email classification into two categories, "Business" and "Personal". For this task, we use two annotated e-mail datasets: Enron and Avocado. We model the task of finding the rarer class (personal emails) in a set of all emails. We are interested in developing models that can be applied to unseen datasets, so that we can detect personal emails in new datasets with no retraining.

In addition to email classification into "business" and "personal", we apply our methods to other tasks: overt display of power detection in emails, hierarchical power detection in emails; and Reddit post classification. We show that these text classification tasks can be improved by incorporating the social network information from the underlying communication graphs.

## 1.2 Thesis Outline

In this section, we give the outline of the thesis. We divide the thesis into three parts. The first part – Data Creation, Data Analysis, and Methods – lays the foundation for the rest of the thesis, describing in detail the datasets we use in our study and the methods we use for analysis and our systems. The second part – Business and Personal Email Classification – presents the core work of this thesis. We apply models for incorporating social network information on the task of email classification into business and personal categories. The third part – Other Applications – shows three other applications of this dissertation in which we exploit the social network in addition to lexical features: overt display of power detection in email, hierarchical power detection in email, and Reddit post classification.

Below we briefly discuss what each chapter contains.

- Chapter 2 discusses the related work in the areas of text classification, incorporating social network information for different text classification tasks, and graph embeddings.

**Part I: Data Creation, Data Analysis, and Methods** In this part, we present the data we use throughout this dissertation and provide some analysis of the data. We also present methods for modeling the tasks.

- Chapter 3 presents existing datasets we use or extend in this thesis. In this chapter, we give a brief history of the Enron company and corpus. In addition, we present different existing datasets based on the Enron original collection. We also discuss in this chapter the Avocado email collection. Also, we present the Reddit post dataset that we use in chapter 17.

- In chapter 4, we present in detail the new annotations for the task of email classification into business and personal categories. We show the annotation scheme we use for this task. We present the datasets which we will be using in Part II.

- Chapter 5 presents different graph structures to represent the communication network. We show in this chapter three types of graphs: bipartite graphs for documents and users, user graphs, and document graphs. These graph structures are generic, and we apply them to different tasks discussed in this thesis.

- In chapter 6, we show social network analysis on the datasets presented in chapter 4. The results show that the networks induced from different types of emails have different properties. This motivates us to incorporate information from the underlying social network for the task of email classification into business and personal presented in Part II.

- In chapter 7, we present social network features we extract from different graph structures representing the communication social networks discussed in chapter 5. We present a variety of features, and we use these features throughout the thesis in different classification tasks.

- Chapter 8 presents methods that we will use throughout this thesis. We show in this chapter: the software framework we have used to conduct research in this thesis; machine learning classifiers and metrics; lexical features we extract from the documents' content.

**Part II: Business and Personal Email Classification** In this part, we discuss the core task of this dissertation, email classification into business and personal.

- Chapter 9 is the introduction to the second part. We introduce in this chapter the task of email classification into business and personal. We show the baselines and machine learning classifiers we use for this task in this chapter.
- In chapter 10, we present methods of lexical modeling for emails. We show results for different lexical models.
- In chapter 11, we present experiments for social network modeling methods. We use different machine learning classifiers, and we compare the performance of classifiers that have access to both social network information from the communication graphs and lexical content of emails with classifiers from the previous chapter. The results here show that incorporating social network information improves the classification performance.
- Chapter 12 shows thread modeling techniques that incorporate the thread structure. We present two methods for thread modeling in this chapter: a simple majority vote for emails in the same thread; and a sequential modeling classifier.
- Chapter 13 presents alternative methods for modeling the social networks. Particularly, we present experiments using a state-of-the-art graph embedding model, GraphSAGE.

- We conclude the second part in chapter 14. We show in this chapter additional evaluations of our methods. Particularly, we evaluate our best models on the test sets and the Sheffield dataset (introduced in subsection 4.2.2).

**Part III: Other Applications** In the third part, we show other applications of the methods presented so far in this thesis. We show two other applications on email: overt display of power (ODP) detection; and hierarchical power prediction. Also, we show application on Reddit post classification.

- In chapter 15, we discuss applying our methods on another task: overt display of power. We extend previous studies that focus on detecting overt display of power at an utterance level to an email level. We show experiments on incorporating social network information with other features presented in the previous studies.
- Chapter 16 presents our third application on emails, detecting hierarchical power in emails.
- Chapter 17 shows the last application in this thesis. We apply our methods on Reddit post classification. We also show another technique for label propagation in the user post graph.
- We conclude the thesis in chapter 18.

## 1.3 Summary of Contributions

The focus of this dissertation is the development of new techniques for incorporating information from the underlying social network of communication for text classification. We combine social network information from graphs representing the underlying communication network with lexical information from the document content. We apply our work to two genres: email and Reddit. In email, our main task is email classification into business and personal (Part II). In addition, we apply our methods on two other tasks: overt display of power (ODP) detection and predicting hierarchical power between pairs in emails. For Reddit, we apply our methods on Reddit post classification. This thesis adds the following research to previous studies:

- We have collected large datasets of emails and annotated them with fine-grained business and personal labels. These datasets are based on two widely available email corpora: Enron and Avocado. We present these datasets with the details of annotations in chapter 4.
- We propose different graph structures to represent the communication network for documents and users. We present these graph structures in chapter 5. We use these graphs for different text classification tasks.
- We conduct social network analysis on graphs induced from the datasets annotated with business and personal labels. We analyze the induced personal and business sub-networks using different SNA measures, and we show in chapter 6 that the two networks have different properties using these measures.
- We propose various social network features extracted from different graph structures representing the underlying social network of communication for both users and documents. This way, we can use social network information for document classification without explicitly modeling different graphs separately using graph models. We discuss these social network features in chapter 7
- We apply our methods on different applications. We show that adding social network information to machine learning models improves the classification performance over models that have access only to the textual content of documents. We compare our proposed hand-engineered social network features with a state-of-the-art graph embedding model, Graph-

SAGE, and our model outperforms it on three tasks. Particularly, our proposed features outperform GraphSAGE on the email tasks: classification into business and personal (Part II), overt display of power detection (chapter 15), and hierarchical power detection (chapter 16). The fourth task (chapter 17) turns out to be a different kind of problem than the other tasks as it does not involve "dyadic" relations. However, for this task, we propose a different, quite simple method, and it outperforms GraphSAGE.

- We also propose an extension of GraphSAGE to heterogeneous bipartite graphs that outperforms the ordinary GraphSAGE for the task of email classification into business and personal (section 13.2).

## 1.4 Ethical Considerations

Analyzing texts from social data is inherently fraught with ethical questions—especially relating to privacy. In this dissertation, we have made considerable effort to engage positively with ethical issues. Particularly, for our primary dataset in the task of email classification into business and personal, we have obtained an IRB waiver for the use of this data and to annotate it using a crowdsourcing platform. For Enron, we have used a publicly available collection of emails. For Avocado, we follow the license instructions that prohibit reproducing the email collection either partially or entirely.

# Chapter 2

# Literature Review

In this chapter, we provide a literature review to situate this thesis among the large body of work on exploiting the social network for text classification. Specifically, we review related work in the following areas: text classification, incorporating social network for NLP tasks, and Graph Embedding. We begin by discussing the topic of text classification in the area of natural language processing and review methods used for text classification. We give an overview of the traditional techniques; then, we discuss the recent methods for text classification with a focus on email classification. Then, we discuss the related work in incorporating social network information for different NLP tasks. Finally, we present the growing area of work in graph embeddings.

## 2.1 Text Classification

Unstructured data in the form of text is ubiquitous: emails, chats, web pages, and online blogs. In recent years, there has been an exponential growth in the number of digital documents and complex texts that require a deeper understanding of machine learning methods to classify texts in many applications accurately. The problem of classification has been widely studied in the data mining, machine learning, database, and information retrieval fields with applications in a variety of domains, such as document organization and medical diagnosis, and news group filtering. Text classification is the task of assigning predefined categories to text documents (Sebastiani, 2002). It is one of the fundamental tasks in the field of Natural Language Processing (NLP) with broad applications such as sentiment analysis (Tan *et al.*, 2011; Wang *et al.*, 2018c), topic labeling (Joachims, 1998; Hingmire *et al.*, 2013; Dieng *et al.*, 2016), spam detection (Kolcz, 2005; Renuka and Visalakshi, 2014), and intent categorization (Sappelli *et al.*, 2016; Lampert *et al.*, 2008). In general, there are four different levels for the scope of text classification systems: document level, paragraph level, sentence level, and sub-sentence level (Kowsari *et al.*, 2019). In this thesis, we are mainly interested in document level text classification.

**Early Work** Text classification dates back to the early 1960s but only in the early 1990s did it become a major sub-field within the discipline of information systems due to the rapid increase of digital documents and the availability of more powerful computational resources (Sebastiani, 2002). Early work on text classification relied heavily on existing classification schemes and domain expert opinions (Van Looy and Magerman, 2019). One of the earliest applications of automatic text classification is automatic document indexing for information retrieval systems relying on a controlled dictionary (Maron, 1961; Borko and Bernick, 1963; Field, 1975; Gray and Harley, 1971). In these systems, each document is assigned one or more key words describing its content, where these key words belong to a finite set called a "controlled dictionary", designed by domain experts. Until the late 1980s, the most popular approach for the creation of automatic document classifiers was a knowledge engineering (KE) one, which relied on logical rules manually defined by domain experts. These rules encode expert knowledge on how to classify documents under the given categories.

Since the early 1990s, the machine learning approach to text classification has gained popularity and has eventually become the dominant approach. In the machine learning approach, a set of documents are manually classified under a category defined by a human annotator; then, a general inductive process (also called the learner) automatically learns the set of rules for classification by observing the features. The machine learning algorithm uses the feature and labeled examples to learn how to classify unseen documents into one of the categories. In machine learning terminology, this classification problem is an activity of supervised learning.

There has been a variety of techniques introduced in the literature to extract features for document representation. A simple approach for document representation that has been widely used in the literature is the “bag-of-words” (BOW) model, in which documents are represented as counts for occurrences of each word in a text. This model encodes information about the terms and their corresponding frequencies in a document without taking into account their locations in the sentence or document. This representation of a set of documents as vectors is also known as Vector Space Model (VSM), as each document is represented as a vector of term frequencies in the vocabulary (Salton *et al.*, 1975). The BOW model encodes every word in the vocabulary as a one-hot-encoded vector such that each of these terms in the vocabulary is represented by an independent (orthogonal) dimension in the vector space, which usually results in very high dimensional sparse vectors with only a few of them taking a frequency value.

Jones (1972) introduced the Inverse Document Frequency (IDF) method to be used in conjunction with term frequency to lessen the effect of frequent words in the corpus that appear in many documents. This combination of TF and IDF is well known as Term Frequency-Inverse document frequency (TF-IDF). The TF-IDF for the term $t$ in the document $d$ is defined as:

$$\text{TF-IDF}(t, d) = \text{tf}_{t,d} \times \log \frac{N}{\text{df}(t)}$$

Where $N$ is the total number of documents in the corpus, $\text{tf}_{t,d}$ is the frequency of $t$ in $d$, and $\text{df}(t)$ is the number of documents where the term $t$ appears at least once. The first factor in the equation, TF ($\text{tf}_{t,d}$), would contribute to improving the recall, while the second factor, IDF (df(t)), would contribute to improving the precision (Tokunaga and Makoto, 1994).

Since the introduction of the TF-IDF model, it has been widely used in the literature for a variety of tasks. Joachims (1998) shows that the TF-IDF model is very effective for topic categorization since that words related to topics are up-weighted and function words are down-weighted by the TF-IDF. Yu *et al.* (2007) use TF-IDF in combination with Naive Bayes and SVMs classifiers to classify ideology for political speech. Martineau and Finin (2009) propose Delta TF-IDF, an improved TD-IDF model for sentiment analysis system using SVMs.

One issue with BOW models (including TF-IDF) is that they do not capture the semantics of words. Specifically, the BOW model maps synonymous words into distinct vectors (Salton and Yang, 1973). For example, the words "airplane", "aeroplane", "plane", and "aircraft" are synonyms and often used in the same context, but the vectors corresponding to these words are orthogonal in the bag-of-words model.

**Word Embeddings** Word embedding is the collective name for a set of language modeling and feature learning techniques. It is a learned representation for text where words with the same meaning have a similar representation in the vector space. Unlike bag-of-words models in which semantically similar words might have orthogonal representation, word embedding models map each word in a vocabulary into a $d$-dimensional vector by creating a matrix in $\mathbb{R}^{N \times d}$ from a vocabulary with $N$ words such that semantically similar words have similar representations in the vector space. The theoretical framework for word embeddings is based on the distributional hypothesis, which says: "You shall know a word by the company it keeps" (Harris, 1954).

Mikolov *et al.* (2013a,b) propose the *word2vec* model, which uses a single layer neural network to learn word embeddings such that words that appear in the same context would have similar word representations. More recently, other ways of learning embeddings have been proposed, which rely not on neural networks and embedding layers but on leveraging word-context matrices to obtain word vector representations. Among the most influential models is the *GloVe* model (Pennington *et al.*, 2014). Another word embedding model that is widely used is *FastText* (Bojanowski *et al.*, 2017). It uses the sub-word information to enrich the word representation. Particularly, it provides an improvement over the word2vec model (Mikolov *et al.*, 2013b) whereby one learns not word embeddings, but character n-gram embeddings (which can be composed to form words). These embeddings are usually learned in an unsupervised setting such that word representations are obtained without the need for labeled corpora. Word embeddings models have shown improvement for different text classification tasks, including sentiment analysis (Jiang *et al.*, 2016; Joulin *et al.*, 2017) and newsgroup topic classification (Lilleberg *et al.*, 2015).

More recently, a novel technique of word representation was introduced in which word vector representations take into account the context in which the words appear. This technique is commonly referred to as "contextualized word representations". Models that use "contextualized word representations" generate embeddings such that the same word might have different representation given different contexts. This technique is useful for representing polysemous words such as "bank". Contextualized word embeddings models, such as *ELMo* (Peters *et al.*, 2018) and *BERT* (Devlin *et al.*, 2018), have achieved groundbreaking performance on a wide range of natural language processing tasks including text classification. These models generate contextualized word vectors after training on very large datasets and can be fine-tuned for different NLP tasks including text classification using relatively small datasets.

**Recent methods for document classification** In recent years, deep learning has gained incredible popularity for different machine learning tasks including natural language processing; and deep learning methods have started to be applied to text classification. In particular, Kim (2014) introduced Convolutional Neural Networks (CNNs) for text classification. Their architecture is a direct application of CNNs, as used in computer vision (LeCun *et al.*, 1998). Socher *et al.* (2013) use Recursive Neural Networks (RNNs) for sentence-level sentiment analysis. Tai *et al.* (2015) use tree-structured Long Short Term Memory networks (LSTMs) for different document classification tasks. Bahdanau *et al.* (2014) introduce the attention mechanism for machine translation as an improvement over the encoder decoder-based neural models. Since then, the attention mechanism has been used for different document classification tasks, including sentiment analysis (Wang *et al.*, 2016; Liu and Zhang, 2017; Ma *et al.*, 2018) and topic classification (Wang *et al.*, 2018a). Yang *et al.* (2016) propose hierarchical attention networks (HAN) for document classification by applying two levels of attention mechanisms: one for the word-level and the other for the sentence-level. They evaluate their models on two tasks: sentiment estimation and topic classification.

The attention mechanism has boosted the performance of sequential models such as RNNs for many NLP tasks. However, a crucial bottleneck of these models is the sequential processing at the encoding step, especially for longer sequence lengths. Vaswani *et al.* (2017) propose the *Transformer* architecture, which is based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. The *Transformer* architecture has been used as the base architecture for recent language models such as *BERT* (Devlin *et al.*, 2018) and OpenAI GPT (Radford *et al.*, 2018).

Most of the previous work on text classification focus on modeling the textual content of documents while ignoring information about the users who exchange the document. In this thesis, we are mainly interested in incorporating information from the underlying social network of communication in order to improve the text classification performance for different tasks. Specifically, we make use of existing methods for modeling the textual content of documents, and we propose methods for incorporating the social network information for text classification.

**Email Classification** A major topic in text classification is email classification. Despite the massive growth of other online social media, email remains a crucial communication medium for both individuals and organizations for both personal and business communications. Kiritchenko and Matwin (2011) show that a typical user daily receives 40-50 emails.

Despite the popularity of email, many machine learning tasks on emails have been hampered because of the lack of availability of task-related data, due to the privacy issues surrounding email. However, two large datasets are available. First, a large dataset of real emails, the Enron corpus Klimt and Yang (2004), was made publicly available by the Federal Energy Regulatory Commission (FERC) during the legal investigation of the company's collapse. Second, in February 2015, the Linguistic Data Consortium distributed a dataset of emails from an anonymous defunct information technology company referred to as Avocado Oard *et al.* (2015).

Since the Enron corpus was made public, many researchers have worked on the Enron corpus for different email classification tasks. One of related work to ours is Jabbari *et al.* (2006) who released "the Sheffield dataset", in which they categorize a subset of more than 12,000 Enron emails into two main categories "Business" and "Personal". Unlike our work, they do not utilize email thread structure, and many emails in the Sheffield dataset are not part of a thread, and some threads are partially labeled. They also present a preliminary experiment for automatic classification of personal and business. We show in chapter 4 our work on annotating emails in which we maintain the thread structure.

The Sheffield dataset has been used in other studies. In particular, Peterson *et al.* (2011) show that the formality level in emails is affected by the interpersonal nature of email (personal or business). They use email gold labels in the Sheffield dataset to determine the email type. Mitra and Gilbert (2012) use the Sheffield dataset to study the proportion of gossip in business and personal emails. Unlike formality in Peterson *et al.* (2011), they find that gossip appears in both personal and business emails and at all levels of the organizational hierarchy and the proportion of gossip email is independent of whether the email is business or personal. In our work, we focus on automatic classification of emails into business and personal. We show in Part II our work on email classification into business and personal.

Prabhakaran *et al.* (2012a) introduce a typology of different types of power relations between dialog participants. They also present an annotated corpus of Enron emails with instances of these power relations between participants. Sappelli *et al.* (2016) categorize email task and intent from multiple dimensions by analyzing message content. They use different email collections including Enron and Avocado.

## 2.2 Incorporating Social Network

Many previous studies on various NLP tasks in the context of social networks mainly focus on textual information and ignore other information that can be extracted from the underlying social network. However, there have been some studies that utilize the social network structure to improve the classification performance for different tasks including: inferring user attributes (Filippova, 2012; Al Zamal *et al.*, 2012; Perozzi and Skiena, 2015; Aletras and Chamberlain, 2018) predicting user stance (Tan *et al.*, 2011; West *et al.*, 2014; Gryc and Moilanen, 2014; Gui *et al.*, 2017; Wang *et al.*, 2018b; Volkova *et al.*, 2014), and extracting inter-personal relations (Krishnan and Eisenstein, 2014; West *et al.*, 2014; Abu-Jbara *et al.*, 2013; Hassan *et al.*, 2012). Most of these studies exploiting social network information are guided by an assumption of *homophily*, i.e., the tendency of individuals to associate and bond with similar others (McPherson *et al.*, 2001). Our work differs from these studies in that we focus on classifying a given document (i.e., email) exchanged between users, not on predicting user information, nor interpersonal relations; except for chapter 16, where we model hierarchical power relations using email exchanged between pairs of people.

**Incorporating Social Network for Email Classification** There has been some previous work on incorporating email communication network information for different email classification tasks. Yoo *et al.* (2009) propose a semi-supervised method for personalized email prioritization. They find that including social features along with message content based features leads to a significant reduction in the prediction error when learning to identify the emails that a given user will consider important. Another task is to predict the recipient of an email. Graus *et al.* (2014) propose a generative model to predict the recipient of an email and report that the optimal performance is achieved by combining features from both the communication graph and email content. Similar to our work, they use both Enron and Avocado. Our work is similar to Wang *et al.* (2012), who propose a model for email classification into "Business" and "Personal". However, unlike our work, they don't use the email content. Their approach requires that the users (i.e., sender and recipients) have been seen in the labeled training data. Therefore, their approach cannot generalize to unseen users, let alone a new corpus (i.e., another email exchange). In contrast, our models do not require users to be seen before and can generalize to unseen nodes and new networks.

## 2.3 Graph Embeddings

Graphs are an important data representation which occur naturally in various real-world applications, and graph analytics has been used in various tasks, including: node classification (Wang *et al.*, 2017; Sen *et al.*, 2008; Jian *et al.*, 2018), link prediction (Wei *et al.*, 2017; Pachev and Webb, 2017), and community detection (Fortunato, 2010; Cavallari *et al.*, 2017).

Node embedding (a.k.a. graph or network embedding) aims to learn low-dimensional representations for nodes in graphs. Recently, network embedding methods have gained attention from the research community. Many recent node embedding models are inspired by neural language embedding models such as word2vec (Mikolov *et al.*, 2013a). These graph embeddings models include:

*DeepWalk* (Perozzi *et al.*, 2014), and *node2vec* (Grover and Leskovec, 2016). In these graph embedding models, a graph is represented as a set of sampled random walk paths. The embeddings for nodes then are learned in an unsupervised approach by applying the *word2vec* model (Mikolov *et al.*, 2013a) on the sampled paths. Hamilton *et al.* (2017b) categorize these models under *shallow learning* approaches as they are inherently *transductive* and do not naturally generalize to unseen nodes. In our work, we are interested in applying models for email classification to new datasets.

GraphSAGE (Hamilton *et al.*, 2017a; Hamilton, 2018) is an inductive graph embedding model. Unlike transductive models, it generalizes to unseen nodes and new graphs without requiring retraining. To do so, it learns a function that maps a node to low-dimensional representation by aggregating neighboring nodes' attribute information. We use GraphSAGE in all classification tasks in this thesis, and we compare its performance with our proposed models.

# Part I

# Data Creation, Data Analysis, and Methods

# Chapter 3

# Existing Datasets

In this chapter, we present existing datasets that we make use of them in this thesis. We describe in detail the source of the data and some processing we perform on these datasets. We discuss our new annotations and datasets in the following chapter (chapter 4). In this thesis, we are interested in exploiting the social network information for text classification. We apply our methods on tasks in two genres: email and online discussion forums. For email, we use email collections from two companies: Enron and Avocado. We use a dataset from Reddit for the online discussion genre. Despite the popularity of email, many machine learning tasks on emails have been hampered because of the lack of availability of task-related data, due to the privacy issues surrounding email. However, two large datasets are available. First, a large dataset of real emails, the Enron corpus, was made publicly available by the Federal Energy Regulatory Commission (FERC) during the legal investigation of the company's collapse. Second, in February 2015, the Linguistic Data Consortium distributed a dataset of emails from an anonymous defunct information technology company referred to as Avocado (Oard *et al.*, 2015).

The chapter is organized as follows; we first discuss the Enron email collection and datasets derived from it in section 3.1. Then, in section 3.2, we present another email corpus, the Avocado email collection. Finally, we discuss the third dataset, the Reddit post collections, in section 3.3.

## 3.1 Enron

In this section, we first give a brief history of the Enron Corporation from its formation until its decline. Then, we introduce the Enron Email Corpus. Following that, we present existing datasets based on the Enron corpus that we use in this thesis.

### 3.1.1 Brief History of Enron

In this subsection, we give a brief history of Enron. For more details, the "Enron—What Happened" [1] article from the Britannica provides an excellent and concise summary of the Enron corporation and the significant events that led to Enron's decline.

Enron was an American energy and services company based in Houston, Texas. It filed for bankruptcy in 2001 and was the largest bankruptcy at that time. In 1985, Enron was formed as a merger of two small regional companies: Houston Natural Gas in Texas and InterNorth in Nebraska. The company started as a natural gas provider and continued growing and expanding until its bankruptcy before the end of 2001. It became a major electricity, natural gas, communications, and pulp and paper company. It had approximately 29,000 employees with claimed revenues of nearly $101 billion during 2000. [2]

In October 2001, a major accounting scandal was publicized. It was revealed that Enron's reported financial condition was sustained by an institutionalized, systemic, and creatively planned accounting fraud, known since as the Enron scandal, which eventually led to the bankruptcy of the

[1] `https://www.britannica.com/topic/Enron-What-Happened-1517868`

[2] `https://archive.fortune.com/magazines/fortune/fortune500_archive/snapshots/2001/478.html`

Enron Corporation. It was one of corporate America's biggest scandals and the largest bankruptcy reorganization in the U.S. history at that time.

In less than a year, Enron had gone from being considered one of the most innovative companies of the late 20th century to being deemed a byword for corruption and mismanagement. In fact, Fortune named Enron ”America's Most Innovative Company” for unprecedented six consecutive years from 1995 through 2000. Also, the Financial Times awarded Enron the ‘energy company of the year’ award in 2000 (Dobson, 2006).

### 3.1.2 Enron Email Corpus

In May 2002, during the legal investigation of the company's collapse, the Federal Energy Regulation Commission (FERC) released the Enron corpus on the web. The corpus contained around 600K emails from the mailboxes of 158 Enron employees at the top level.

The email corpus included the information about the sender, the set of recipients, date, time, subject, and the email body. The email attachments were not included in the initial release. We refer to these 158 users throughout this thesis as the **core** Enron group. After the initial release, various researchers noticed many integrity issues in the corpus. Subsequently, the corpus underwent many iterations of cleaning up and reformatting, which resulted in many different versions of the corpus. Since then, the dataset has been used for various natural language processing (NLP) and social network analysis (SNA) applications.

As one of the earliest works on the Enron email collection, Klimt and Yang (2004) at Carnegie Mellon University (CMU) performed the first major iteration of cleaning up and fixing some data integrity issues. They provided a usable version of the dataset for the research community. They report that the raw Enron corpus contains 619,446 messages belonging to 158 users. After cleaning the corpus by removing some messages such as duplicates and computer-generated ones, they obtained a total of 200,399 messages belonging to the core 158 users with an average of 757 messages per user. We refer to this dataset as the "CMU/CALO" dataset. A further cleaning up was done by Shetty and Adibi (2004) at the ISI, who released a MySQL version of the corpus based on the work of Klimt and Yang (2004). This dataset is commonly referred to as the "ISI" dataset. Later, Diesner and Carley (2005) added the position and location information to the dataset.

In a separate line of work, Yeh and Harnly (2006) used the original Enron release from the FERC to construct the thread structure for Enron emails. They automatically assembled the thread information of the emails in the corpus. They restored some missing emails from their quoted form in other emails. They also co-reference multiple email addresses belonging to one employee and assign unique identifiers and names to employees. Therefore, each employee is associated with a set of email addresses and names. Agarwal *et al.* (2012) added the organizational hierarchy information to this dataset. We discuss this release in subsection 3.1.4; we refer to this release as "Columbia release".

### 3.1.3 Enron Overt Display of Power Corpus ENRON-ODP-UTTERANCE

In this thesis, we make use of the Enron Over Display of power corpus (Prabhakaran *et al.*, 2012b). This corpus is an extension of a previously annotated corpus presented in Hu *et al.* (2009), which was annotated with Dialog Functional Units (DFU). The corpus contains 122 email threads with 360 messages, with each message segmented into a sequence of DFUs with 1734 utterances and 20,740 word tokens.

The ENRON-ODP-UTTERANCE corpus is annotated with four types of power: hierarchical power, situational power, influence, and control of communication. The dataset is also annotated at the utterance level with overt display of power instances. Table 3.1 summarizes the data.

We use this corpus in chapter 15 for our work on detecting overt display of power at an email level. We extend the ODP-UTTERANCE dataset to the email level ODP.

| | | |
|---|---|---|
| Threads | | 122 |
| Utterances | Total | 1734 |
| | Utterances with ODP (POS) | 86 (95.04%) |
| | Utterances without ODP (NEG) | 1648 (4.96%) |

Table 3.1: Summary of the ENRON-ODP-UTTERANCE dataset (Prabhakaran *et al.* (2012b)).

### 3.1.4 Enron Organizational Hierarchy Dataset ENRON-POWER (Columbia Release)

Another Enron dataset in the literature is the Enron Organizational Hierarchy Dataset by Agarwal *et al.* (2012). [3,4] It is a MongoDB database containing hierarchical relation information of Enron's employees as well as departments. It is based on the work of Yeh and Harnly (2006). The corpus contains 279,844 email messages that belong to 93,421 unique email addresses. Additionally, this release maintains the thread structure.

Persons and their email addresses are stored as a MongoDB collection named "Entries". Of which, there are 3,187 entries with the hierarchy information. Some of these entries represent employees, and other entries represent departments.

Some of the entries representing people have multiple nodes distinguishing between various positions for the same person at various points in time. Among these entries, there are 1,518 entries having user ids such that we can map it to the emails (senders or recipients). The dataset was constructed by studying the original Enron organizational charts found in emails. Ar earlier attempt to predict Enron's organizational hierarchy was made by Shetty and Adibi (2004). They assembled the set of job titles of the core 158 Enron employees (with full mailboxes released). However, there

[3] `http://www.cs.columbia.edu/~vinod/data/gender_identified_enron_corpus.tar.gz`

[4] `http://www.cs.columbia.edu/~rambow/enron/`

are limitations of this gold standard:

- It is small: as it contains information about only 158 entities.
- It does not have hierarchical information: it states job title, but there is no information about whether or not two entities are professionally related.

The original Enron release from the FERC has 158 mailboxes belonging to the Enron core group. However, in later releases, some mailboxes representing different people were merged together. We show in Appendix B the list of mailboxes as in the original release by the FREC as well as the list of mailboxes after they were merged in other releases.

## 3.2 Avocado

Another publicly available email corpus is the Avocado Research Email Collection (Oard *et al.*, 2015), distributed by the Language Data Consortium (LDC) .[5] The corpus is available with a license that prevents reproducing any part of the collection. For this license restriction, we limit the examples discussed in this thesis to the Enron corpus.

The Avocado corpus consists of emails and attachments taken from 279 accounts of a defunct information technology company referred to as “Avocado”. The full Avocado corpus contains 938,035 emails that were sent or received between 1995 and 2003. Most of the accounts are those of Avocado employees; the remainder represent shared accounts such as ”Leads”, or system accounts such as ”Conference Room Upper Canada”. The original Avocado email collection from the LDC is divided into metadata and text. The metadata files describe folder structure, email characteristics, and contacts. The text files contains the extracted text of the items in the account’s folder.

[5] `https://catalog.ldc.upenn.edu/LDC2015T03`

Unlike Enron, only a few studies make use of it. Sappelli *et al.* (2016) annotate a subset of the Avocado corpus with e-mail intent and task intent using proposed annotation schemes. They also studied predicting the number of tasks in emails. Graus *et al.* (2014) use Avocado to evaluate models trained on Enron for recipient recommendation.

## 3.3 Reddit

Reddit is an American online social news aggregation, web content rating, and discussion platform. It has more than a million communities (forums) known as "subreddits", where people can post news and content or comment on other people's posts. As of 2020, Reddit is ranked 6th as the most visited website in the United States and 20th globally. [6]

In this thesis, we are interested in Reddit post classification into the subreddits they belong to. We discuss this task in chapter 17.

Reddit raw data can be pulled via the Reddit API. However, it is a time-consuming process, especially if the requested dataset is massively large. Hamilton *et al.* (2017a); Hamilton (2018) released a dataset for this task as part of their work on a graph embedding model GraphSAGE. This dataset was a result of a preprocessing of a larger dataset. We discuss both datasets in the following subsections.

### 3.3.1 Archive's Reddit Dataset REDDIT-FULL

In 2015, massive datasets for Reddit posts and comments were released by the Reddit user "Stuck-_In_the_Matrix". The datasets became later available on the Internet Archive. [7] The collection consists of two subsets: Reddit submission corpus and Reddit comment corpus. Entities of the

[6] `https://www.alexa.com/topsites`

[7] `https://archive.org/`

datasets are represented as JSON objects with all fields that are available through Reddit's API.

#### 3.3.1.1 Full Reddit Submission Corpus (REDDITSUBMISSION-FULL):

This corpus consists of posts (submissions) of more than 200 million posts represented as JSON objects with all attributes for posts with a size of approximately 42 GB. Attributes include *score data, author, title, self_text, media tags*, and all other attributes available via the Reddit API. The data is complete from January 01, 2008, thru August 31, 2015, with partial data available for an earlier time.

This subset can be downloaded from `https://archive.org/details/FullReddit SubmissionCorpus2006ThruAugust2015`

#### 3.3.1.2 Reddit Comments Corpus (REDDITCOMMENTS-FULL)

The second subset of REDDIT-FULL is a collection of Reddit comments from October of 2007 until May of 2015. Comments in Reddit are direct replies to a given post or other comments in a post. This corpus contains more than 1.7 billion comments as JSON objects containing comment text, score, author name, subreddit, position in comment tree, and other fields that are available through Reddit's API. This dataset is over 1 terabyte uncompressed and can be downloaded from `https://archive.org/details/2015_reddit_comments_corpus`.

### 3.3.2 GraphSAGE Reddit Posts Dataset GRAPHSAGE-REDDIT

As part of their work, the authors of GraphSAGE (Hamilton *et al.*, 2017a; Hamilton, 2018) released a processed subset of REDDIT-FULL. They sampled posts from 50 large communities made in the month of September 2014. It consists of 232,965 posts. They process the REDDIT-FULL dataset by construing a post-to-post graph that contains posts as nodes, and they link two nodes (posts) if at least one user comments on both posts. The average degree for posts (nodes) is 492. Additionally, they extract features for posts from the original dataset. They concatenate three different feature sets:

(i) The average embedding of the post title.

(ii) The average embedding of all the post's comments.

(iii) The post's score.

(iv) The number of comments made on the post.

For lexical embeddings (i and ii), they use off-the-shelf 300-dimensional GloVe CommonCrawl word vectors (Pennington *et al.*, 2014). The dataset contains the following files:

- *reddit-G_full.json* A networkx-specified JSON file describing the input graph. Nodes have 'val' and 'test' attributes specifying if they are a part of the validation and test sets, respectively.
- *reddit-class_map.json* A JSON-stored dictionary mapping the graph node ids to classes.
- *reddit-id_map.json* A JSON file mapping the Reddit post graph node ids to consecutive integers.
- *reddit-feats.npy* A numpy-stored array of node features; ordering given by id_map.json.

This dataset does not explicitly contain information about users who commented on different posts, and we cannot directly retrieve such information from this dataset. Notably, the files do not have information about who is the author of a given post nor information about who made a comment on a post.

# Chapter 4

# New Datasets: Business and Personal Emails

In this chapter, we introduce new datasets that we use in this thesis. The main contribution in terms of datasets is for the task of email classification into business and personal. For this task, we use two corpora: Enron and Avocado. Additionally, we have extended existing datasets discussed in chapter 3 for other tasks:

- Overt Display of Power: we discuss the extended dataset in chapter 15.
- Hierarchical Power Detection in Emails: we discuss the dataset in chapter 16.
- Reddit post classification: we make use of the Reddit datasets discussed in section 3.3 in chapter 17.

For the task of email classification into business and personal, we have annotated subsets of Enron and Avocado corpora into five categories of “Business” and “Personal”. We have used the Amazon Mechanical Turk (AMTurk) crowdsourcing platform to annotate a subset of the Enron corpus. In addition, due to the license constraints, we have hired two undergraduate students to annotate a subset of the Avocado corpus. In our study, we use these two sets as well as the Enron dataset distributed by Jabbari *et al.* (2006) (which we refer to as the “Sheffield set”).

We first present the annotation scheme used for labeling emails into business and personal in section 4.1. Then, we discuss in detail our annotation results for the two corpora: Enron and Avocado. We discuss Enron in section 4.2 and Avocado in section 4.3. Finally, we show results for inter-annotator agreement in the datasets in section 4.4.

## 4.1 Annotation Scheme

Annotators were given email threads of various lengths and asked to annotate each email in the thread and to annotate the thread as a whole. We randomly sampled threads from each corpus, and we do not perform a manual investigation by removing short or ambiguous emails as the goal of our study is to predict the class for an email in real scenarios.

Classifying email content into business and personal can be subjective and hard when emails are ambiguous. For example, suppose an email is about an invitation to a picnic for employees’ families. In that case, one annotator might label this email as a business email with the perspective that the email is about a business-related event. On the other hand, another annotator might have a perspective that this is a personal event even though it is organized by the company. Therefore, to simplify the task for the annotators and to clear up the ambiguity, we have provided the annotators with detailed instructions to annotate each email with one of the following six labels and criteria:

1. Business: the content of the message is clearly professional (even if the language used is very friendly), and it does not contain any personal content; it should be related to the company’s work.

2. Somehow Business: the main purpose of the message is professional, but it has some personal

parts.

3. Mixed: the content of the message belongs to two or more of the categories (typically because the sender combines different content in one email).

4. Somehow Personal: the main purpose of the message is personal, but it has some business-related content.

5. Personal: the content of the message is clearly personal (even if the language used is very formal), and it does not contain any professional part.

6. Cannot Determine: if there is no enough content to determine the category.

We added some detailed instructions to deal with certain cases:

- If a message is about a social event inside the company, such as celebrating a new baby of an employee, or a career promotion, it belongs to the second category ("somehow business").

- If a message is about a social event outside the company but still related to the company, such as a picnic (usually family members are invited), it belongs to the fourth category ("somehow personal").

- If a message is about a social event which is not related to the company, such as a charity, but company employees are encouraged to participate, it belongs to the fourth category ("somehow personal").

- If a message is too short to determine its category (or even empty), it should have the same category as the message it is responding to, or the message it is forwarding.

- If a message is ambiguous, try to read other messages in the thread to clarify.

- If a message is a spam or in the rare case that the first message of a thread is very short or empty, say "cannot determine".

In this thesis, we are interested in binary classification of emails into "business" and "personal". The complex labeling scheme described here will be useful for different tasks in the future. However, even with the detailed instructions given to annotators, there were many cases such that annotators have not agreed on a label.

Table 4.1 shows an example of a thread where each annotator assigned a different label. This thread begins with an email, which is a reply to another email (not in the thread) sent to many employees at Enron. The employee (Ana Castanon) requests to remove her from the mailing list. The second email is the same as the first but with a different sender who copied and pasted the first email. The last email is similar to the two emails but with an email signature added.

For the goal of our study, we aim to group these labels into binary classes: business and personal. We normalize the labels as follows: we group "Business" and "Somehow Business" into one category, "Business", and "Personal", "Somehow Personal" and "Mixed" into one category, "Personal", while "Cannot Determine" remains the same.

Table 4.2 and Table 4.3 show examples of a business and a personal thread in the Enron corpus, respectively. In these threads, all annotators gave the same label "business" and "personal", respectively.

| Message | Labels |
|---|---|
| First Message<br>**Subject:** Reply Requested: Do You Code Or Approve Invoices?<br>**From:** Ana Castanon<br>**To:** Chris Nowak, Marie Newhouse, iBuyit,<br>All Enron Employees United States, Eric Linder<br>————————————<br>Take me off of this list. PLEASE!!!!! | First Annotator: Personal<br>Second Annotator: Business<br>Third Annotator: Mixed |
| Second Message<br>**From:** Rebecca Torres<br>**To:** Ana Castanon, Chris Nowak, Marie Newhouse, iBuyit<br>, All Enron Employees United States, Eric Linder<br>————————————<br>Take me off of this list. PLEASE!!!!! | First Annotator: Personal<br>Second Annotator: Business<br>Third Annotator: Mixed |
| Third Message<br>**From:** Jason McMahon<br>**To:** Ana Castanon, Chris Nowak, Marie Newhouse, iBuyit,<br>All Enron Employees United States, Eric Linder<br>————————————<br>Take me off this list please | First Annotator: Personal<br>Second Annotator: Business<br>Third Annotator: Mixed |

Table 4.1: Example of a thread from the Enron corpus in which annotators assigned different labels to each email

First Message

**Subject:** Master Agreement

**From:** Richard Weiss

**To:** Sara Shackleton;

______________________________

Sara:

All of your comments in regard to the guarantee are fine.

I am enclosing a copy of the legal opinion that we will provide.

Please forward an execution copy of the master agreement. We are ready to sign.

Second Message

**From:** Sara Shackleton

**To:** Stephanie Panus

______________________________

*forward*

Third Message

**From:** Stephanie Panus

**To:** Richard Weiss; Sara Shackleton; Kaye Ellis

______________________________

Richard,

Can you please send us the finalized version of the form of Lehman

guaranty to include as Exhibit A with the execution copy of the Master Agreement?

Table 4.2: Example of a business thread from Enron in which annotators assigned the same labels to each email. The second massage is forwarding the first message.

First Message

**Subject:** Concert

**From:** Carol St Clair

**To:** Suzanne Adams

---

Suzanne:

Can you find out when Madonna will be in Philly?

We will be there from 6/30 through 7/4

and I was hoping that she would be there at that time.

Can you check for her dates in NY or DC as well? Thanks.

Second Message

**From:** Suzanne Adams

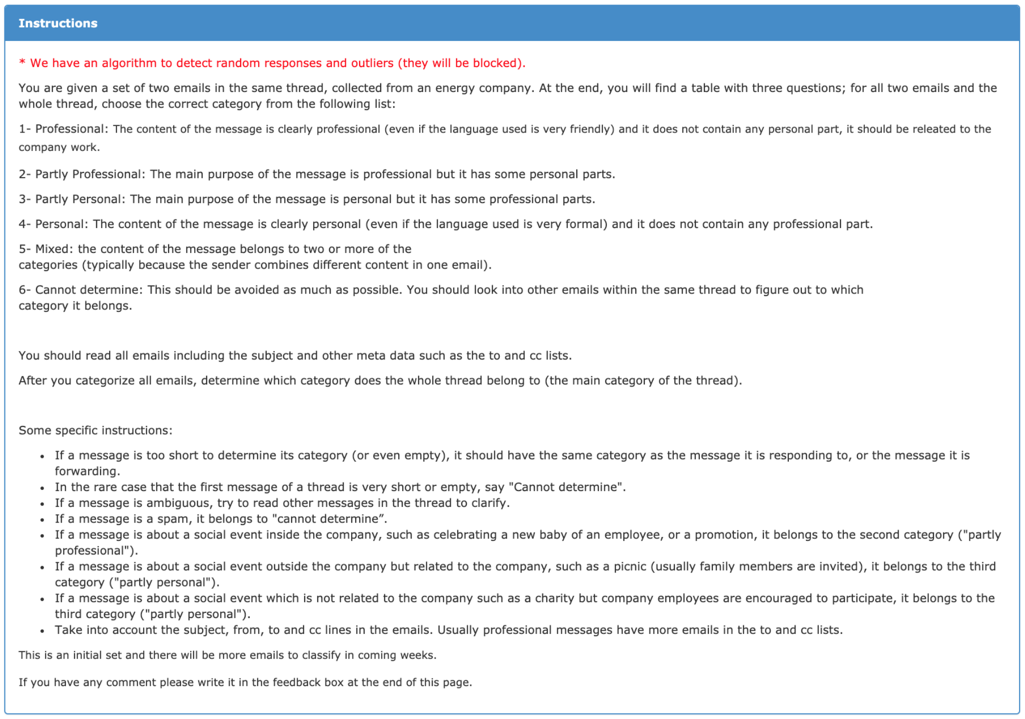
**To:** Carol St Clair

---

Here's the web page with her schedule on it. It looks like it's going to be the end of July.

*url*

Table 4.3: Example of a personal thread from Enron in which annotators assigned the same labels to each email.


**Instructions**

* We have an algorithm to detect random responses and outliers (they will be blocked).

You are given a set of two emails in the same thread, collected from an energy company. At the end, you will find a table with three questions; for all two emails and the whole thread, choose the correct category from the following list:

1- Professional: The content of the message is clearly professional (even if the language used is very friendly) and it does not contain any personal part, it should be releated to the company work.

2- Partly Professional: The main purpose of the message is professional but it has some personal parts.

3- Partly Personal: The main purpose of the message is personal but it has some professional parts.

4- Personal: The content of the message is clearly personal (even if the language used is very formal) and it does not contain any professional part.

5- Mixed: the content of the message belongs to two or more of the
categories (typically because the sender combines different content in one email).

6- Cannot determine: This should be avoided as much as possible. You should look into other emails within the same thread to figure out to which
category it belongs.

You should read all emails including the subject and other meta data such as the to and cc lists.

After you categorize all emails, determine which category does the whole thread belong to (the main category of the thread).

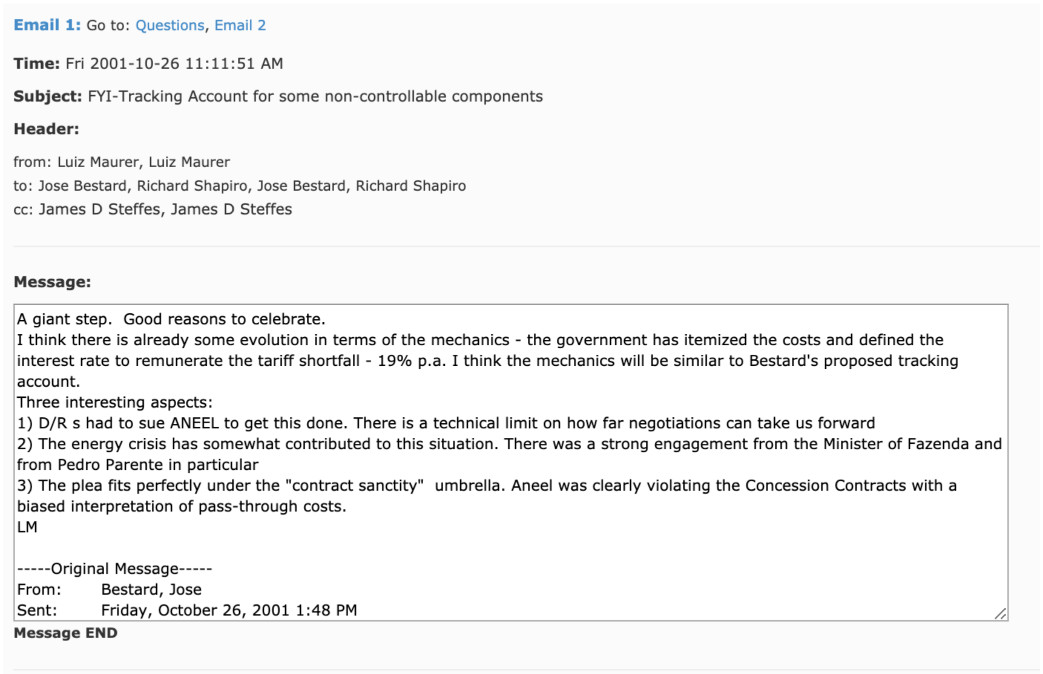
Some specific instructions:

- If a message is too short to determine its category (or even empty), it should have the same category as the message it is responding to, or the message it is forwarding.
- In the rare case that the first message of a thread is very short or empty, say "Cannot determine".
- If a message is ambiguous, try to read other messages in the thread to clarify.
- If a message is a spam, it belongs to "cannot determine".
- If a message is about a social event inside the company, such as celebrating a new baby of an employee, or a promotion, it belongs to the second category ("partly professional").
- If a message is about a social event outside the company but related to the company, such as a picnic (usually family members are invited), it belongs to the third category ("partly personal").
- If a message is about a social event which is not related to the company such as a charity but company employees are encouraged to participate, it belongs to the third category ("partly personal").
- Take into account the subject, from, to and cc lines in the emails. Usually professional messages have more emails in the to and cc lists.

This is an initial set and there will be more emails to classify in coming weeks.

If you have any comment please write it in the feedback box at the end of this page.

(a) Instruction segment.


**Email 1:** Go to: Questions, Email 2

**Time:** Fri 2001-10-26 11:11:51 AM

**Subject:** FYI-Tracking Account for some non-controllable components

**Header:**

from: Luiz Maurer, Luiz Maurer
to: Jose Bestard, Richard Shapiro, Jose Bestard, Richard Shapiro
cc: James D Steffes, James D Steffes

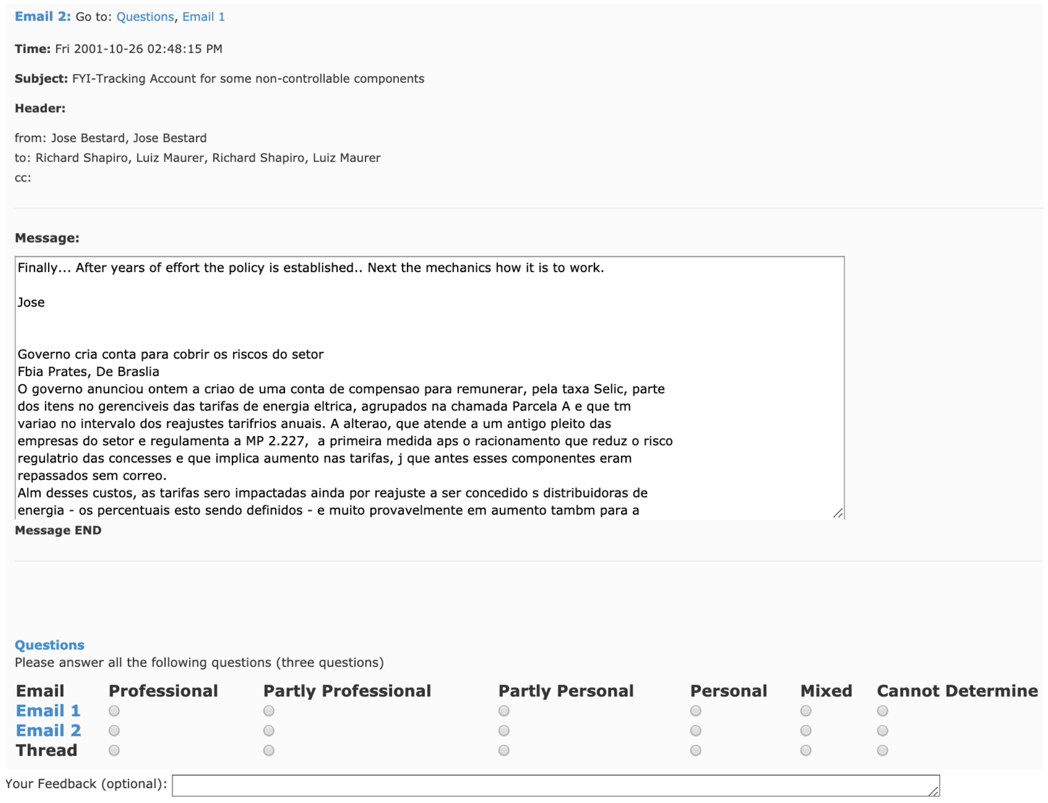
**Message:**

```
A giant step.  Good reasons to celebrate.
I think there is already some evolution in terms of the mechanics - the government has itemized the costs and defined the
interest rate to remunerate the tariff shortfall - 19% p.a. I think the mechanics will be similar to Bestard's proposed tracking
account.
Three interesting aspects:
1) D/R s had to sue ANEEL to get this done. There is a technical limit on how far negotiations can take us forward
2) The energy crisis has somewhat contributed to this situation. There was a strong engagement from the Minister of Fazenda and
from Pedro Parente in particular
3) The plea fits perfectly under the "contract sanctity"  umbrella. Aneel was clearly violating the Concession Contracts with a
biased interpretation of pass-through costs.
LM

-----Original Message-----
From:      Bestard, Jose
Sent:      Friday, October 26, 2001 1:48 PM
```

**Message END**

(b) First Email.


**Email 2:** Go to: Questions, Email 1

**Time:** Fri 2001-10-26 02:48:15 PM

**Subject:** FYI-Tracking Account for some non-controllable components

**Header:**

from: Jose Bestard, Jose Bestard
to: Richard Shapiro, Luiz Maurer, Richard Shapiro, Luiz Maurer
cc:

**Message:**

```
Finally... After years of effort the policy is established.. Next the mechanics how it is to work.

Jose


Governo cria conta para cobrir os riscos do setor
Fbia Prates, De Braslia
O governo anunciou ontem a criao de uma conta de compensao para remunerar, pela taxa Selic, parte
dos itens no gerenciveis das tarifas de energia eltrica, agrupados na chamada Parcela A e que tm
variao no intervalo dos reajustes tarifrios anuais. A alterao, que atende a um antigo pleito das
empresas do setor e regulamenta a MP 2.227,  a primeira medida aps o racionamento que reduz o risco
regulatrio das concesses e que implica aumento nas tarifas, j que antes esses componentes eram
repassados sem correo.
Alm desses custos, as tarifas sero impactadas ainda por reajuste a ser concedido s distribuidoras de
energia - os percentuais esto sendo definidos - e muito provavelmente em aumento tambm para a
```

**Message END**

**Questions**
Please answer all the following questions (three questions)

| Email | Professional | Partly Professional | Partly Personal | Personal | Mixed | Cannot Determine |
|---|---|---|---|---|---|---|
| Email 1 | ○ | ○ | ○ | ○ | ○ | ○ |
| Email 2 | ○ | ○ | ○ | ○ | ○ | ○ |
| Thread | ○ | ○ | ○ | ○ | ○ | ○ |

Your Feedback (optional):

(c) Second email and questions.

Figure 4.1: Example of a HIT page with 2 emails. a shows the instructions segment, b shows the first email, and c shows the second email and questions.

## 4.2 Enron

In this section, we present the Enron datasets we use throughout this thesis. In this thesis, we conduct experiments on Enron using datasets based on two releases: the Sheffield release (Jabbari *et al.*, 2006) and the Columbia release (subsection 3.1.4). We first discuss the annotation process in subsection 4.2.1. Then, discuss each release in subsection 4.2.2 and subsection 4.2.3. Finally, in subsection 4.2.4, we present the final datasets that we will be using in the rest of this dissertation.

### 4.2.1 Annotation using AMTurk

Amazon Mechanical Turk [1] (AMTurk) is a marketplace for completing virtual tasks that require human intelligence. We use it as a crowd-sourcing platform for annotating Enron emails into "business" and "personal". A Human Intelligence Task, or HIT, is a set of questions that needs an answer. A HIT represents a single, self-contained virtual task. To make it easier for annotators, we created different HITs such that each HIT contains a whole thread with all of its emails. Figure 4.1 shows the page layout for AMTurk HITs. We ask annotators to assign a label for each email in the thread and for the whole thread. We also have an optional field *feedback* if the annotator has a specific issue that we have not addressed in the instructions.

We first ran a pilot test for HITs, which is identical to the actual one. We internally tested the interface to make sure that it is clear and easy. Then, we ran a small batch of HITs to test how annotators on AMturck would perform. In the small batches, we assigned 5 annotators to each HIT. Then, by studying the annotation results of the first set of batches, we found that labeling emails by three annotators is sufficient and cost less. Therefore, we decided to limit the number of annotators per HIT to three. Thus, most of the Enron emails and threads were annotated by 3 Turkers. Note that the group of Turkers is not fixed among threads as it differs from a thread to another.

To determine the consensus label, we assign each of the categories in the list mentioned in section 4.1 a numerical value between 1 and 6, with 6 being "cannot determine" and otherwise a larger number indicating that the email is more personal and a smaller number indicating that the email is more business. Then, we discard any "cannot determine" label, and if there are one or more labels other than "cannot determine", we limit voting to these labels. If all labels are "cannot determine", the voting's final result is "cannot determine" too. Then, we compute the majority vote

[1] `https://www.mturk.com/`

for all labels from the three Turkers. In the case of ties, we take the floor of the mean for the ties' ordinal values. Finally, we normalize the labels as follows: we group "Business" and "Somehow Business" into one category, "Business", and "Personal", "Somehow Personal" and "Mixed" into one category, "Personal". For instance, if the labels are 1, 2, 6, the majority vote result is 1, 2. The mean is 1.5, and the floor is 1. The final label is 1 "Business". Another example is if the labels are 1, 5, 6, the majority vote result is 1, 5. The mean is 3. The final label, after normalization, is "Personal".

| Set | # Business Emails | # Personal Emails | Total |
|---|---|---|---|
| original release | 11,220 (75.7%) | 3,598 (24.3%) | 14,818 |
| after deleting duplicates | 9,857 (75.7%) | 3,168 (24.3%) | 13,025 |
| emails that are part of threads | 3,966 (81.6%) | 895 (18.4%) | 4,861 |
| emails that are not part of threads | 5,891 (72.2%) | 2273 (27.8%) | 8,164 |

Table 4.4: Summary of the Sheffield Dataset.

### 4.2.2 Sheffield Release

In 2006, Jabbari *et al.* (2006) released "the Sheffield dataset", in which they categorize a subset of more than 12,000 Enron emails into two main categories "Business" and "Personal". Unlike our work, they do not utilize email thread structure, and many emails in the Sheffield dataset are not part of a thread, and some threads are partially labeled. Note that, their annotation scheme is different than ours. Our annotation scheme considers the content of the emails as the main dimension to be considered in the annotation. We provide the annotators with detailed instructions about each category. In their work, final categories were created to reflect the topic as the only dimension considered in the annotation. Table 4.4 shows statistics of the Sheffield release.

| | |
|---|---|
| Total number of sampled threads | 3,943 |
| Total number of emails in threads | 11,025 |
| Number of threads that all emails have labels | 3,923 |
| Number of emails in these threads | 10,964 |
| Total number of emails with no label | 27 |
| Number of threads that only some emails have labels | 18 |
| Total number of emails in these threads | 57 |
| Number of threads that all emails have no labels | 2 |
| Number of emails in these threads | 4 |
| Total number of bubble emails in the sample | 470 |
| Number of threads that contain at least one bubble email | 441 |
| Number of emails in these threads | 1,660 |
| Final number of emails | 10,551 |
| Final number of emails (labeled) | 10,528 |

Table 4.5: Summary of our annotated Enron corpus.

### 4.2.3 Our Dataset

The annotated emails by Turkers (subsection 4.2.1) are a subset of the "Columbia Enron corpus" (subsection 3.1.4) released by Agarwal *et al.* (2012), which has more than 36,000 threads and 270,000 emails. We choose this version of Enron because, unlike Jabbari *et al.* (2006), it maintains the thread structure of emails. Particularly, we are interested in the sequential modeling of emails using the thread structure.

The set of users in Enron is divided into two parts: **core** and **non-core**. From this collection, we have randomly sampled around 4000 threads with different numbers of emails per thread (2, 3, 4, and 5). The total number of emails is 11,025. The set of core people are those whose inboxes were taken to create the Enron email network (a set of 158 people). The set of non-core people are the remaining people in the network who send an email to and/or receive an email from a member of the core group. The sample has 3,222 emails overlapping (after excluding "Cannot determine" emails) with the Sheffield set of Jabbari *et al.* (2006).

**Bubble emails** The Columbia release is based on an earlier one, released by Yeh and Harnly (2006), which also maintains the thread structure. Some of the emails in the threads were extracted from quoted emails, and these quoted emails do not have sender and recipient information. This kind of email is called "bubbles" (Yeh and Harnly, 2006). Since we are interested in incorporating social network information for email classification, we need the information about the sender and recipients. Therefore, we will discard these bubble emails in our study.

Table 4.5 shows the summary of labeled emails after the annotation process described in section 4.2.1. The first two lines show the number of sampled threads and emails, respectively. In the second box, the first line shows the number of threads in which all emails were assigned a label (other than "Cannot Determine"); the second line shows the total number of emails in these threads. In the third box, the first line shows the number of emails without a label (labeled as "Cannot Determined"); the second line shows the number of threads in which only part of emails have labels (other than "Cannot Determined"); the third line shows the total number of emails in these threads. We will use these emails without labels after assigning them the majority label from their corresponding thread. We will use them for training but not for testing. The fourth box shows the number of threads in which all emails were labeled "Cannot Determine". There are only 2 such threads; each has only 2 emails with a total of 4 emails. The fifth box shows the number of bubble emails (described in the previous paragraph) and the threads containing them. The first line shows the total number of bubble emails in the sample; the second line shows the number of threads in which there is at least a single bubble email; the third line shows the total number of emails (including non-bubbles) in these threads. Finally, in the last box, the first line shows the final number of emails we will be using subsequently. This number is after excluding bubble emails and emails in threads in which all emails do not have labels; the last line shows the total number of emails from the previous line having a label (other than "Cannot Determine").

| | Emails | | |
|---|---|---|---|
| Set | Business | Personal | Total |
| Enron$_T$ | 9,127 (86.7%) | 1,401 (13.3%) | 10,528 (+23) |
| Sheffield$_{all}$ | 9,857 (75.7%) | 3,168 (24.3%) | 13,025 |
| Enron$_\cup$ | 16,377 (80.5%) | 3,961 (20.5%) | 20,338 |
| Enron$_{\cap A}$ | 2,506 (88%) | 342 (12%) | 2,848 (88.6%) |
| Enron$_{\cap D}$ | — | | 367 (11.4%) |
| Enron$_\cap$ | — | | 3,215 |

Table 4.6: Annotation result of the Enron Corpus. See subsection 4.2.4 for the description of notations; the extra 23 emails in Enron$_T$ are unlabeled ("cannot determine") but belong to threads where other emails are labeled; we assign them the majority label for the thread they belong.

### 4.2.4 Enron Datasets

In this subsection, we present the Enron datasets we will be using in the rest of this dissertation. We use our annotated emails (subsection 4.2.3) as well as the Sheffield set (subsection 4.2.2) to construct the final datasets. Table 4.6 shows the summary of the Enron datasets with the following notations:

- Enron$_T$: The threads and emails obtained from the AMTurk annotation as described in subsection 4.2.3. Emails in this dataset belong to threads in which each email has a label. Also, there are 23 other emails without labels (labeled “Cannot Determine”). However, these emails belong to threads where other emails have labels (other than “Cannot Determine”). We use these extra 23 emails for training by assigning labels to them using the threads’ majority label.

- Sheffield$_{all}$: All the Sheffield set after deleting duplicates.
- Enron$_{\cap A}$: The intersection between Enron$_T$ and Sheffield$_{all}$ for which both agree in labels.
- Enron$_{\cap D}$: The intersection between Enron$_T$ and Sheffield$_{all}$ for which the two sets disagree in labels.
- Enron$_\cap$: The intersection between Enron$_T$ and Sheffield$_{all}$.
- Enron$_\cup$: Sheffield$_{all}$ $\cup$ (Enron$_T$ $-$ Enron$_\cap$). The union of Sheffield$_{all}$ and Enron$_T$; in case of disagreement in the label, we use Sheffield$_{all}$ labels.

## 4.3 Avocado

Another publicly available email corpus is the Avocado Research Email Collection (Oard *et al.*, 2015), distributed by the Language Data Consortium (LDC) .[2] The corpus consists of emails and attachments are taken from 279 accounts of a defunct information technology company referred to as ”Avocado”. Most of the accounts are those of Avocado employees; the remainder represent shared accounts such as ”Leads”, or system accounts such as ”Conference Room Upper Canada”.

The original collection from LDC is divided into metadata and text files. The metadata files describe folder structure, email characteristics, and contacts. The text files contain the extracted text of the items in the account’s folder.

[2]`https://catalog.ldc.upenn.edu/LDC2015T03`

| | |
|---|---|
| Total number of sampled threads | 2,000 |
| Total number of emails in threads | 5,339 |
| Single-annotated threads | 1,600 |
| Single-annotated emails | 4,274 |
| Single-annotated emails with no label | 55 |
| Double-annotated threads | 400 |
| Double-annotated emails | 1,065 |
| Double-annotated emails with no label | 4 |
| Total number of emails with label | 5280 |
| Total number of emails with no label | 59 |
| Number of threads that all emails have labels | 1,975 |
| Number of emails in these threads | 5,277 |
| Number of threads that some emails have no labels | 25 |
| Number of emails in these threads | 62 |

Table 4.7: Summary of the annotated Avocado corpus

**Constructing the thread structure** The original release of the Avocado email collection does not explicitly have the thread structure for emails. We construct the thread structure for the Avocado corpus using the information in the custodian metadata files as follows:

- We parse the XML metadata files to get the email ids ("id" field) and the ids of the reply-to email ("reply_to" field).
- We construct a directed graph $G_{Avocado}$ such that nodes represent emails, and a direct edge from an email $i$ to an email $j$ indicates that $i$ is a reply to email $j$.
- We extracted all weakly-connected components from the graph $G_{Avocado}$ such that each component represents a thread. Then, we assign a unique number to that thread.
- A weakly-connected component in the graph $G_{Avocado}$ is a tree in which the root represents

the first email in the thread.

Then, we assign a unique id to each weakly connected component in $G_{Avocado}$. We use the graph structure to extract the email location in the thread. We have released the Avocado thread structure with our annotation as MongoDB entries (Alkhereyf and Rambow, 2020).

**Avocado Annotation** For the annotation of the Avocado corpus, we hired two in-house undergraduate students to annotate two overlapping subsets of the Avocado corpus, using the same instructions as we gave the Turkers described in section 4.1. The licensing conditions for this corpus appear to prohibit using AMTurk. In case of disagreement, we arbitrarily choose the first annotator's label for consistency, unless the first is "cannot determine", in which case we choose the second. The Avocado Email Collection has 62,278 threads and 937,958 emails.

We have randomly sampled a total of 2,000 threads and 5,339 emails from the Avocado corpus with different numbers of emails per thread as in Enron.

Each in-house annotator for Avocado labeled 1,200 threads, with 400 threads in common. The first annotator has 3,197 emails, the second has 3,207, and there are 1,065 emails in common (assigned to both annotators). After obtaining the final labels as described in section 4.1, we got a total of 1,976 threads and 5,280 emails labeled as either "Business" or "Personal" from the Avocado corpus.

### 4.3.1 Avocado Datasets

Table 4.8 shows the summary of the Avocado datasets with the following notations:

- Avocado$_1$: The threads and emails labeled by the first annotator, as described in section 4.1.
- Avocado$_2$: The threads and emails labeled by the second annotator as described in section 4.1.

| Set | Business | Personal | Total |
|---|---|---|---|
| Avocado$_1$ | 2,927 (92.1%) | 251 (7.9%) | 3,178 |
| Avocado$_2$ | 2,851 (90.5%) | 298 (9.5%) | 3,149 |
| Avocado$_{\cap A}$ | 948 (93.3%) | 68 (6.7%) | 1,016 (97%) |
| Avocado$_{\cap D}$ | — | — | 31 (3%) |
| Avocado$_\cap$ | — | — | 1,047 |
| Avocado$_\cup$ | 4,810 (91.1%) | 470 (8.9%) | 5,280 |
| Avocado$_T$ | 4,810 (91.1%) | 467 (8.9%) | 5,277 |

Table 4.8: Annotation result of the Avocado corpus. See subsection 4.3.1 for the description of notations.

- Avocado$_{\cap A}$: The intersection between Avocado$_1$ and Avocado$_2$ in which both agree in labels.
- Avocado$_{\cap D}$: The intersection between Avocado$_1$ and Avocado$_2$ in which they disagree in labels.
- Avocado$_\cap$: The intersection between Avocado$_1$ and Avocado$_2$.
- Avocado$_\cup$: All the threads and emails labeled as described in section 4.1: Avocado$_1$ $\cup$ (Avocado$_2$ $-$ Avocado$_\cap$). In case of disagreement, we choose the label from the first annotator.
- Avocado$_T$: A subset of Avocado$_\cup$ such that all emails belong to complete threads (i.e., all emails in the thread have labels).

## 4.4 Inter-Annotator Agreement

In this section, to measure the reliability of the annotation email datasets; we analyze inter-annotator agreement (IAA) of the annotation task of emails into business and personal categories on both

Enron and Avocado corpora. We use different measures for IAA as described in Artstein and Poesio (2008). For inter-annotator agreement analysis, we use the Avocado$_\cap$ dataset, which is a sub-set of Avocado$_T$, and Enron$_T$ (described in section 4.1). Emails in Enron$_T$ are labeled by 3 different annotators using AMTurk (some emails has 5 annotators), while Avocado$_\cap$ is labeled by two in-house annotators. Note that for Enron, the set of annotators is not the same among different emails.

First, we define the amount of observed agreement $\text{agr}_i$ on item (i.e., email) $i$ as the proportion of the agreeing annotation pairs out of the total number of annotation pairs for that email:

$$\text{agr}_i = \sum_{k \in K} \frac{\binom{n_{ik}}{2}}{\binom{c_i}{2}} = \sum_{k \in K} \frac{n_{ik}(n_{ik}-1)}{c_i(c_i-1)}$$

Then, the overall observed agreement is the mean:

$$A_o = \frac{1}{N} \sum_{i \in I} \text{agr}_i$$

Where $N$ is the total number of items (i.e., emails), $K$ is the set of categories, $n_{ik}$ is the number of coders (annotators) who assigned item $i$ to category $k$, and $c_i$ is the number of coders (annotators) who annotate item $i$.

$A_o$ alone is not sufficient for calculating IAA since it does not consider the distribution of items (i.e., emails in our case) among categories. As in our case, we expect a higher percentage for agreement since that the business category is much more frequent than personal. We use $\pi$ and $\kappa$ measures to overcome this issue. $\pi$ uses the prior distribution of the categories, while $\kappa$ takes into account the distribution for individual annotators. We use Fleiss's Multi-$\pi$ (a generalization of Scott's $\pi$) on both Enron and Avocado, and for Avocado, we only use Choen's $\kappa$ since the set of annotators in Enron is not the same among emails. We define $A_e$ as the expected agreement by chance. Both $\pi$ and $\kappa$ have the exact definition for $A_o$, but different definitions for $A_e$.

$$\pi, \kappa = \frac{A_o - A_e}{1 - A_e}$$

For multi-$\pi$, since that the number of annotators per email is not the same, as some emails in Enron$_T$ are labeled by more than 3 annotators; we modify the definition of $A_e^\pi$ described in Artstein and Poesio (2008) to be:

$$A_e^\pi = \sum_{k \in K} (\hat{p}(k))^2 = \frac{1}{N^2} \sum_{k \in K} \Big( \sum_{i \in I} \frac{n_{i,k}}{c_i} \Big)^2$$

Where $\hat{p}(k)$ is the probability of category $k$ or the observed proportion of items (i.e., emails) assigned to category $k$ by all annotators, the expected pairwise agreement $\hat{p}(k)^2$ is the joint probability that two arbitrary coders will assign an item (email) to the category $k \in K$, $N$ is the total number of emails, and $n_{i,k}$ denotes the frequency of labels of the category $k$ in the email $i$. Our modification allows us to deal with the issue of having different numbers of annotators among emails. For $A_e^\kappa$ and then $\kappa$, we use the same definitions described in Artstein and Poesio (2008).

| | | Normalized | Non-Normalized |
|---|---|---|---|
| Avocado | Keeping "Cannot Determine" | $A_o = 0.958$<br>$\pi = 0.741$<br>$\kappa = 0.742$ | $A_o = 0.889$<br>$\pi = 0.582$<br>$\kappa = 0.584$ |
| | Excluding "Cannot Determine" | $A_o = 0.97$<br>$\pi = 0.798$<br>$\kappa = 0.799$ | $A_o = 0.901$<br>$\pi = 0.602$<br>$\kappa = 0.604$ |
| Enron | Keeping "Cannot Determine" | $A_o = 0.806$<br>$\pi = 0.418$ | $A_o = 0.642$<br>$\pi = 0.270$ |
| | Excluding "Cannot Determine" | $A_o = 0.844$<br>$\pi = 0.465$ | $A_o = 0.668$<br>$\pi = 0.279$ |

Table 4.9: IAA scores for Enron$_T$ and Avocado$_\cap$. Normalized means the result of IAA on labels after grouping "Business", and "Somehow Bussiness" into one group, "Business"; "Personal", "Somehow Personal" and "Mixed" into one group: "Personal". "Non-normalized" means we keep categories as they are.

Table 4.9 shows IAA measures for Enron$_T$ and Avocado$_\cap$ in different settings:

- *Keeping "cannot determine":* we keep the "cannot determine" labels when calculating $\text{agr}_i$ for each email.
- *Excluding "cannot determine":* means we discard all "cannot determine" labels when we calculate $\text{agr}_i$; if there are less than two remaining labels for the email $i$, we discard this email from the calculation of $A_o$.
- *Normalized* means we group fine-grained categories as described in section 4.1; we group "Business" and "Somehow Business" into one category "Business"; and "Personal", "Somehow Personal", and "Mixed" into one category "Personal".
- *Non-normalized:* means we keep the categories as they are without the normalization step described above.

The example of a thread in Table 4.1 and the numbers in Table 4.9 show that human classification of emails into business and personal categories is a non-trivial task as there are some cases in which all annotators (even trained ones as in the case of Avocado) disagree on a label. The $\pi$ and $\kappa$ values for Avocado are very similar, which suggests that the observed distributions of the two in-house annotators are almost identical.

Sappelli *et al.* (2016) annotated emails from Enron and Avocado with different e-mail intent and task dimensions, such as the number of tasks for the recipient stated in the email and the implicit reason for the email. As in our work, they use Amazon Mechanical Turk to annotate the Enron dataset and two in-house trained annotators for Avocado. They use Cohen's $\kappa$ to measure the inter-annotator agreement. Similar to our results in Table 4.9, they report a lower inter-annotator agreement in Enron than in Avocado. They state that the high agreement on the Avocado set suggests that trained annotators reach a higher agreement than non-trained annotators. They also provide another explanation is that the messages in the Avocado set are easier to categorize. These explanations can be applied to our work too.

Based on the strength of agreement according to Landis and Koch (1977)), where values between $0.0 - 0.2$ can be seen as slight, $0.2 - 0.4$ as fair, $0.4 - 0.6$ as moderate, $0.6 - 0.8$ as substantial, and $> 0.8$ as almost perfect agreement. The inter-annotator scores of the non-normalized labels for Enron are fair, while all other scores are moderate or better. In our experiments in Part II, we will use only the normalized data since we are interested in binary classification of emails into "business" and "personal".

# Chapter 5

# Graph Representations

Graphs are ubiquitous and occur naturally in various real-world applications, including social networks, and word co-occurrence networks. In this thesis, we are interested in incorporating information from the social network induced from the communication network for text classification. Graphs are natural representations for social networks of different types. However, the choice of a representation of documents and users as a graph structure is a crucial step before applying machine learning models. Different graph structures carry different information about how documents and users are related.

In this chapter, we present various graph structures to represent the underlying communication network of documents and users. We will use these graph structures throughout this thesis. We first review background information about graph terminologies in section 5.1. Then, we discuss different graph structures that we use throughout this thesis.

## 5.1 Background

In this section, we define some graph concepts and terms that we frequently use throughout this dissertation. We use definitions and notations from different resources (Goyal and Ferrara, 2018;

Cai *et al.*, 2018; Liben-Nowell and Kleinberg, 2007; Cui *et al.*, 2018).

### 5.1.1 Graphs

A graph $G = (V, E)$ is a set of nodes (or vertices) $V$ with a set of edges $E$ that link nodes $E \subseteq \{(u, v) \mid u, v \in V\}$. The edges can be directed or undirected and then the graph. For undirected graphs, a pair of nodes is unordered (symmetric), while it is ordered for directed graphs (asymmetric). Edges might have an associated numerical value called a *weight* representing information such as the capacity of the link between the two ends. Many real-world applications, such as social networks, can be modeled as graphs.

### 5.1.2 Adjacency Matrix

The adjacency matrix $A$ of graph $G$ is a square matrix $n \times n$, where $n$ is the number of nodes in $G$. $A_{i,j}$ indicates whether or not the nodes $i$ and $j$ are adjacent (linked by an edge) in the graph $G$ by using binary values (1 if connected; 0 otherwise) or the weight of the edge between $i$ and $j$. If the graph is undirected, the adjacency matrix $A$ is symmetric.

### 5.1.3 Bipartite Graphs:

A bipartite graph is a graph in which nodes can be divided into two disjoint sets $U$ and $V$ such that no two nodes within the same set are connected, and every edge connects a node in $U$ to a node in $V$.

### 5.1.4 Homogeneous and Heterogeneous Networks:

Networks with a single type of nodes are referred to as *homogeneous* or 1-mode networks. An example of homogeneous networks is the friends' network on Facebook, where nodes represent people and edges linking two nodes indicate that the two ends are connected. Networks with multiple types of nodes are referred to as *heterogeneous* or multi-mode networks. The network of OpenTable (online restaurant-reservation service) is an example of heterogeneous (2-mode) networks with two categories of nodes: restaurants and diners. Edges link a diner $d$ with a restaurant $r$ if $d$ has a reservation at $r$.

### 5.1.5 Jaccard's Coefficient:

the Jaccard coefficient measures similarity between two sets of elements. For graphs, the Jaccard coefficient of nodes $u$ and $v$ is the proportion of shared neighboring nodes between $u$ and $v$ relative to the total number of nodes connected to $u$ or $v$; it is defined as:

$$J(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

where $\Gamma(u)$ denotes the set of neighbors of $u$.

### 5.1.6 Centrality Measures:

Centrality measures identify the most important nodes within a graph. There are various centrality measures that have been proposed over the years.

#### 5.1.6.1 In-degree, Out-degree, Total-degree Centrality:

In-degree for a node is defined as the number of incoming edges, while out-degree is the number of outgoing edges. Both in-degree and out-degree are defined for directed graphs. The total degree (or simply degree) of a node is the number of all edges. It is defined for both directed and undirected graphs. In-degree, out-degree, and total-degree centralities are the corresponding degree value normalized by dividing by the maximum possible degree $n - 1$, where $n$ is the number of nodes in the graph.

#### 5.1.6.2 Eigenvector Centrality:

Eigenvector centrality of node $v$ is $x_v$ where: $x$ is the eigenvector associated with the largest eigenvalue of the adjacency matrix $A$. A large eigenvector score means that a node is connected to many nodes which themselves have high scores.

#### 5.1.6.3 Betweenness Centrality:

Betweenness is a measure of centrality in a graph based on shortest paths. Betweenness centrality of a node $v$ is the number of shortest paths that pass through $v$ normalized by all possible shortest paths between all pair of nodes:

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

where $V$ is the set of nodes, $\sigma(s,t)$ is the number of shortest paths between $s$ and $t$, and $\sigma(s,t|v)$ is the number of those paths passing through some node $v$ other than $s$, and $t$.

#### 5.1.6.4 Closeness Centrality:

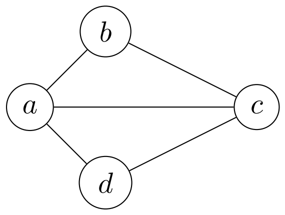
measures how close a node $v$ to other nodes in a graph $G$, it is calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph:

$$c_C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)}$$

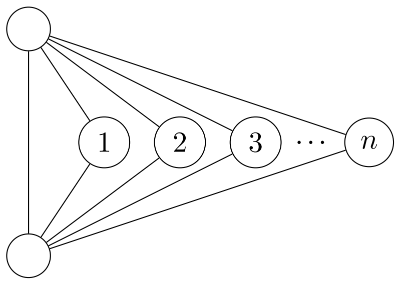
where $d(v,u)$ denotes the length of the shortest path between $u$ and $v$.

#### 5.1.6.5 HITS:

Hyperlink-Induced Topic Search (HITS) algorithm (Kleinberg, 1999), also known as hubs and authorities, was mainly developed to rank web pages relevant for a particular topic. It assigns two scores for nodes in a directed graph $G$: *hub score* and *authority score*. A high hub score for a node indicates that the node points to many other nodes, and a high authority score indicates that the node is linked by many different nodes (with high hubs).



(a) Clustering coefficient of nodes: $c_a = c_c = 2/3, c_b = c_d = 1; C(G) = 1/4(2 + 4/3) \approx 0.83$
Transitivity of the graph: $T(G) = 3 \times 2/8 = 0.75$



(b) $T(G) \to 0, C(G) \to 1$ as $n \to \infty$

Figure 5.1: Illustration of local clustering coefficient and transitivity (global clustering coefficient). The examples are adapted from Safro (2014).

### 5.1.7 Clustering Coefficient and Transitivity

The clustering coefficient is a measure of the tendency of nodes to cluster together. There are two versions of this measure: global and local. The global version is usually known as *transitivity*. Clustering coefficient and transitivity are based on *triads* of nodes. A triad is three nodes that are connected by either two (open triad) or three (closed triad or triangle) edges.

The clustering coefficient for a graph is the average clustering coefficient for all nodes:

$$C(G) = \frac{1}{n} \sum_{v \in G} c_v$$

Where $n$ is the number of nodes in $G$ and $c_v$ is the clustering coefficient of node $v$, which is the fraction of possible triangles through $v$:

$$c_v = \frac{\lambda(v)}{\tau(v)} = \frac{2\lambda(v)}{deg(v)(deg(v) - 1)}$$

where $\lambda(u)$ is the number of triangles through node $v$ or the number of links between neighbors of $v$, $\tau(u)$ is the number of triads at $v$ and $deg(v)$ is the degree of $v$.

The transitivity (or global clustering coefficient) for a given graph $G$ is based on the relative number of triangles in the graph, compared to the total number of triads; it is defined as:

$$T(G) = \frac{3\lambda(G)}{\tau(G)} = \frac{3 \text{ number of triangles}}{\text{number of all triads}}$$

Figure 5.1 illustrates the two measures.

#### 5.1.7.1 Modularity

Modularity measures the strength of the partition of a network into modules (clusters or groups). Networks with high modularity scores have dense connections between the nodes within the same cluster and sparse connections between nodes in different clusters (Clauset *et al.*, 2004). The modularity score for a given partition on a graph $G$ is defined in Newman (2011) as:

$$Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

where $m$ is the number of edges in $G$, $A$ is the adjacency matrix of $G$, $k_i$ is the degree of node $i$, and $\delta(c_i, c_j)$ is 1 if nodes $i$ and $j$ are in the same community (cluster) and 0 otherwise.

## 5.2 Bipartite User-Document Representation

One natural (and comprehensive) representation for documents and users is a bipartite graph. In this graph, we represent users and documents in disjoint sets and link a user $i$ to a document $j$ if the user $i$ is a participant in the document $j$. This representation of users and documents is generic and can be applied to different applications and genres. For instance, in email, we create a link for the sender of the email to the corresponding email, then a link from the email to the corresponding recipients; we refer to this structure as the *bipartite user-document network*. Figure 5.2 illustrates this graph.

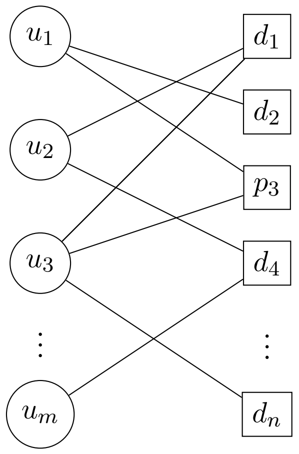
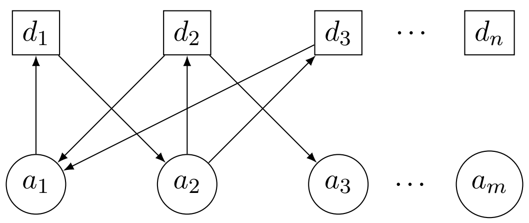
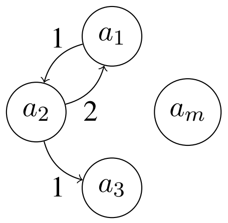


Figure 5.2: A bipartite graph for $m$ Documents and $n$ Users.



(a) Bipartite document-user graph



(b) User graph

Figure 5.3: A user graph induced from a bipartite graph of documents and users using *one-mode projection* on the user nodes. The edge weight reflects the number of documents sent from the source to the target.

## 5.3 User Graph

Another representation for the communication network is a graph (not bipartite) whose nodes represent people (e.g., email addresses) and whose edges represent document communication such that an edge exists if there is at least one document has been exchanged between the two end nodes; we refer to this structure as the *user network*. This graph is a result of a one-mode projection of the user nodes in the bipartite graph. Figure 5.3 illustrates these two types of graphs.

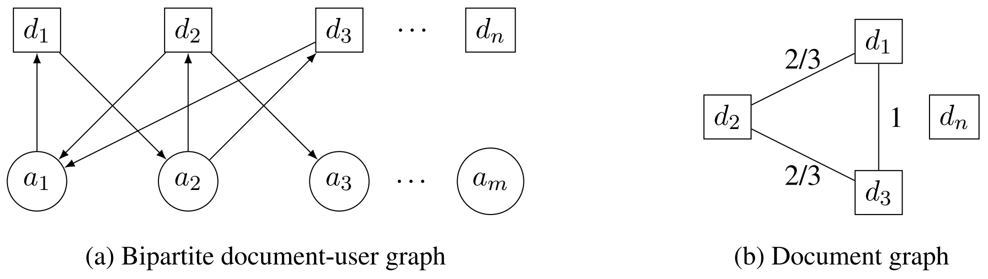


Figure 5.4: A document graph induced from a bipartite graph of documents and users using *one-mode projection* on the document nodes. The edge weight reflects the fraction of common participants over the total number of participants (union).

## 5.4 Document Graph

Another graph that can be induced from the bipartite graph (section 5.2) is a graph for documents rather than users. This graph has only documents as nodes. Documents are linked if at least they share a single participant. This graph is simply a one-mode projection of the user nodes in the bipartite graph. Figure 5.4b illustrates a document graph induced from a bipartite document-user graph by applying a one-mode projection on the document nodes.

## 5.5 Graph Directionality and Other Properties

Graph directionality – whether the graph is directed or undirected – can carry different types of information. For different tasks we present in this thesis, we construct both directed and undirected graphs for both the user graph and the bipartite user-document graph.

In directed graphs, each edge has a source and destination node, which explicitly shows document's directionality (e.g., sender and recipients). In contrast, in undirected graphs, the directionality of the communication is not reflected within the edges.

Another graph property we take into account is the edge weight. For the user graph, the edge weight reflects the number of documents that have been exchanged between the two ends. We make use of both the graph directionality and the edge weights in our work.

In the directed bipartite network, the weights are always 1 in the bipartite graph. For the directed user network, edge directions indicate that the source user has sent documents (e.g., emails) to the target user, and the edge weight reflects the number of documents that have been sent from the source to the target, while in the undirected document network, edges indicate that the two connected nodes (i.e., users) have exchanged emails regardless of who sent the document.

**Threshold values** For some applications, the projected document and/or user graphs from the corresponding bipartite graph might be very dense and adding noise. For instance, suppose that we have an email exchange network represented as a bipartite graph, and there are many users who have exchanged a few emails or even a single email. An example of these emails is a public email announcement that is sent to a large number of people. Projecting this graph into a user graph where all users are linked if they have exchanged even a single email might add noise. Therefore, it is useful to set a certain threshold $t$ such that users are linked only if they have exchanged or participated in at least $t$ number of documents. Similarly, for the document graph, we can also set a similar threshold such that documents are linked only if they share a certain number (or fraction) of participants.

# Chapter 6

# Social Network Analysis of the Enron Corpus

In this chapter, we present various social network analyses performed on the Enron datasets described in chapter 4. We perform different social network analyses on various graphs constructed using the datasets discussed in subsection 4.2.4. In particular, we use the $\text{Enron}_{\cup}$ dataset to conduct the social network analysis. We choose $\text{Enron}_{\cup}$ because it is the largest dataset, which allows us to include as many labeled emails as possible to analyze the induced social networks. Our goal in this chapter is to investigate if there is a signal in the underlying social network graphs constructed from emails that can distinguish between personal and business emails.

In section 6.1, we introduce three sub-networks induced from the labeled emails in $\text{Enron}_{\cup}$; we analyze different properties and measures on the personal and business sub-networks. Then, in section 6.2, we apply clustering algorithms to divide the Enron core user graph into clusters, study the distribution of personal and business emails exchanged within and between clusters. In section 6.3, we study signed social networks induced from the labeled examples. We conclude the chapter in section 6.4

## 6.1 Personal and Business Sub-networks

In this section, we analyze different graphs induced from the datasets in subsection 4.2.4. In particular, we analyze the differences between three user graphs (see user graph in section 5.3); each graph is constructed using different labels from the datasets. We create three graphs using labeled emails in Enron$_\cup$: all graph, business graph, and personal graph. We refer to these three graphs as: $G_{\text{all}}$, $G_{\text{bus}}$, and $G_{\text{pers}}$, respectively. Nodes in these graphs represent users, and edges indicate that the two ends have exchanged emails.

To construct a graph, for each email in Enron$_\cup$, we create undirected weighted edges between the sender and all recipients where weights reflect the number of emails exchanged between the two nodes (i.e., users). Figure 5.3b (page 57) illustrates this graph.

Similarly, we construct the sub-networks of the personal and business emails using only personal or business emails, respectively. In addition to edge weights, we also compute the normalized weights by adding $\frac{1}{rec(e)}$ for each email (instead of 1) to edges connecting the sender and every recipient. $rec(e)$ denotes the number of recipients in email $e$. In other words, in the non-normalized weights, for each email, we extract all pairs of sender and recipients; then, we add 1 to their corresponding edge weight. In the normalized weights, instead of adding 1 to the weights of each edge representing pairs of sender and recipients for a given email, we add the fraction $\frac{1}{rec(e)}$ such that the total weight for a given email is 1.

We study these three graphs (i.e., $G_{\text{all}}$, $G_{\text{bus}}$, and $G_{\text{pers}}$) using various social network analysis measures to investigate whether the email communication graphs constructed from the labeled emails have different characteristics. In addition to these three graphs, we construct three random graphs by keeping the same number of nodes and edges in the corresponding graph, but we rewire edges randomly. We do so to study if there are some properties associated with the size of the graph or with the distribution of edge types (i.e., personal vs. business). Note that there are 3,317 nodes (users) that appear in all of the three graphs; these nodes represent unique users who exchange both business and personal emails.

| | Business Only Edges | Personal Only Edges | Mixed Edges |
|---|---|---|---|
| Number of edges | 65,044 (83.33%) | 10,204 (13.07%) | 2,807 (3.6%) |
| Total weights | 125,833 (74.71%) | 14,665 (8.71%) | 27,928 (16.58%) |
| Business weights | 125,833 (85.04%) | 0 | 22,141 (14.96%) |
| Personal weights | 0 | 14,665 (71.70%) | 5,787 (28.30%) |
| Average edge weight | 1.935 | 1.437 | 9.949 |
| Normalized total weights | 12,968.91 (63.77%) | 2,414.1 (11.87%) | 4,954.99 (24.36%) |
| Normalized business weights | 12,968.91 (79.19%) | 0 | 3,408.09 (20.81%) |
| Normalized personal weights | 0 | 2,414.1 (60.95%) | 1,546.9 (39.05%) |

Table 6.1: Distribution of edges in the user graph $G_{\text{all}}$ constructed from Enron$_{\cup}$. Normalized business and personal weights indicate weights normalized by the number of recipients for each email.

### 6.1.1 Edge Distribution

In this subsection, we discuss the distribution of edge types in $G_{\text{all}}$. Table 6.1 shows statistics on three types of edges in $G_{\text{all}}$. "Only business" and "only personal" denote edges, where all emails exchanged between the two ends of the edge are only business emails or only personal emails, respectively. "Mixed edges" indicates that the two ends for a given edge have exchanged both business and personal emails. The non-normalized weights are computed such that we add 1 to the corresponding edge weight for the pair of sender and recipient in every email in the dataset. The normalized numbers are computed such that we add $\frac{1}{rec(e)}$ to the corresponding edges; where $rec(e)$ indicates the number of recipients in a given email. Note that the sum of the normalized weights is equal to the number of emails in the Enron$_{\cup}$.

The first line in Table 6.1 shows the number of different types of edges (regardless of the weights) in the graph $G_{\text{all}}$.

The second to the fourth lines show the total number of business, and personal weights, respectively, given the edge type. Note that these are non-normalized weights, and the same email might be counted more than once if the email has more than one recipient. The fifth line, “average edge weight”, is the total number of weights for a given edge type divided by the number of edges for that type. The numbers in this line are computed by dividing the numbers in the second line by the number in the first line. The last three lines in Table 6.1 show the normalized total, business, and personal weights, respectively.

The numbers in Table 6.1 show that most of the edges are business only; a few edges are personal only; fewer edges are mixed. This means that most of the unique sender and recipient pairs exchange only business emails, fewer pairs exchange only personal emails, and only a small percentage of pairs exchange both types of emails. However, this small sub-network of pairs contributes to a large amount of email exchanged in Enron$_{\cup}$ as indicated by the total weight of mixed edges. This is due to the fact that we have the complete email correspondence only for the core group of Enron. For the people who are not in the core group, we have almost no email correspondence among them: the only emails between people in this group are emails involving at least one person in the core group as a joint recipient.

### 6.1.2 Email Distribution

In this subsection, we analyze how business and personal emails are exchanged through different edge types. Table 6.2 shows the number of business and personal emails sent through different types of edges; for each email, we study the type of all edges (an edge for each pair of sender and recipient) that the email sent through. If there are some recipients linked with business only edges,

| | All Business Edges | All Personal Edges | Mixed Edges |
|---|---|---|---|
| Business Emails | 10662 (65.1%) | 0 | 5715 (34.9%) |
| Personal Emails | 0 | 1959 (49.46%) | 2002 (50.54%) |

Table 6.2: Number of business and personal emails sent through different types of edges in $G_{\text{all}}$. The edge types "all business" and 'all personal" indicate that all emails exchanged between the two users are only business or only personal, respectively. The "mixed" edge type means that the two ends (users) have exchanged both business and personal emails.

and others linked with personal only edges; we consider that the email is sent through mixed edges.

The numbers show that around two thirds of the business emails (65.1%) are exchanged through business only edges, and half of the personal emails are exchanged through personal only edges.

| | Actual Graphs | | | Randomly Rewired Edges | | |
|---|---|---|---|---|---|---|
| Measure | $G_{\text{all}}$ | $G_{\text{bus}}$ | $G_{\text{pers}}$ | $G_{\text{all}}$ | $G_{\text{bus}}$ | $G_{\text{pers}}$ |
| # Nodes | 24,995 | 18,710 | 9,602 | 24,995 | 18,710 | 9,602 |
| # Edges | 78,055 | 67,851 | 13,011 | 78,055 | 67,851 | 13,011 |
| Density | 0.00024 | 0.00039 | 0.00027 | 0.00024 | 0.00039 | 0.00027 |
| Average Degree | 6.2456 | 7.2404 | 2.7101 | 6.2456 | 7.2404 | 2.7101 |
| # Triangles | 186,129 | 174,310 | 3,898 | 64 | 68 | 4 |
| # Triads | 8,515,567 | 7,766,555 | 414,620 | 488,314 | 493,438 | 35,368 |
| Avg. Clustering | 0.1911 | 0.2232 | 0.0671 | 0.00045 | 0.000424 | 0.00011 |
| Transitivity | 0.0655 | 0.0658 | 0.0183 | 0.00039 | 0.00041 | 0.00034 |

Table 6.3: SNA measures of graphs: $G_{\text{all}}$, $G_{\text{bus}}$, and $G_{\text{pers}}$ and their corresponding graphs with randomly rewired edges.

### 6.1.3 Sub-network SNA Measures

In this subsection, we show different SNA measures for the three graphs: $G_{\text{all}}$, $G_{\text{bus}}$, and $G_{\text{pers}}$. In addition, we construct three other graphs where edges are randomly rewired by keeping the same number of nodes and edges in the corresponding graph while randomly rewiring edges. We do so because we are interested in analyzing how some SNA measures differ when we randomly rewire edges in each graph. Particularly, we are interested in analyzing the tendency for nodes to form clusters in the three graphs. In other words, we are interested in analyzing whether people tend to form clusters with others who exchange business emails differently than with whom they exchange personal emails.

Table 6.3 shows different SNA measures for these three graphs. The first box shows the number of nodes, the number of edges, the graph density, and the average degree. Note that rewiring edges does not change these numbers in the corresponding graph. The numbers show that the business sub-network is denser than the personal one, which means that business emails are exchanged within larger groups. In fact, the average number of recipients for business emails in $\text{Enron}_{\cup}$ is 9.03, and for personal email is 5.16.

The second box in Table 6.3 shows the number of triangles, the number of triads, the average clustering, and transitivity. A triangle in a graph $G$ is a set of three nodes that are mutually adjacent in $G$. A triad is a set of three nodes that are connected by either two (open triad) or three (closed triad or triangle) edges. Average clustering is the average clustering coefficient for all nodes in the given graph, and it measures the tendency of nodes to cluster together (local level). Transitivity measures the global clustering coefficient of the graph, and it is based on the relative number of triangles in the graph, compared to the total number of triads. For more information about these metrics, see subsection 5.1.7 on page 55.

The numbers show that the actual graphs have much higher numbers in all measures than their corresponding randomly rewired graphs. This is expected since that the actual graphs represent real social networks. Graphs representing social networks tend to have larger numbers of triangles, transitivity, and cluster coefficient than other graphs. We observe that the business graph $G_{\text{bus}}$ has a higher average clustering coefficient and transitivity score than the personal graph $G_{\text{pers}}$. This indicates that people exchanging business emails tend to form clusters more often than those exchanging personal emails. In other words, for two nodes who share a common neighbor with whom they both exchange business emails, the chance that they will exchange a business email with each other is much higher than in the corresponding situation for personal emails. Particularly, the chance that a pair of people in the business graph that share a common neighbor is around 3.5 times higher than for a pair in the personal graph.

The analysis of these graphs ($G_{\text{all}}$, $G_{\text{bus}}$, and $G_{\text{pers}}$) shows that the three graphs have different properties. That motivates using information extracted from graphs to improve the classification of emails into business and personal.

## 6.2 Clusters

In this section, we analyze the emails exchanged within and between clusters of users. We cluster nodes in a user graph induced from the Enron$_{\cup}$ and study if there is a signal in clusters that can distinguish personal and business emails.

The users in the whole email exchange communications in the original release by Agarwal *et al.* (2012) are divided into two sets: **core** and **non-core** (described in subsection 4.2.3). Since we have only the complete correspondence for the core network, we will use the core network in this section to study clusters. The Enron core network consists of 158 people, but their corresponding user graph consists of 143 nodes. The difference in the numbers is due to the fact that in the Columbia release

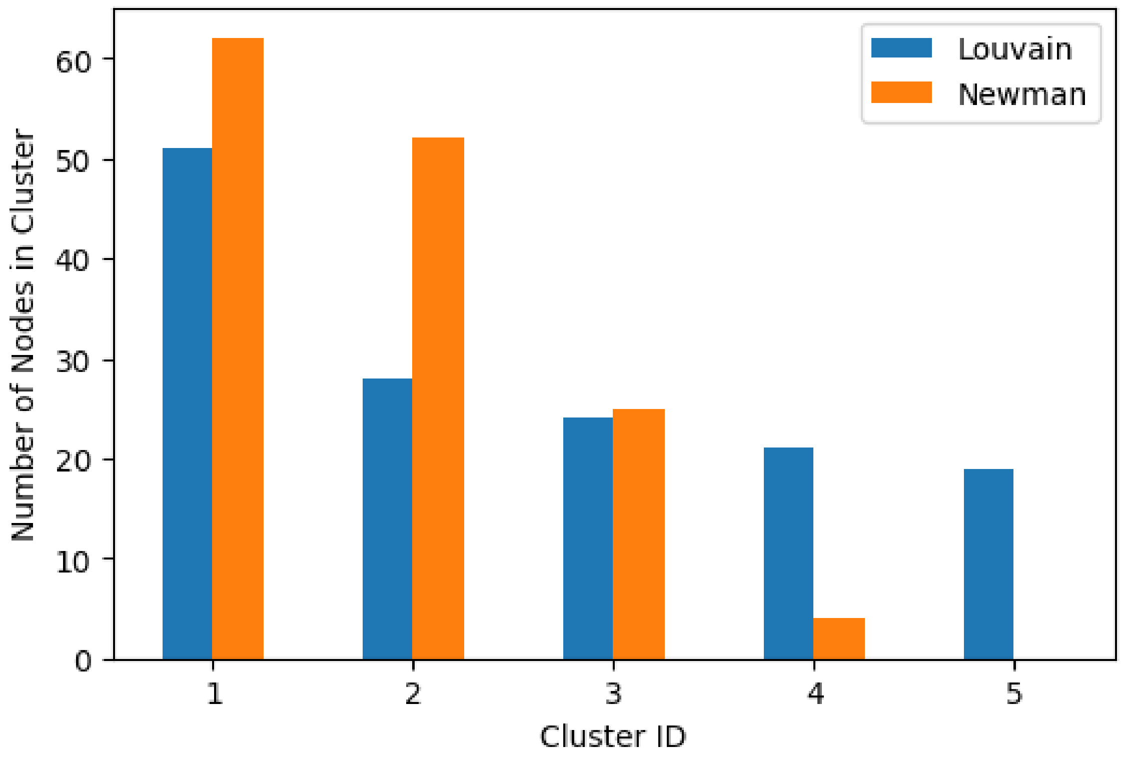


Figure 6.1: Number of nodes in each cluster detected using two clustering algorithms: Louvain and Newman. Note that the two algorithms detect different numbers of clusters: 5 and 4, respectively.

(Agarwal *et al.*, 2012), some mailboxes representing different people are merged into one user id (UID) in the MongoDB (see Appendix B).

First, we construct the graph of the core network using only emails in Enron$_\cup$ in which the sender and all recipients are in the core network. There are 4,458 such emails in Enron$_\cup$. Then, we group users into clusters using two clustering algorithms: Louvain (Blondel *et al.*, 2008) and Newman (Clauset *et al.*, 2004) algorithms. Both algorithms detect communities in graphs via maximizing the modularity score (described in subsubsection 5.1.7.1). Louvain is a greedy algorithm, while Newman performs agglomerative hierarchical clustering. Note that graph clustering by maximizing the modularity score is proven to be an NP-complete problem (Brandes *et al.*, 2006, 2007); therefore, the clusters obtained using these algorithms are sub-optimal. We use the Organization

Risk Analyzer (ORA) toolkit (Carley *et al.*, 2008) to detect sub-groups in the Enron core network using the algorithms mentioned above. ORA automatically picks the number of clusters that yields the largest decrease in the modularity score. We have also tried to use other clustering algorithms, such as Grivan-Newman, but that led to many clusters consisting of a single user only. The two algorithms, i.e., Louvain and Newman, found two different numbers for clusters. Particularly, the number of clusters for the Louvain algorithm is 5, while for the Newman algorithm is 4. Figure 6.1 shows the number of nodes in each cluster for each clustering algorithm. We observe that the Louvain algorithm detects more clusters with a smaller variation in the number of nodes in each cluster than the Newman algorithm.

After clustering Enron's core network, we analyze the distribution of business and personal emails in these clusters by counting the number of emails for which the sender and all recipients belong to the same or different clusters. In case that there is an email where one or more recipients are in a different cluster, we consider this email to be exchanged between different clusters.

| Clustering Method | Louvain | Newman |
|---|---|---|
| # All emails in same cluster | 3,763 | 3,507 |
| # Business emails in same cluster | 3,093 (82.2%) | 2,872 (81.89%) |
| # Personal emails in same cluster | 670 (17.8%) | 635 (18.11%) |
| # All emails in different clusters | 695 | 951 |
| # Business emails in different clusters | 648 (93.24%) | 869 (91.38%) |
| # Personal emails in different clusters | 47 (6.8%) | 82 (8.62%) |

Table 6.4: Distribution of business and personal emails exchanged within and between clusters in the core network of Enron. We use two clustering algorithms: Louvian and Newman.

Table 6.4 shows the distribution of business and personal emails among clusters. The first set of numbers show the number of emails exchanged within the same cluster for each email type; the second set of numbers show those numbers for emails exchanged between different clusters.

We observe that most of the emails are exchanged within the same clusters. This is expected as the clustering algorithms try to detect clusters by maximizing the modularity score such that nodes within the same cluster have dense connections and sparse connections between nodes in different clusters. Therefore, we expect that we have fewer edges between nodes in different clusters, and then, less number of emails exchanged between these nodes than between those within the same cluster. Also, using both clustering algorithms, we observe that the conditional distributions of personal and business emails are different given that the email is exchanged within the same or different clusters. Particularly, the percentage of personal emails is much less when the participants are in different clusters than when they are within the same cluster.

| | Bus-Bus-Bus | Bus-Bus-Pers | Bus-Pers-Pers | Pers-Pers-Pers |
|---|---|---|---|---|
| $G_{\text{all}}$ | 142,243 (76.42%) | 27,257 (14.64%) | 12,731 (6.84%) | 3,898 (2.09%) |
| Permuted $G_{\text{all}}$ | 107,982 (58.01%) | 64,504 (34.66%) | 12,800 (6.88%) | 843 (0.45%) |

Table 6.5: Number of triads for each of the four possible triads in $G_{\text{all}}$ and permuted $G_{\text{all}}$ where signs are randomly shuffled (the graph structure is preserved). Note that the “mixed” edge type are merged with the ”Pers” edge type.

## 6.3 Signed Networks

In this section, we analyze signed networks induced from the labeled emails in Enron$_{\cup}$. A signed social network is a graph in which each edge has a positive or negative sign. Signs allow edges between individuals to carry information about the nature of the relationship. For instance, individuals in a dyad (an edge linking a pair of nodes) may be friends or foes; they may be in a formal or informal relationship. Then, the sign of the edge between these individuals indicates the type of their relationship. Several theories characterize signed social networks: in structural balance theory, edge signs indicate friendship and enmity, with some triads of signed edges being stable, and others being unstable (Cartwright and Harary, 1956). Particularly, a triad with an odd number of positive signs ($+--$ or $+++$) is considered stable, while a triad with an even number of positive signs ($++-$ or $---$) is considered unstable.

Although the concept of signed social networks is mainly for positive (friendly) and negative (antagonistic) binary relationships, which does not apply to business vs. personal relationship, we try in this section to investigate whether there is a signal of social balance in our data sets such that there are some types of triads preferred more than other types in the data sets.

As discussed in section 6.1, most of the edges in $G_{\text{all}}$ are either business only or personal only, which means that most of the pairs of users in Enron$_{\cup}$ exchange only business or only personal emails; and a tiny number of pairs exchange both business and personal emails.

We use the induced graph $G_{\text{all}}$ of the labeled emails of Enron$_{\cup}$ (the same graph in section 6.1). In order to obtain binary categories for edge types, we combine personal and mixed edges into "Pers" type, and we use "Bus" to denote business type edges. We also construct a similar graph but with signs randomly shuffled (i.e., permuted) in which we keep the same structure of the graph (i.e., edges are connected to the same node), but we randomly shuffle signs (the same distribution of the signs is maintained), we refer to this randomly shuffled graph as "Permuted $G_{\text{all}}$". We construct the Permuted $G_{\text{all}}$ graph to analyze which signed triads are preferred in the actual graph $G_{\text{all}}$ by comparing the distribution of signed triads to the randomly shuffled graph Permuted $G_{\text{all}}$.

Table 6.5 shows the number of all possible triads in $G_{\text{all}}$ and Permuted $G_{\text{all}}$. The numbers show that there is a signal of structural balance in the signed graph induced from the labeled network of email exchange. Note that the notion of structural balance in this context is different than in signed networks in general. Particularly, triads with homogeneous signs are preferred in the actual network over the sign permuted one. More specifically, the numbers of BBB (all business) and

PPP (all personal) triads drop in Permuted $G_{\text{all}}$; heterogeneous triads are dispreferred in the actual graph $G_{\text{all}}$ than in the permuted one. Particularly number of BBP (two individuals having a personal relationship and a mutual business friend) increases when we shuffle the signs. Less dispreferred is BPP; when a pair of users have a business relationship and a mutual personal relationship with a third person, the number of such triads almost does not change in Permuted $G_{\text{all}}$ in comparison to $G_{\text{all}}$.

Since the distributions of business and personal edges are different (business is much more frequent than personal), the number of dispreferred triads (heterogeneous triads) is higher than the number of all personal triads (PPP) in both graphs. However, the numbers and ratios change when we signs are permuted. Particularly, the number of all personal triads (PPP) increases by more than 350% from the permuted $G_{\text{all}}$ to the actual $G_{\text{all}}$, the number of BBP triads increases by more than 130% from the actual to permuted $G_{\text{all}}$, the number of all business triads (BBB) increases by 30% in the actual graph than the permuted one, while the number BPP triads almost remains the same.

These observations suggest that there is evidence for the global presence of structural balance in Enron$_{\cup}$, given the different notion of structural balance in this context. Specifically, the observations suggest that homogeneous triads (all business or all personal) are more preferred in the actual graph over a graph with permuted edge signs. However, given that the majority of edges are business, there are more heterogeneous triads than the homogeneous all personal triads.

## 6.4 Conclusion

In this chapter, we conduct social network analyses on different graphs induced from the labeled Enron emails. We use the Enron$_{\cup}$ dataset annotated with business and personal labels. We analyze the induced personal and business networks using different SNA measures, and we show that the two networks have different properties using these measures. In addition, we apply clustering algorithms on the core Enron user network, and study the type of emails exchanged between users; we show that the distribution of emails being business or personal is different, given that the emails are exchanged within the same cluster or between different clusters. Finally, we study the signed networks for the Enron users where edges connecting users labeled business or personal; we show that homogeneous triads (i.e., all business or all personal) are preferred in the actual graph over another graph constructed by permuting the edge signs. The summary of our findings is:

- The business and personal sub-networks have different graph properties using various SNA measures.
- People tend to cluster more often in the business sub-network than in the personal sub-network. In particular, for two nodes who share a common neighbor with whom they both exchange business emails, the chance that they will exchange a business email with each other is much higher than in the corresponding situation for personal emails.
- There are many more pairs who exchange only business emails than those who exchange only personal email.
- Given the different notion of the structural balance in this context, there is evidence for the global presence of structural balance in Enron$_{\cup}$. Specifically, homogeneous triads (all business or all personal) are preferred than heterogeneous triads.

These analyses presented in this chapter indicate that there are social network signals that can be used to distinguish between business and personal email. This motivates us to incorporate social network information for the classification task of emails into business and personal.

## Chapter 7

# Features Extracted from the Social Network

In chapter 5, we have presented various graph structures for the communication network such that different graph structures carry different information about the communications between users. For instance, the undirected user graph shows connections between users regardless of the direction of communication. These representations can be generalized to different genres and applications. For instance, the bipartite graph of documents and users can represent an email communication network such that users are linked to emails if they are the sender or recipients. Also, the same structure can be used to represent the Reddit communication network such that posts are linked to the set of the participating users. However, these representations cannot be directly used with standard machine learning classifiers without extracting features from them.

In this chapter, we present social network features for modeling the communication network. We extract various features from user and document nodes in the corresponding directed and undirected graphs of both the bipartite user-document graph and the user graph discussed in chapter 5. Some features are defined for only certain types of graphs (i.e., user vs. bipartite user-document; directed vs. undirected graphs), while other features are defined for all types of graphs. Then, we use these features with standard machine learning classifiers. These features are generic, and we use them in all text classification tasks in this thesis.

## 7.1 Social Network Feature Sets

In this section, we present our proposed social network features that can be extracted from different graph structures representing the communication network. We propose a variety of features that capture different information about users and their communication. Although these features are generic and can be generalized to different graph communication networks, we focus here on the social network graphs extracted from the email communication network. For other genres, the features still hold when using the graph structures here. However, the notion of senders and recipients can be different.

| Feature | Directed Graph? | Undirected Graph? | User Graph? | Bipartite Graph? |
|---|---|---|---|---|
| In-, Out-Degree | ✓ | | ✓ | ✓ |
| Total Degree | ✓ | ✓ | ✓ | ✓ |
| # Common Neighbors | | ✓ | ✓ | |
| # Sender's triangles | | ✓ | ✓ | |
| Jaccard's coefficient | | ✓ | ✓ | |
| Clustering coefficient | | ✓ | ✓ | |
| In-, Out-degree centrality | ✓ | | ✓ | |
| Degree centrality | ✓ | ✓ | ✓ | |
| Betweenness centrality | ✓ | ✓ | ✓ | ✓ |
| Eigenvector centrality | ✓ | ✓ | ✓ | ✓ |
| Closeness centrality | ✓ | ✓ | ✓ | ✓ |
| Auth-Hub Score | ✓ | ✓ | ✓ | ✓ |

Table 7.1: Social Network Features. Checkmarks indicate that a feature is extracted only from the corresponding graph(s).

Table 7.1 shows all the social network features we use in our experiments. We have chosen the feature names to be as self-explanatory as possible. We divide them into three sets, as indicated by double horizontal lines in Table 7.1. First, node features that can be computed from its edges only. Second, features extracted by considering the node and its neighbors (i.e., adjacent nodes). Finally, for the third set, the values on a node feature depend on the node position in the whole graph. These three sets of features allow us to extract local and global properties of individual nodes. Chapter 5 provides details on the mathematical definitions for the features used here.

### 7.1.1 First Feature Set

The features in the first feature set are extracted from a given node and its immediate edges. They capture the engagement of a given node with other nodes. The *in-degree* score of a node is the number of incoming edges to this node, while *out-degree* indicates the number of outgoing edges. Both *in-degree* and *out-degree* are defined for directed graphs. The *total degree* (or simply degree) of a node is the number of all edges connected to that node. It is defined for both directed and undirected graphs. For directed graphs, the *total degree* is the sum of *in-degree* and *out-degree*. For undirected graphs, it is the number of edges connected to the node.

Formally, the *in-degree* of the node $v$ in the graph $G$ is $indeg(v) = \sum_{u \in V} A_{u,v}$ where $V$ is the set of nodes (vertices) in $G$, and $A$ is the adjacency matrix of the graph $G$. $A_{v,u} = 1$ if there is an edge from $u$ to $v$; zero otherwise. Similarly, the *out-degree* of the node $v$ is $outdeg(v) = \sum_{u \in V} A_{v,u}$. The total degree $degree(v)$ for a node $v$ in directed graphs is the sum of in-degree and out-degree $degree(v) = indegree(v) + ourdegree(v)$. For undirected graphs, $indegree(v) = outdegree(v) = degree(v)$.

We extract these degree scores from both the user graph and the bipartite graph. In the user graph, the *in-degree* for the user node $v$ is the number of other users who sent at least one email to the user $v$. *out-degree* is the number of other users who received at least one email from $v$, and the *total degree* indicates the number of people who have exchanged emails (sent or received) with this user. In the bipartite graph, the *in-degree* score for a user node indicates how many emails have been received by this user, and the *out-degree* indicates how many emails have been sent by this user. The total degree is the amount of all emails in which the user is participating. For emails, *in-degree* is always equal to 1 (as any email always has only a single sender), so we ignore it. While out-degree indicates the number of recipients.

### 7.1.2 Second Feature Set

The second set of features measure dyadic relations, and they are extracted from the user graph only. Unlike the first set, features in this set capture information about pair relations and not only the local properties for a node. Then, extracting such features involve other nodes linked to the node of interest. For instance, for a given email, the features in this set measure the relation between the sender and the recipient(s).

*Number of common neighbors* measures the number of common nodes shared between the sender and recipient(s). The number of common neighbors alone might not be a good indicator of how close a pair of users is in case that one of them is part of many triangles. To overcome this issue, we calculate the *number of triangles* involving the sender. Then, we use it as a normalization factor for the number of common neighbors between the sender and recipient(s). The intuition is that if the sender has only a few triangles, then a high number of common neighbors indicates that the two users are well connected through common people. In contrast, a high number of triangles for the sender indicates that the sender is directly linked to many people who are linked to each other. We also compute *Jaccard's coefficient* score between the sender and recipient(s), which is simply the normalized number of common neighbors by the total neighbors (the union). The last feature in this set is the *local clustering coefficient*, which measures how close neighbors are for a given node to form a clique.

### 7.1.3 Third Feature Set

Features in the last set measure the global importance of nodes in graphs. The *degree centralities* are the normalized degree scores (*in, out,* and *total*) by the maximum possible degree. *Degree centralities* measure the importance of a node by looking at its direct neighbors. This might be useful for users but not emails as there are important emails sent to small number of users and less important emails sent to many users (e.g., announcements). Thus, we compute them only for users in the user graph. Other centrality measures: *betweenness*, *eigenvector*, and *closeness centralities* take into account nodes other than direct neighbors. We compute the scores for both user and email nodes in both the bipartite and user graphs. All centrality features compute the importance of a node differently.

## 7.2 Final Social Network Feature Vector

As we are interested in classifying documents (particularly emails), we extract features corresponding to the document (i.e., email) and its participants. For each email, we extract features described above from the corresponding email node in the bipartite email-user graph as well as features from both the sender and the recipients (either in the “to” or “cc” list ) from both the user graph and the bipartite email-user graph. In case the email has multiple recipients, we compute the max, min, and average of the value corresponding to each feature. Then, we feed these features to machine learning models. We also compute the weighted version for different features. For instance, we compute the weighted version for the in-degree for a user such that each email received by the user adds 1 to the weighted in-degree for that user.

# Chapter 8

# Overview of Methods Used

In this chapter, we describe the methods used in this thesis. We start by describing the general software frameworks that we use to build the different systems of analysis in section 8.1. We present the machine learning algorithms we will be using throughout this thesis in section 8.2. After that, we discuss different evaluation metrics for machine learning experiments in section 8.3. In section 8.4, we present our method for lexical modeling of the document content. Finally, in section 8.5, we present a state-of-the-art graph embedding model, GraphSAGE; we show how we use it for different email classification tasks. In the following chapters, we will compare the performance GraphSAGE with models that use our proposed social network features (presented in chapter 7).

## 8.1 Software Framework

In this section, we show the general software framework that we use throughout this thesis.

**scikit-learn** We make use of the scikit-learn Python package (Pedregosa *et al.*, 2011) as a general machine learning library. It provides implementations for various of machine learning classifiers, including: SVMs, Logistic Regressions, and Decisions Trees.

**NetworkX** is a Python package for the creation, manipulation, and study of the structure of networks (Hagberg *et al.*, 2008). We use this package for working with graphs throughout this thesis. Particularly, we use it to construct graphs and extract social network features from them.

**NLTK** the natural language toolkit (Loper and Bird, 2002), is a Python platform for natural language processing. It provides a suite of text processing libraries for classification, tokenization, stemming, and tagging. We make use of NLTK in this thesis for text processing.

**Stanford CoreNLP** (Manning *et al.*, 2014) provides a set of human language technology tools. We make use of this toolkit for text processing. Particularly, we use it for part-of-speech tagging and lemmatization in chapter 15.

**Keras, PyTorch** Keras (Chollet and others, 2015), and PyTorch (Paszke *et al.*, 2019) are open-source frameworks for deep learning. We use these frameworks for the implementation of neural network models throughout this thesis.

## 8.2 Machine Learning Classifiers

In this section, we briefly describe the main machine learning algorithms that we use in this thesis.

Machine learning approaches are traditionally divided into three broad categories: 1) supervised learning, 2) unsupervised learning, and 3) semi-supervised learning. In supervised learning, the learning algorithm is provided with example inputs and their output label, and the goal is to learn a general rule that maps inputs to outputs. In contrast, in unsupervised learning, the algorithm aims to learn a pattern from input data without knowing the output labels. Semi-supervised learning is used in scenarios where some (but not enough) supervision is provided to the algorithm. Specifically, semi-supervised learning uses both labeled and unlabeled data.

In this thesis, we mainly work on supervised machine learning algorithms for text classification. In this section, we overview the supervised learning methods that we use in this thesis.

**SVMs** A Support Vector Machine, introduced by Cortes and Vapnik (1995), is a supervised, binary, and discriminative classifier that finds a maximum-margin hyper-plane that optimally separates the instances in feature space so that it can classify unseen instances. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the "kernel trick", implicitly mapping their inputs into high-dimensional feature spaces. In various experiments in this thesis, we use two commonly used kernels: a linear kernel and a Gaussian radial basis function (RBF) kernel. We use the implementation for SVMs provided by the *scikit-learn* Python library.

**Decision Trees** A Decision Tree (DT) is a predictive supervised learning method expressed as a recursive partition of the feature space to sub-spaces that constitute a basis for prediction. One disadvantage of Decision Trees is that they are prone to overfit the training data. Random Forests address this issue by constructing multiple decision trees such that each decision tree is trained on a randomly selected subset of training data and features (ensemble learning). Similar to Random Forests, extremely randomized trees – commonly known as Extra-Trees Geurts *et al.* (2006), are an ensemble machine learning algorithm that combines the predictions from many decision trees. The key difference between random forests and extra trees is that random forests subsample the training data with replacement, while extra trees use the whole data. In this thesis, we use the implementation for Extra Tress provided by the *scikit-learn* Python library.

**Logistic Regression** A logistic regression classifier is a linear model that learns the relation between the input and the target value by linearly combining input values using weights or coefficient values. Then, the combined values are transformed using the logistic function (or the sigmoid function). In the experiments throughout this thesis, we use the implementation for logistic regression classifiers provided by the *scikit-learn* Python library.

**Neural Networks** Artificial neural networks (ANNs) or simply neural networks are a family of nonlinear machine learning algorithms. They are typically organized in layers, and each layer is made up of a number of interconnected nodes. In this thesis, we use different neural networks models. We use the keras and PyTorch frameworks for the implementation of neural network models throughout this thesis.

### 8.2.1 Dummy Classifiers

Throughout this thesis, we show the results of dummy classifiers that we use as baselines. These baselines do not learn from any features except the class distributions. We use two kinds of dummy classifiers: random classifier and All-Class classifier. Below, we describe each of them in detail.

**Random Classifier** A random classifier predicts the labels for the test set with respect to the class distribution in the train set. Due to its randomness, each run will return different results. Instead of reporting the results for a single run or the average of multiple runs for a random classifier, we report the expected value that can be computed using the prior class probabilities.

**All-Class** Another baseline we use in different experiments in this thesis is the ”all-class” classifier. This classifier simply assigns one of the classes to every example in the evaluation set.

## 8.3 Evaluation Metrics

In this section, we present the evaluation metrics for the machine learning experiments we use throughout this thesis. We use Recall, Precision, F-measure, and Accuracy (or Error Rate) as the classification performance measures, which have been conventional in benchmark evaluations for text classification tasks. We first define terms associated with these metrics: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

In supervised learning, for a given class and an example in the data, a model will learn to assign a label for that class (positive), or a label for another class (negative) to that example. For binary classification tasks, we can fix one class to be the positive class and the second class to be the negative class. In multi-class classification, we define these numbers for each class such that when the label represents that class, we consider it positive; otherwise, negative.

**True Positive (TP)** are data points classified as positive by the model that are actually positive (correct).

**False Positive (FP)** are data points the model identifies as positive that are actually negative (incorrect).

**True Negative (TN)** are data points classified as negative by the model that are actually negative (correct).

**False Negative (FN)** are data points the model identifies as negative that are actually positive (incorrect).

**Accuracy** measures the proportion of the total number of predictions that were correct. It is the proposition of True Positives and True Negatives.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

For some tasks, when the dataset is imbalanced, with one category representing the overwhelming majority of the data points, the accuracy score might not be a useful metric for the classifier performance. For example, a classifier that always predicts the majority class can achieve a high accuracy score when the dataset is imbalanced. Below we show other metrics that tackle this problem.

**Recall** is the proportion of actual positives that were identified correctly. It measures the model's ability to find all the data points for a given class in a dataset.

$$Recall = \frac{TP}{TP + FN}$$

**Precision** is the proportion of positives that was actually correct. It measures the model's ability to identify only the relevant data points for a given class in a dataset.

$$Precision = \frac{TP}{TP + FP}$$

Recall and precision can be maximized at the expense of the other metric. For instance, a model that maximizes the recall score can simply assign the label we are interested in to all instances. However, this will decrease the precision for the given class. On the other hand, when we maximize the precision, the model might correctly predict only a few instances to be relevant while missing many relevant instances for a given class.

**F-1** is the harmonic mean of precision and recall taking both metrics into account. Therefore, optimizing the F-1 score overcomes the issue above.

$$F\text{-}1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

## 8.4 Lexical Modeling

Word embeddings is the collective name for a set of language modeling and feature learning techniques. It is a learned representation for text where words that have the same meaning have a similar representation. Word embedding models map each word in a vocabulary into a $d$-dimensional vector by creating a matrix in $\mathbb{R}^{N \times d}$ from a vocabulary with $N$ words such that semantically similar words are similar in vector space.

We use word embeddings for lexical modeling of document content in different classification tasks in this thesis. There are different models in the literature for obtaining word embeddings from the textual content. In this thesis, we use two commonly used word embedding models: *FastText* (Bojanowski *et al.*, 2017) and *Glove* (Pennington *et al.*, 2014).

For *FastText*, we use the CBOW mode with the default argument values. Arguments include the size of word vectors (dimensions), the size of the context window, and the maximum and minimum length of n-grams. To represent a document, we average the corresponding vector for each character n-gram of every word in the document; then, we compute the average vector for all words in the document.

For *GloVe*, we use various pre-trained *GloVe* (Pennington *et al.*, 2014) vector sets that are available online, each trained using different corpora and embedded into various dimension sizes. We use GloVe pre-trained word vector sets such that each document is represented by a vector of a fixed number of dimensions equal to the dimensionality of GloVe word vector set. We average all word vectors in the document using the pre-trained word vectors as follows:

$$d_j = \frac{\sum_i^n f_{ij}\mathbf{v}_i}{\sum_i^n f_{ij}}$$

Where $f_{ij}$ is the frequency of the word corresponding to vector $\mathbf{v}_i$ in document $d_j$, $\mathbf{v}_i$ is the word embedding vector in GloVe set.

## 8.5 GraphSAGE

GraphSAGE (SAmple and aggreGatE), introduced by Hamilton *et al.* (2017a), is a recent state-of-the-art inductive model for learning node embeddings for different tasks, including node classification. Unlike most existing approaches for generating node embeddings that are inherently transductive, GraphSAGE can generalize to previously unseen data. It learns an embedding for a given node by aggregating information from its neighboring nodes and from the attributes of that node.

The key idea behind GraphSAGE that it learns a function that maps a node to low-dimensional representation by aggregating neighboring nodes' attribute information. Node attributes can be the textual content for the node represented as the average word embeddings for the document corresponding to that node.

We use GraphSAGE for the classification tasks in this thesis and compare its performance with models that use our proposed social network features in chapter 7. Particularly, for email classification tasks: email classification into business and personal (chapter 13), overt display of power detection (chapter 15), and hierarchical power detection (chapter 16). For Reddit post classification (chapter 17), we replicate experiments presented in the original study (Hamilton *et al.*, 2017a), and we compare their results with the results of our models.

GraphSAGE is designed for homogeneous graphs where nodes belong to one type. To make use of GraphSAGE in our experiments on email tasks, we construct a document graph (section 5.4), which has only emails as nodes (since we also need access to the lexical content for GraphSAGE). In this graph, nodes represent emails, and edges link emails if they share a certain percentage of participants. We do not distinguish between senders and recipients as participants. Then, we feed the GraphSAGE supervised model with this graph of emails with their corresponding labels, and furthermore, we use the lexical features described in section 8.4 as node attributes.

We use the Jaccard similarity to measure the similarity between the participant sets of two emails, and then, we link two emails with an edge if their similarity score is above a certain threshold. We define Jaccard similarity $J$ between two emails as:

$$J(e_i, e_j) = \frac{\mid \tau(e_i) \cap \tau(e_j) \mid}{\mid \tau(e_i) \cup \tau(e_j) \mid}$$

Where $\tau(e_i)$ denotes the set of participants in email $e_i$ (both the sender and recipients). In different experiments throughout this thesis, we use different threshold values for $J$ such that we find the threshold value that optimizes the performance for a given task.

# Part II

# Business and Personal Email Classification

## Chapter 9

# Introduction to Part II

In this part, we present the core work of this thesis: email classification into business and personal. We are interested in modeling both the email content and the underlying social network of interaction. We assume that the ultimate application of our work is a setting in which we train models on a company (i.e., Enron) and apply them to another company (i.e., Avocado).

In this thesis, we study document classification in the context of written conversations. As our main task, we choose the classification of email into personal or business. There are several reasons for this choice:

- We are interested in how personal relationships affect communication, taking into account that the same pair of people may have multiple types of relationships.
- The task we choose is relevant. Email remains a crucial communication medium for both individuals and organizations for both personal and business communications. Kiritchenko and Matwin (2011) show that a typical user daily receives 40-50 emails. And despite the massive growth of other social media over the past decade, company email is still used for personal purposes, as the recent Avocado corpus shows (section 4.3).
- Unlike other text classification tasks, particularly for emails (e.g., spam filtering), email clas-

sification into business and personal has not received much attention, and it remains a challenging (as shown in the human inter-annotator agreement we discussed in section 4.1 and reported in (Jabbari *et al.*, 2006) and unsolved task.

- The social network of interaction is relevant, as we discussed earlier in this thesis. We have shown that business and personal emails form different networks of communication (chapter 6).
- Two large data sets are available, the Enron corpus and a data set of emails from an anonymous defunct information technology company referred to as Avocado.

In this chapter, we give an overview of this part of the thesis. In section 9.1, we discuss our motivation and problem definition. We show the datasets we use throughout this part in section 9.2.

## 9.1 Motivation

Given an email sent from a user $u$ to a set of participants $\{p_1, p_2, ...p_n\}$ we want to classify the email into "business" or "personal" categories. We have shown in section 6.1 that the same pair of sender and recipient might exchange both business and personal emails. However, there are many pairs who exchange either only business or only personal emails. In addition, we have shown in section 6.1 that the induced personal and business networks have different properties. This motivates us to incorporate social network information for improving the classification performance. We are also interested in modeling the threads as they tend to discuss the same topic. We show in subsection 9.2.1 that threads tend to contain either all business or all personal emails, with a few exceptions.

## 9.2 Datasets

For the task of email classification into "business" and "personal", we use two corpora: Enron and Avocado. In chapter 4, we have introduced the datasets and annotations for the task of email classification into "business" and "personal". We train on Enron and test on Enron and Avocado.

We use four datasets for the experiments throughout this part, namely: Enron$_{\cup}$, Enron$_{T}$, Enron$_{\cap A}$, and Avocado$_{T}$ (described in section 4.2 and section 4.3). We exclude Avocado$_{\cup}$ from our experiments since that it is the same as Avocado$_{T}$, but it has only three extra emails (emails that belong to threads with some emails without labels).

We split Enron$_{\cup}$, Enron$_{T}$, and Enron$_{\cap A}$ into train, development, and test sets with 50%, 25%, and 25% of the emails, respectively. We divide Avocado$_{T}$ equally into development, and test sets (since we will not train on Avocado). The datasets are partitioned into train, development and test sets chronologically by the time of their first email, such that the training set contains the earliest threads, and the test set contains the latest threads. For the rest of this part, we refer to the train, development and test sets by subscripts $tr, dev$, and $tes$, respectively. Note that Enron$_{\cap A}$ $\subset$ Enron$_{T}$ $\subset$ Enron$_{\cup}$; then their train, development, and test sets are also subsets of the larger corresponding set (e.g., Enron$_{\cap A\ tr}$ $\subset$ Enron$_{T\ tr}$). Note that not all emails in Enron$_{\cup}$ belong to threads, and some threads in Enron$_{\cap A}$ are not complete; i.e., some emails are not included (see section 4.1 for details).

The three Enron datasets: Enron$_{\cap A}$, Enron$_{T}$, and Enron$_{\cup}$ differ in size and the quality of the labels. Enron$_{\cap A}$ is the smallest one, with the highest agreement on the labels. Ultimately, we are interested in testing the best models on Enron$_{T}$ as it is the only dataset that maintains the thread structure.

| State | Business | Personal | End |
|---|---|---|---|
| Beginning | 3,439 (87.7%) | 484 (12.3%) | 0 |
| Business | 5,606 (61.6%) | 79 (0.9%) | 3,412 (37.5%) |
| Personal | 52 (3.72%) | 834 (59.7%) | 511 (36.6%) |

| State | Business | Personal | End |
|---|---|---|---|
| Beginning | 73 (66.4%) | 37 (33.6%) | 0 |
| Business | 60 (32.4%) | 79 (42.7%) | 46 (24.9%) |
| Personal | 52 (31.9%) | 47 (28.8%) | 64 (39.3%) |

Table 9.1: Transition probability for threads in Enron$_{Tcomp}$. The left table for all threads, the right one for mixed threads. The absolute number of emails is given.

### 9.2.1 Sequence of Labels in Threads

In this subsection, we analyze the sequence of email labels in threads. We use Enron$_T$ except for 18 threads (57 emails) that have some emails without labels ("cannot determine"). We refer to this set as Enron$_{Tcomp}$. We use this data set because it maintains the thread structure, and every email belongs to a complete thread. The total number of threads in Enron$_{Tcomp}$ is 3,923, and the total number of emails is 10,494.

First, we investigate whether there are some threads where the dialogue shifts from personal to business or vice versa. We find that there are 3,366 (85.8%) business threads (contain only business emails), 447 (11.39%) personal threads, and 110 (2.8%) mixed threads that have both personal and business emails. Table 9.1 shows transition probabilities (and the absolute number of emails) of all threads and mixed threads in Enron$_{Tcomp}$. We observe that shifting from business to personal happens more often than shifting from personal to business.

This shows that there are very few threads in which the conversation shifts from personal to business or vice versa, which motivates us to model the thread structure of emails. In case that there is an ambiguous email in a given thread, the rest of the emails in this thread would help to classify it. We present in chapter 12 our work in thread modeling.

| Baseline | Set | Accuracy | Business F-1 | Personal F-1 |
|---|---|---|---|---|
| Expected Random | Enron$_{\cup}$ | 68.6 | 80.5 | 19.5 |
| | Enron$_{\cap A}$ | 78.9 | 88 | 12 |
| | Enron$_{T}$ | 76.9 | 86.7 | 13.3 |
| | Avocado$_{T}$ | 83.9 | 91.2 | 8.8 |
| All-Business | Enron$_{\cup}$ | 80.5 | 89.2 | 0 |
| | Enron$_{\cap A}$ | 88 | 93.6 | 0 |
| | Enron$_{T}$ | 86.7 | 92.9 | 0 |
| | Avocado$_{T}$ | 91.2 | 95.4 | 0 |

Table 9.2: Results of different baselines trained on the corresponding train set and tested on the corresponding development set. Here, we report the expected values for the random classifier.

## 9.3 Simple Baselines

We define two simple baselines: a random classifier, and an all-business classifier. The former predicts the classes by respecting the train set's class distribution, while the latter predicts the majority class (i.e., "business"). Table 9.2 shows the results of these two baselines on the different datasets described in chapter 4. For both baselines, we report the performance on the development set based on the class distribution of the corresponding train set.

The random baseline can be used to measure a model's performance on the minority class (personal). For the all-business baselines, the personal F-1 score could be trivially beaten (zero score), but it is harder to beat the business F-1 score since that the datasets are highly unbalanced (all data sets have more than 80% business emails). A model that has a personal F-1 score higher than the random classifier, and a business F-1 score higher than the all-business classifier, we consider it robust.

In addition to these two simple baselines, we conduct experiments with a more robust model, namely GraphSAGE. We compare its performance with our models in chapter 11.

| Classifier | Parameter | Parameter Space |
|---|---|---|
| SVM | $\gamma$ | $10^{-4,-3,-2,-1,0}$ |
| | kernel | RBF, linear |
| | $C$ | $1, 10, 100, 1000$ |
| Extra-Trees | # trees | $10, 20, 30, 50, 100, 200$ |
| | Split Criteria | Gini, Entropy |
| | Min Sample | 1, 3, 10 |
| DNNs | # Hidden Layers | $1, 2, 3, 4$ |
| | # Units | $1, 5, 10, 20, 50, 100$ |
| | Loss Func. | Hinge, Binary Cross Entropy |
| | Activation | Linear, ReLu, Tanh, Sigmoid |
| All | Class-weights | {B:1, P:1}, {P:1, B:2} {P:1, B:3}, balanced |

Table 9.3: Machine learning classifiers' hyperparameter space. B: Business, P: Personal. Balanced: class weights are adjusted inversely proportional to class frequencies in the training set.

## 9.4 Machine Learning Setup

For modeling individual emails, we experiment with three classifiers: Deep Neural Networks (DNNs), Support Vector Machines (SVMs), and extremely randomized trees (commonly known as Extra-Trees) (Geurts *et al.*, 2006). Table 9.3 shows the hyper-parameter space for these classifiers.

For DNNs, we use a multi-layer perceptron (MLP) with different hyperparameters (i.e., number of hidden units, loss functions, etc.). We experiment with both binary cross-entropy and hinge loss as the loss function. For the non-linearity activation on the output layer, we use the *sigmoid* function when using the binary cross-entropy loss, and *tanh* with hinge loss. For hidden layer activation functions, we experiment with all activation functions presented in 9.3. We use the Keras framework (Chollet and others, 2015) for the implementation of neural networks, and we use the Adam optimizer (Kingma and Ba, 2014) with the default parameters as provided by Keras.

For SVMs and Extra-tress, we tune the hyper-parameters using grid-search with 3-fold cross-validation on the train set. For neural networks, we tune on the development set using grid-search. For sequential modeling of emails in threads, we apply two approaches: majority-vote, and LSTMs. We discuss them in detail in chapter 12. In all experiments in this part, we optimize the Personal F-1 score since we are interested in identifying personal emails, which are rare. We also report accuracy and Business F-1. We report all three measures, along with recall and precision, since all measures together give a more complete understanding of the performance of our classifiers. In our results, we report the model with the optimal hyper-parameters that maximize the Personal F-1 score.

## 9.5 Part Outline

In the following chapters, we show results for different experiments for email classification into business and personal. In chapter 10, we present experiments using lexical modeling methods for the email content. Then, we conduct different experiments with social network modeling of the communication network in chapter 11. In chapter 12, we discuss techniques for thread modeling instead of modeling individual emails. We show in chapter 13, graph embedding models as alternative approaches for modeling the social network.

# Chapter 10

# Lexical Modeling

In this chapter, we present methods and experiments for modeling the email lexical features for the business and personal email classification task. We present results for different word embedding models. We evaluate models on both Enron and Avocado.

We present methods for modeling the email content in section 10.1. In section 10.2, we evaluate different lexical models for the email classification task, and we select the best model with the highest performance for the subsequent experiments. In section 10.3, we conduct experiments using a variety of machine learning classifiers on different datasets described in chapter 4. Then, in section 10.4, we conduct a post-hoc analysis to investigate the difference in the performance of the lexical models on Enron and Avocado. We conclude the chapter in section 10.5

## 10.1 Modeling Emails

For modeling the lexical content for emails, we use texts from both the subject and email body. We use a TF-IDF model as a baseline. We use two word embedding models to represent emails as a low-dimensional vector: GloVe and FastText. For GloVe, we use off-the-shelf pre-trained vectors. For FastText, we pre-train the word embedding vectors on the whole Enron collection. As a prepossessing step, we replace every digit with “8”, and lowercase all letters except for models trained on cased datasets.

| | Vector Set | Accuracy (%) | Business F-1 (%) | Personal F-1 (%) |
|---|---|---|---|---|
| | TF-IDF | 94.0 | 96.6 | 70.6 |
| Pre-trained Glove Vectors | 6B.50d | 91.6 | 95.2 | 64.1 |
| | 6B.100d | 92.6 | 95.8 | 67.5 |
| | 6B.200d | 93.5 | 96.4 | 71.1 |
| | 6B.300d | 92.9 | 96.0 | 70.8 |
| | 27B.25d | 92.9 | 96.0 | 70.4 |
| | 27B.50d | 91.6 | 95.2 | 63.2 |
| | 27B.100d | 91.9 | 95.4 | 66.3 |
| | 27B.200d | 93.2 | 96.2 | 71 |
| | 42B.300d | 94.1 | 96.7 | 75.5 |
| | 840B.300d | 94.0 | 96.6 | 74.7 |
| Trained on Enron | FastText | **94.3** | **96.8** | **76.0** |

Table 10.1: Results of SVM classifiers trained using FastText versus different sets of pre-trained GloVe word vectors; a TF-IDF model is shown as a baseline. All classifiers are trained on Enron$_{\cap A\ tr}$ and tested on Enron$_{\cap A\ dev}$. Glove vector set names denote the corpus size (number of tokens) and dimensions, e.g., 27B.25d is trained on a corpus of 26 billion tokens; the vector size is 25.

## 10.2 Obtaining Best Word Embedding Vector Set

In this section, we show results of different word embedding vectors evaluated on Enron$_{\cap A}$ to obtain the best word embeddings as lexical features for the email classification task into business and personal. We use various *GloVe* pre-trained vector sets (Pennington *et al.*, 2014) and task-specific embeddings of emails using *FastText* (Bojanowski *et al.*, 2017) trained on the whole Enron email collection. In early experiments of our work, we used only *GloVe* vectors. Then, when *FastText* was released, we used it in our experiments for lexical modeling.

There are various pre-trained *GloVe* vector sets available online; each set was trained on different corpora and embedded into multiple dimension sizes. We experiment with different pre-trained *GloVe* vector sets.

We use GloVe pre-trained word vector sets such that each email is represented by a vector of a fixed number of dimensions equal to the dimensionality of GloVe word vector set. We average all word vectors in the email using the pre-trained word vectors as follows:

$$e_j = \frac{\sum_i^n f_{ij}\mathbf{v}_i}{\sum_i^n f_{ij}}$$

Here, $f_{ij}$ is the frequency of the word corresponding to vector $\mathbf{v}_i$ in email $e_j$, $\mathbf{v}_i$ is the word embedding vector in GloVe set.

For *FastText*, we use task-specific embeddings trained on the whole Enron email collection. To represent an email, we average the corresponding vector for each character n-grams of every word in the email; then, we compute the average vector for all words in the email.

We use the CBOW mode for *FastText* embeddings with the default argument values. Arguments include the size of word vectors (dimensions), the size of the context window, and the maximum and minimum length of n-grams.

Table 10.1 shows the results of classification using different word embeddings trained on Enron$_{\cap A\ tr}$ and tested on Enron$_{\cap A\ dev}$; a Term Frequency-Inverse Document Frequency (TF-IDF) model is shown as a baseline. In this model, we represent each email as a vector of normalized term frequencies. To reduce the high dimensionality of the term (i.e., word) vectors, we select the top 500 words using the $\chi^2$ feature selection method. Here, we use individual emails only and we do not look at other emails in the same thread.

In the GloVe pre-trained vectors, we use different sets of pre-trained vectors available online. These sets differ in the datasets used to obtain the embeddings, as well as the dimension of the final vectors. Vector set names denote the corpus size (number of tokens) and dimensions. For example, the 27B.25d model is trained on a corpus of 26 billion tokens; the vector size is 25. The 42B.300d and 840B.300d models are trained on the Common Crawl dataset [1]; the 27B models are trained on a Twitter dataset; and the 6B models are trained on a Wikipedia dataset.

In all models (i.e., word embeddings and TF-IDF), we use SVM classifiers with different kernels and we tune the hyper-parameters using grid-search. Table 9.3 shows the hyper-parameter space for the SVM classifiers.

The results show that classification of emails using FastText embeddings outperforms GloVe models. For GloVe pre-trained vectors, in general, more training data is better, and more dimensions are better. However, the best pre-trained GloVe vector set is the 300-dimensional 42B.300d which is trained on a large corpus of 42 billion tokens, rather than the larger 840 B words-based embeddings. Note that the 42B.300d model is uncased, while the 840B.300d is cased. In the latter, we do not lowercase the vocabulary. Being cased might explain the relatively lower performance of the 840B.300d which is trained on a larger dataset in comparison to the 42B.300d model. In addition, the TF-IDF model performs quite well, and only the top embeddings models only outperform it. From these results, we decide to use the FastText embeddings for lexical modeling of emails in all further experiments.

## 10.3 Intra-corpus and Cross-corpora Performance

In this section, we show the results of experiments for lexical classification of the email content on two settings: intra-corpus and cross-corpora. In the intra-corpus setting, we train and test on

[1] `https://commoncrawl.org/`

| | | | | Business | | | Personal | | |
|---|---|---|---|---|---|---|---|---|---|
| Train set | Classifier | test set | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| $\text{Enron}_{\cup}$ | SVM | Enron | 89.74 | 93.49 | 92.07 | 94.97 | 75.73 | 80.38 | 71.59 |
| | | Avocado | 81.29 | 88.73 | 80.64 | 98.63 | 44.87 | 88.16 | 30.09 |
| | Extra-Trees | Enron | 89.72 | 93.53 | 92.88 | 94.2 | 74.90 | 77.02 | 72.89 |
| | | Avocado | 86.06 | 91.9 | 86.57 | 97.94 | 50.0 | 80.7 | 36.22 |
| | Neural-net | Enron | 90.35 | 93.99 | 93.49 | 94.5 | 75.49 | 77.17 | 73.87 |
| | | Avocado | 87.69 | 92.91 | 88.27 | 98.07 | 53.37 | 81.58 | 39.66 |
| $\text{Enron}_{T}$ | SVM | Enron | 91.02 | 94.76 | 93.61 | 95.94 | 68.48 | 73.94 | 63.76 |
| | | Avocado | 91.44 | 95.21 | 93.16 | 97.36 | 59.64 | 73.25 | 50.3 |
| | Extra-Trees | Enron | 89.69 | 93.97 | 92.62 | 95.36 | 64.29 | 70.36 | 59.18 |
| | | Avocado | 90.15 | 94.47 | 92.08 | 96.99 | 55.02 | 69.74 | 45.43 |
| | Neural-net | Enron | 91.02 | 94.75 | 93.32 | 96.22 | 69.04 | 75.9 | 63.32 |
| | | Avocado | 91.74 | 95.4 | 93.78 | 97.08 | 59.48 | 70.18 | 51.61 |
| $\text{Enron}_{\cap A}$ | SVM | Enron | 94.28 | 96.75 | 96.59 | 96.92 | 75.95 | 76.92 | 75.0 |
| | | Avocado | 91.17 | 95.05 | 92.83 | 97.39 | 59.05 | 73.68 | 49.27 |
| | Extra-Trees | Enron | 94.28 | 96.76 | 96.76 | 96.76 | 75.64 | 75.64 | 75.64 |
| | | Avocado | 91.14 | 95.07 | 93.49 | 96.7 | 56.34 | 66.23 | 49.03 |
| | Neural-net | Enron | 94.28 | 96.72 | 95.56 | 97.9 | 77.65 | 84.62 | 71.74 |
| | | Avocado | 89.85 | 94.24 | 90.96 | 97.77 | 57.05 | 78.07 | 44.95 |

Table 10.2: Results of lexical models trained on $\text{Enron}_{\cup}$ , $\text{Enron}_{T}$, and $\text{Enron}_{\cap A}$. For Enron, models are trained on the corresponding train set and tested on the corresponding development set. For Avocado, all models tested on $\text{Avocado}_{T\ dev}$.

Enron, while in the cross-corpora setting, we train on Enron and test on Avocado. We experiment with different Enron datasets in subsection 4.2.4; each dataset has a different class distribution and size. $\text{Enron}_{T}$ is our annotation for Enron, and this dataset maintains the thread structure such that all emails belong to threads. $\text{Enron}_{\cup}$ is the union of $\text{Enron}_{T}$ and the Sheffield dataset (subsection 4.2.2). In case of disagreement in labels between the two sets, we use the Sheffield labels. $\text{Enron}_{\cap A}$ is the intersection of $\text{Enron}_{T}$ and the Sheffield datasets such that both agree on the label. Note that $\text{Enron}_{\cup}$

is the largest set with the highest disagreement in the labels. Enron$_{\cap A}$ is the smallest with the highest agreement in the labels. For more information about the datasets, see section 4.2. We experiment with different machine learning classifiers discussed in section 9.4.

Table 10.2 shows the results for different lexical models trained on different Enron datasets and tested on Avocado$_{T\ dev}$ and the corresponding Enron dataset. We observe that the performance in the cross-corpora setting is lower than the performance in the intra-corpus setting. The difference in the performance between Enron and Avocado is lowest when we train on Enron$_T$. For Enron, since we are testing on different datasets with different class distributions, there is no direct comparison between the models trained and evaluated on different Enron datasets. However, we observe that models on Enron$_{\cap A}$ perform relatively better than other datasets. This is due to the fact that Enron$_{\cap A}$ has easier emails as the human agreement is higher than the other sets.

For Avocado, models trained on the largest Enron dataset Enron$_{\cup}$ perform the worst. Models tested on Avocado tend to over-predict the personal class, especially when train on Enron$_{\cup}$. Note that this dataset has the largest percentage of personal emails (20%), while Avocado has less than $9\%$ of personal emails. The relatively high percentage of personal emails in Enron$_{\cup}$ causes the high personal recall when we train on Enron$_{\cup}$ and evaluate on Avocado. Models trained on Enron$_T$ and Enron$_{\cap A}$ perform very similar when evaluated on Avocado. In general, models trained on Enron$_T$ perform slightly higher; the best model on Avocado (using personal F-1 score) is SVM trained on Enron$_T$.

We observe that SVMs and Neural Networks perform better than Extra-Trees when evaluating on Enron. On Avocado, we observe the same pattern except that SVMs perform poorly when trained on Enron$_{\cup}$. In general, SVMs tend to predict more personal emails on Avocado than other models, except for Neural Networks trained on Enron$_{\cap A}$.

## 10.4 Post-hoc Analysis

To investigate the relatively low performance of the lexical features on Avocado compared to performance on Enron, we select the top 1,000 words using $\chi^2$ and TF-IDF in business and personal emails for both Avocado and Enron. We find that there are only 7 business words in common: *changes*, *information*, *issue*, *meeting*, *please*, *review*, and *thanks* but more than 150 personal words in common. Examples of common personal words: *birthday*, *girlfriend*, *lunch*, *saturday*, *surgery*, and *xmas*. We also find that most of the business words that are in the top word in Avocado but not in Enron are IT-related terms such as: *application*, *bug*, *hp*, and *wireless*. In contrast, the top business words in Enron but not in Avocado are general business terms such as: *agreement*, *committee*, *contract*, *market*, and *transaction*, but a few words are related to Enron business (i.e., Energy) such as *gas* and *energy*. For personal words that are not common, we find words such as names of places (e.g., *alabama*) and hobbies (e.g., *tennis*). See Appendix A for the complete lists of words.

## 10.5 Conclusion

We have shown in this chapter experiments on modeling the lexical content of emails for the task of email classification into business and personal. We experiment with various classifiers using features from two word embedding models: GloVe and FastText. We also use TF-IDF as a baseline. The results show that embeddings obtained from the more recent FastText model outperform those from GloVe. In addition, we observe that the performance drops when we test on another corpus (i.e., Avocado) and our analysis shows that the two companies, Enron and Avocado, have different words used in the two different classes. Particularly, for the business class, as the two companies are in two different industries, i.e., Energy and Information Technology. Business terms are quite different among the two companies.

# Chapter 11

# Social Network Modeling

In this thesis we are interested in incorporating social network information for different text classification tasks. As our main task, we want to incorporate the social network information for email classification into business and personal. We have shown in chapter 6 that different SNA metrics for business and personal emails are distinguishable. This motivates us to incorporate information from the underlying social network of communication for the task of email classification into business and personal. In this chapter, we present models that incorporate social network information for the task of business and personal email prediction. Particularly, we use features introduced in chapter 7 for the task of modeling business and personal emails. We first show the summary of our approach to exploiting the social network induced from the email communication network. Then in section 11.2, we present results for intra-corpus evaluations in which we conduct experiments on Enron. In section 11.3, we show results of the cross-corpora experiments where we train models on Enron and test on Avocado. We conduct a post-hoc analysis in section 11.4 where we show social network weights. We conclude the chapter in section 11.5.

## 11.1 Experimental Settings

In this section, we present our approach for incorporating the social network information for the email classification into business and personal.

The email exchange network can be represented as graphs with different structures. One possible structure is to represent the email exchange network as a bipartite graph with two disjoint sets of nodes: documents (i.e., emails) and users (i.e., people) such that edges link documents with users, as edges between an email and users exist if and only if their email address appears as either the sender or a recipient in that email; we refer to this structure as the *bipartite user-email network*. Another structure is a graph (not bipartite) whose nodes represent people (i.e., email addresses) and whose edges represent email communication such that and edge exists if there is a least one email has been exchanged between the two end nodes; we refer to this structure as the *user network*. We have shown in chapter 5 how to construct these graphs.

For each corpus (i.e., Enron and Avocado), we construct directed and undirected graphs from these two networks (i.e., the *bipartite user-email network* and the *user network*) such that we have four graphs in total: undirected vs. directed, and bipartite user-email vs. user graphs.

In directed graphs, each edge has a source and destination node, which shows explicitly the directionality of the email (i.e. sender and recipients), while in undirected graphs, the directionality of communication is not reflected within edges. For the user graph, the edge weight reflects the number of emails that have been exchanged between the two ends and the direction; for the user-email graph, the weights are always 1. Then, we extract different social network features from these 4 graphs: undirected and directed, bipartite graph and user graph. We have shown the social network features we extract in chapter 7. For a given corpus (i.e., Enron or Avocado), we use the whole email collection, including all labeled and unlabeled emails to build these four graphs. We include features from both the sender and the recipients (either in the "to" or "cc" list ). In case that the email has multiple recipients, we compute the average, max, min of the value corresponding to each feature.

In addition to the social network features, we use the lexical features described in chapter 10. We show results for models trained using the social network features, the lexical features, and the combination of both.

We use three machine learning classifiers: SVMs, Extra-trees, and Feed-forward Neural Networks. We have shown in section 9.4 details for these classifiers.

For all classifiers we use three feature sets: net; using social network features only (chapter 7), lexical, using word embeddings only (chapter 10), all, is the combination of the two feature sets. In the all feature setting, for neural networks, we concatenate the two networks (branches) of the lexical and the network features. For SVMs, we compute the average of the two kernels (a kernel for each feature set). We use three metrics to measure the performance, namely: accuracy score, Business F-1 score and Personal F-1 score. We are mainly interested in optimizing the Personal F-1 score since it is the minority class. Similar to experiments in chapter 10, we conduct experiments with different Enron datasets; each has different class distribution and human inter-annotator agreement. For more information about the datasets, see subsection 4.2.4.

## 11.2 Intra-corpus results

In this section, we show experiments on using social network features as well as combining them with lexical features discussed in chapter 10. We train the models and test them on the same company, Enron.

Table 11.1 shows the results for the intra-corpus setting. The results show that all classifiers beat the random baseline on all datasets by using social network features only. However, by using social network features only, the only models that beat the all-business baseline on the business F-1 scores are Extra-trees and Neural Networks on Enron$_{\cup}$.

We observe that the lexical features alone always perform better than the social network features alone. However, incorporating social network information with lexical features improves the performance (an exception is Extra-trees on Enron$_{\cap A}$). The improvement by adding social network features is lowest in Enron$_T$ in which models also have the the lowest personal F-1 scores. Although Enron$_T$ and Enron$_{\cap A}$ have similar distributions of business and personal emails, models on Enron$_T$ have substantially lower scores in all feature sets than Enron$_{\cap A}$. Note that Enron$_{\cap A}$ is the intersection between Enron$_T$ and Sheffield$_{all}$ for which both agree in labels, which suggests that Enron$_{\cap A}$ is easier than Enron$_T$.

We also see that models with social network features only have substantially higher personal F-1 scores (and very similar business F-1 scores) on Enron$_{\cup}$ than other sets. We believe this is because the ratio of personal emails is higher in Enron$_{\cup}$ than in other sets and the size of Enron$_{\cup}$ is double the size of Enron$_T$.

These observations and results suggest that our proposed social network features require more data relative to the lexical features as we see that the difference in performance between net and lexical features is much higher on Enron$_{\cap A}$ and Enron$_T$ than on Enron$_{\cup}$.

| | | | | Business | | | Personal | | |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | Classifier | features | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| Enron$_{\cup}$ | SVM | Net | 83.11 | 89.05 | 85.75 | 92.62 | 63.11 | 72.51 | 55.86 |
| | | Lexical | 89.74 | 93.49 | 92.07 | 94.97 | 75.73 | 80.38 | 71.59 |
| | | All | 90.64 | 94.05 | 92.46 | **95.71** | **78** | **83.32** | 73.31 |
| | Extra-Trees | Net | 83.36 | 89.25 | 86.25 | 92.47 | 63.22 | 71.77 | 56.48 |
| | | Lexical | 89.72 | 93.53 | 92.88 | 94.2 | 74.90 | 77.02 | 72.89 |
| | | All | 89.99 | 93.66 | 92.28 | 95.08 | 76.28 | 80.8 | 72.23 |
| | Neural-net | Net | 83.73 | 89.7 | 87.75 | 91.74 | 61.26 | 66.84 | 56.54 |
| | | Lexical | 90.35 | 93.99 | **93.49** | 94.5 | 75.49 | 77.17 | **73.87** |
| | | All | **90.75** | **94.18** | 92.74 | 95.67 | 77.43 | 82.41 | 73.01 |
| Enron$_{T}$ | SVM | Net | 83.54 | 90.44 | 89.7 | 91.19 | 40.8 | 43.0 | 38.82 |
| | | Lexical | 91.02 | 94.76 | 93.61 | 95.94 | 68.48 | 73.94 | 63.76 |
| | | All | **91.32** | **94.94** | **93.76** | 96.14 | **69.58** | 75.24 | **64.71** |
| | Extra-Trees | Net | 81.56 | 88.97 | 85.69 | 92.52 | 43.77 | 54.4 | 36.62 |
| | | Lexical | 89.69 | 93.97 | 92.62 | 95.36 | 64.29 | 70.36 | 59.18 |
| | | All | 89.82 | 94.03 | 92.33 | 95.79 | 65.5 | 73.29 | 59.21 |
| | Neural-net | Net | 83.5 | 90.42 | 89.7 | 91.15 | 40.56 | 42.67 | 38.64 |
| | | Lexical | 91.02 | 94.75 | 93.32 | 96.22 | 69.04 | 75.9 | 63.32 |
| | | All | 90.85 | 94.62 | 92.72 | **96.60** | 69.35 | **78.5** | 62.11 |
| Enron$_{\cap A}$ | SVM | Net | 84.79 | 91.13 | 88.57 | 93.85 | 46.56 | 56.41 | 39.64 |
| | | Lexical | 94.28 | 96.75 | 96.59 | 96.92 | 75.95 | 76.92 | 75.0 |
| | | All | 94.43 | 96.84 | **96.76** | 96.92 | 76.43 | 76.92 | 75.95 |
| | Extra-Trees | Net | 87.05 | 92.64 | 92.32 | 92.96 | 46.25 | 47.44 | 45.12 |
| | | Lexical | 94.28 | 96.76 | **96.76** | 96.76 | 75.64 | 75.64 | 75.64 |
| | | All | 92.92 | 95.94 | 94.71 | 97.2 | 72.51 | 79.49 | 66.67 |
| | Neural-net | Net | 83.28 | 90.1 | 86.18 | 94.39 | 46.38 | 61.54 | 37.21 |
| | | Lexical | 94.28 | 96.72 | 95.56 | 97.9 | 77.65 | **84.62** | 71.74 |
| | | All | **95.18** | **97.25** | 96.59 | **97.92** | **80.49** | **84.62** | **76.74** |

Table 11.1: Results of models on Enron$_{\cup}$ , Enron$_{T}$, and Enron$_{\cap A}$. Trained on the corresponding train set and tested on the corresponding development set.

.

### 11.2.1 Performance on Enron$_T$

In the previous section, we have conducted experiments using different Enron datasets in which we train on the corresponding train set and evaluate on the corresponding development set. Therefore, we do not really know what the conclusion is with respect to a single development set. In this subsection, we evaluate on a single datasets using models trained on different datasets.

We select best models from Table 11.1 which are trained in the corresponding train set and tested on the corresponding development set, then, we test them on Enron$_{Tdev}$. We choose Enron$_T$ because it is the only dataset that maintains the thread structure and we are interested in modeling the thread structure. We will experiment with modeling the thread structure in the following chapter (chapter 12).

Table 11.2 shows the results for the best models from Table 11.1 tested on Enron$_T$. We observe that the best model is trained on Enron$_T$ based on all main scores: accuracy, Business F-1, and Personal F-1.

| | | | | Business | | | Personal | | |
|---|---|---|---|---|---|---|---|---|---|
| Train | Test | Model | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| Enron$_{\cap A}$ | Enron$_T$ | NN | 90.85 | 94.66 | 93.42 | 95.93 | 68.07 | 73.94 | 63.06 |
| Enron$_{\cup}$ | Enron$_T$ | SVM | 89.77 | 93.94 | 91.29 | **96.75** | 67.31 | **79.8** | 58.19 |
| Enron$_T$ | Enron$_T$ | SVM | **91.32** | **94.94** | **93.76** | 96.14 | **69.58** | 75.24 | **64.71** |

Table 11.2: Evaluating best models from Table 11.1 on Enron$_T$. All models use all features (both lexical and network). Trained on the corresponding train set and tested on the corresponding test set.

## 11.3 Cross-corpora Results

In this section, we present results for cross-corpora experiments on the social network modeling for emails; we train on Enron and test on Avocado.

For evaluating the cross-corpora performance, we test on Avocado$_{T\ dev}$ using different models from the previous section which are trained on Enron. We tune these models on their corresponding development set (not on Avocado). Note that Avocado$_{T\ dev}$ has a lower personal email ratio than all Enron sets. Table 11.3 shows the cross-corpora results.

We observe that the performance in the inter-corpora setting is lower than the performance in the intra-corpus setting for both social network and lexical features. All models beat the random baseline on the personal F-1 score, fewer models beat the random baseline on the business F-1 score, and only three models beat all the baselines on business F-1: SVM with all features trained on Enron$_T$, Neural Net with lexical features on Enron$_T$ and SVM with all features trained on Enron$_{\cap A}$. Similar to the intra-corpus setting, adding network features improves the personal F-1 score (except neural nets on Enron$_T$) and the best performance is achieved by training on Enron$_{\cap A}$. Models trained on Enron$_{\cup}$ with lexical features only tend to over-predict personal emails since that they have relatively higher personal recall than models trained on other datasets.

|  |  |  |  | Business |  |  | Personal |  |  |
|---|---|---|---|---|---|---|---|---|---|
| Trained on | Classifier | features | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| Enron$_{\cup\ tr}$ | SVM | Net | 87.01 | 92.81 | 91.75 | 93.89 | 32.88 | 36.84 | 29.68 |
|  |  | Lexical | 81.29 | 88.73 | 80.64 | **98.63** | 44.87 | **88.16** | 30.09 |
|  |  | All | 86.55 | 92.2 | 87.02 | 98.04 | 51.17 | 81.58 | 37.27 |
|  | Extra-Trees | Net | 89.24 | 94.16 | 94.94 | 93.39 | 31.73 | 28.95 | 35.11 |
|  |  | Lexical | 86.06 | 91.9 | 86.57 | 97.94 | 50.0 | 80.7 | 36.22 |
|  |  | All | 89.89 | 94.28 | 91.17 | 97.6 | 56.59 | 76.32 | 44.96 |
|  | Neural-net | Net | 79.05 | 87.68 | 81.55 | 94.8 | 30.26 | 52.63 | 21.24 |
|  |  | Lexical | 87.69 | 92.91 | 88.27 | 98.07 | 53.37 | 81.58 | 39.66 |
|  |  | All | 87.80 | 92.96 | 88.10 | 98.38 | 54.52 | 84.65 | 40.21 |
| Enron$_{T\ tr}$ | SVM | Net | 81.06 | 89.03 | 84.08 | 94.59 | 30.94 | 49.12 | 22.58 |
|  |  | Lexical | 91.44 | 95.21 | 93.16 | 97.36 | 59.64 | 73.25 | 50.3 |
|  |  | All | **92.35** | **95.76** | 94.61 | 96.94 | 60.7 | 68.42 | **54.55** |
|  | Extra-Trees | Net | 89.2 | 94.15 | 95.02 | 93.28 | 30.66 | 27.63 | 34.43 |
|  |  | Lexical | 90.15 | 94.47 | 92.08 | 96.99 | 55.02 | 69.74 | 45.43 |
|  |  | All | 90.91 | 94.92 | 92.87 | 97.05 | 57.14 | 70.18 | 48.19 |
|  | Neural-net | Net | 90.11 | 94.76 | 97.89 | 91.83 | 12.12 | 7.89 | 26.09 |
|  |  | Lexical | 91.74 | 95.4 | 93.78 | 97.08 | 59.48 | 70.18 | 51.61 |
|  |  | All | 90.68 | 94.76 | 92.29 | 97.38 | 57.73 | 73.68 | 47.46 |
| Enron$_{\cap A\ tr}$ | SVM | Net | 87.31 | 92.99 | 92.08 | 93.91 | 33.4 | 36.84 | 30.55 |
|  |  | Lexical | 91.17 | 95.05 | 92.83 | 97.39 | 59.05 | 73.68 | 49.27 |
|  |  | All | 92.01 | 95.54 | 93.7 | 97.46 | **61.57** | 74.12 | 52.65 |
|  | Extra-Trees | Net | 89.58 | 94.43 | **96.68** | 92.28 | 19.35 | 14.47 | 29.2 |
|  |  | Lexical | 91.14 | 95.07 | 93.49 | 96.7 | 56.34 | 66.23 | 49.03 |
|  |  | All | 90.04 | 94.37 | 91.46 | 97.48 | 56.53 | 75.0 | 45.36 |
|  | Neural-net | Net | 84.89 | 91.53 | 89.34 | 93.82 | 30.12 | 37.72 | 25.07 |
|  |  | Lexical | 89.85 | 94.24 | 90.96 | 97.77 | 57.05 | 78.07 | 44.95 |
|  |  | All | 91.14 | 95.05 | 93.12 | 97.06 | 57.76 | 70.18 | 49.08 |

Table 11.3: Results of different models tested on Avocado$_{T\ dev}$.

| Top Personal Social Network Features | | Top Business Social Network Features | |
|---|---|---|---|
| D_recipient_degree_centrality_min | 9.68 | D_recipient_degree_centrality_max | -10.19 |
| U_recipient_w_degree_avg | 8.33 | D_recipient_degree_centrality_avg | -10.15 |
| U_#common_neighbors_norm_triangle | 7.83 | U_sender_#triangles | -9.87 |
| D_recipient_hub_score_max | 5.48 | U_recipient_w_degree_max | -5.77 |
| D_recipient_w_in_degree_min | 5.32 | U_recipient_w_between_centrality_max | -5.58 |
| D_recipient_w_out_degree_max | 5.19 | D_recipient_w_in_degree_avg | -5.28 |
| U_#common_neighbors | 4.98 | #recipients | -5.09 |
| U_recipient_w_between_centrality_avg | 4.95 | U_recipient_w_degree_min | -5.08 |
| U_recipient_degree_centrality_max | 4.66 | D_recipient_out_degree_avg | -4.35 |
| U_recipient_degree_max | 4.66 | D_B_email_hub | -4.18 |
| U_recipient_between_centrality_avg | 4.63 | D_sender_degree_centrality | -3.97 |
| D_recipient_w_in_degree_max | 4.56 | D_recipient_hub_score_avg | -3.64 |
| D_recipient_auth_score_avg | 4.17 | U_Jaccard | -3.55 |
| D_recipient_w_out_degree_avg | 4.11 | D_sender_eigenvector_centrality | -3.29 |
| D_B_email_page_rank | 3.57 | D_sender_indegree_centrality | -3.27 |
| D_recipient_eigenvector_centrality_min | 3.07 | U_recipient_eigenvector_centrality_max | -3.03 |
| D_sender_out_degree | 2.94 | D_recipient_in_degree_min | -2.78 |
| D_recipient_between_centrality_avg | 2.83 | D_recipient_in_degree_centrality_min | -2.78 |
| D_B_email_eigenvector_centrality | 2.66 | D_recipient_out_degree_centrality_avg | -2.49 |
| D_sender_w_in_degree | 2.62 | D_recipient_out_degree_max | -2.43 |

Table 11.4: Post-hoc analysis of social network features. We show the Top 20 **net** feature weights for the business and personal classes. "U/D" denotes that the feature is extracted from the undirected/directed user graphs, respectively. "W" denotes that the graph is weighted. "B" denotes that the feature is extracted from the bipartite user-email graph. Absence of "B" denotes that the feature is extracted from the user graph.

## 11.4 Post-hoc Analysis

In this section, we perform a post-hoc analysis to study which social network features help more in distinguishing business and personal emails.

Instead of performing an exhaustive feature ablation analysis, we follow the method presented by Guyon *et al.* (2002) of inspecting feature weights. This approach requires a linear model as the weights assigned for each feature will denote the strength and direction of their relation with the class that is being predicted. To compute the class weights, we use an SVM classifier with a linear kernel using **net** features only; then, we calculate the feature weights which denote the feature importance. We train on the Enron$_\cup$ dataset as models trained on this dataset have relatively the highest performance when using the social network (net) features only. Note that the linear SVM classifier in this section performs slightly lower than the correspondent classifier in Table 11.1 which has an RBF kernel.

Table 11.4 shows the top 20 personal and business weighted features in the trained linear SVM model using **net** features only. Note that for features involving the pair of the sender and recipient, we compute these numbers for every pair of sender and recipient; then, we average the numbers.

We observe that most of the top weighted features are extracted from the user graphs (both directed and undirected). In contrast, only a few features are extracted from the bipartite user-email graph. This indicates that in general features extracted from the bipartite user-email graph are less informative for distinguishing business and personal emails. Additionally, most of the top features involve the recipient more than the sender or the pair of sender and recipient.

The top personal **net** features include: *U_#common_neighbors_norm_triangle* and *U_#common_neighbors*. The former is the number of common neighbors in the undirected user graph normalized by the number of triangles for the sender, whereas the latter is the non-normalized version. *U_#common_neighbors_norm_triangle* is directly proportional to the number of common neighbors between the sender and recipient and inversely proportional to the number of triangles for the sender. These features indicate that if a pair of a sender and recipient share many neighbors and the sender do not have many transitive relations, it is more likely that an email they exchange is personal.

On the other hand, the top business **net** features include: *U_sender_#triangles*. Which means that a high number of triangles for the sender indicates that the email is more likely to be a business email. Additionally, an interesting top business feature is the Jaccard similarity (*U_Jaccard*), which is the number of common neighbors normalized by the total neighbors for the sender and recipient. Unlike *U_#common_neighbors_norm_triangle*, this feature takes into account the neighbors for the recipient who are not common neighbors with the sender. It indicates that if the sender and the recipient have a large number of common neighbors, it is more likely that the email is business. These numbers agree with the analysis we have shown in chapter 6 that people who exchange business emails tend to cluster more often than those who exchange personal emails; then, they form more triangles. Another interesting feature in the top business **net** features is *#recipients*. It indicates that an email is more likely to be business if it has a high number of recipients.

We observe that different centrality scores capture different things. Note that in case that the email has multiple recipients, we include the average, max, and min scores as distinct features. Sometimes the average, max, and min scores are indicators for opposite classes. For instance, the top personal features include *D_recipient_degree_centrality_min*, *U_recipient_w_degree_avg*, whereas the top business features include *D_recipient_degree_centrality_max* and *D_recipient_degree_centrality_avg*. Note that the total degree in the directed user graph includes both the incoming and outgoing edges. Then, a higher number for the total degree might be caused by either a higher number for the out or in degree or both.

In addition, the features that use the edge weight (amount of emails exchanged) capture different things than features that do not. An interesting observation is that business emails tend to be exchanged between people who have many neighbors (high degree), while the personal emails are sent to people who respond more. For instance, the top personal features include *D_recipient_w_out_degree_avg* and *D_recipient_w_out_degree_max*. These features measure the amount of emails sent by the recipient. On the other hand, the business top features include unweighted features that measure the number of distinct neighbors but not the amount of emails exchanged such as: *D_recipient_out_degree_avg*, *D_recipient_in_degree_min*, *D_recipient_in_degree_centrality_min*, *D_recipient_out_degree_centrality_avg*, *D_recipient_out_degree_max*, and *D_recipient_out_degree_centrality_avg*.

## 11.5 Conclusion

We presented in this chapter experiments on email classification into business and personal using two corpora: Enron and Avocado. We train on Enron and test on both corpora. We incorporate information from the underlying social network of email communication for this task. The experiments show that the social network features extracted from different graphs, including the user and bipartite graph of users and emails help in improving the classification performance. They also help in the cross-corpora setting when we train on a company and evaluate on another. In addition, we experiment with various machine learning classifiers and train on different Enron datasets each labeled differently. We also conduct an analysis on the social network features by inspecting the feature importance using their weights. Our analysis agree with the observation we present in chapter 6.

# Chapter 12

# Modeling Thread Structure

In the previous chapters, we have presented models on individual emails modeling without taking into account other emails in the same thread. However, the class of emails in the same thread are related; in fact, we observe that only 2.8% of threads in our Enron data set contain both "Personal" and "Business" emails.

In this chapter, we discuss our approach of incorporating information from other emails in the same thread in order to improve the classification. We try two methods: first, a simple approach that re-predicts email labels based on the majority of the predicted email labels in the same thread; second, by using sequential models on threads, namely, LSTMs. Here, we only test on two datasets: Enron$_T$ and Avocado$_T$, since these two sets have complete labels for all emails in the threads.

## 12.1 Majority Vote

We apply a very simple technique for modeling the thread structure. First, we select the models in chapter 11 with the highest personal F-1 score. For Enron, we select the best models for each dataset in Table 11.1, i.e., Enron$_\cup$ , Enron$_T$, and Enron$_{\cap A}$; then, we test these models on Enron$_T$. For Avocado, we select the best model on Avocado$_T$ among all training sets from Table 11.3. We

|  |  |  |  | Business |  |  | Personal |  |  |
|---|---|---|---|---|---|---|---|---|---|
| Train | Test | Model | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| Enron$_{\cap A}$ | Enron$_T$ | NN | 90.85 | 94.66 | 93.42 | 95.93 | 68.07 | 73.94 | 63.06 |
|  |  | Majority | 91.23 | 94.85 | 93.02 | 96.76 | 70.52 | 79.48 | 63.38 |
| Enron$_{\cup}$ | Enron$_T$ | SVM | 89.77 | 93.94 | 91.29 | 96.75 | 67.31 | 79.8 | 58.19 |
|  |  | Majority | 89.56 | 93.76 | 90.45 | 97.34 | 67.9 | **83.71** | 57.11 |
| Enron$_T$ | Enron$_T$ | SVM | 91.32 | 94.94 | **93.76** | 96.14 | 69.58 | 75.24 | **64.71** |
|  |  | Majority | **91.49** | **95.0** | 93.12 | **96.96** | **71.47** | 80.78 | 64.08 |
| Enron$_{\cap A}$ | Avocado$_T$ | SVM | **92.01** | **95.54** | **93.7** | 97.46 | 61.57 | 74.12 | **52.65** |
|  |  | Majority | 91.17 | 95.01 | 91.87 | **98.36** | **62.11** | **83.77** | 49.35 |

Table 12.1: Applying thread majority vote on best models from tables 11.1 and 11.3. Model column indicates models used for training; “Majority” indicates results after applying majority vote on the predicted labels. All models use all features (both lexical and network). We use the corresponding training set for training, and the corresponding development set for testing.

test on Enron$_T$ as it is the only dataset that maintains the thread structure.

We first predict individual emails without looking at other emails in the same thread using the best models from the previous chapter (chapter 11). Then, we compute the majority vote of all emails in the same thread by assigning the majority label to each email in the thread, in case that there is no majority (i.e., the numbers of predicted business and personal labels are the same), we consider “personal” to be the majority label.

Table 12.1 shows the results of applying majority vote on the best models in chapter 11. In all cases, this simple method gives an improvement on the personal F-1 score. However, in some cases, we observe that we have lower business F-1 scores. The highest gain on all scores is when training and testing on Enron$_T$.

|  |  |  | Business |  |  | Personal |  |  |
|---|---|---|---|---|---|---|---|---|
| Test Set | Method | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| Enron$_{T\ dev}$ | LSTMs | **91.88** | **95.26** | **94.11** | 96.45 | 71.49 | 77.2 | **66.57** |
|  | LSTMs + Maj. Vote | 91.71 | 95.14 | 93.61 | **96.73** | **71.58** | **79.15** | 65.32 |
| Avocado$_{T\ dev}$ | LSTMs | **93.67** | **96.5** | **95.48** | 97.54 | 67.06 | 74.56 | **60.93** |
|  | LSTMs + Maj. Vote | 93.64 | 96.47 | 95.07 | **97.91** | **68.06** | 78.51 | 60.07 |

Table 12.2: Sequential modeling of threads using LSTMs and LSTMs with majority vote. All models are trained on Enron$_{T\ tr}$.

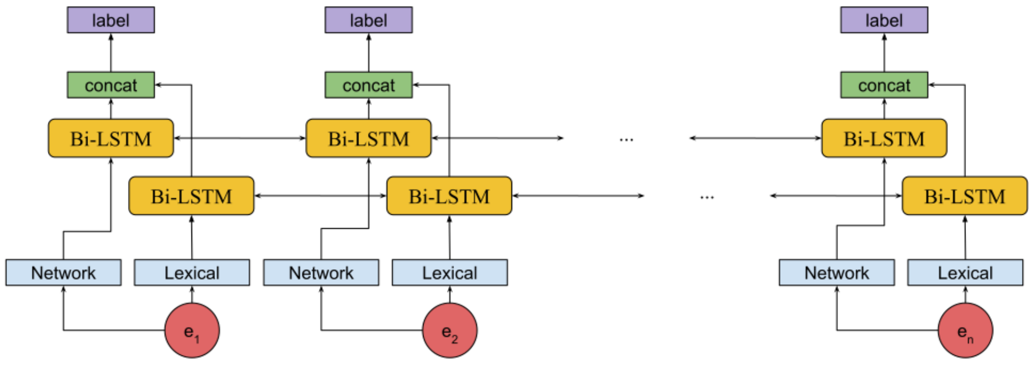


Figure 12.1: Two concatenated BiLSTMs for thread sequential modeling; one for lexical features and the other for social network features.

## 12.2 Thread Sequential Modeling Using LSTMs

In the previous section we show a simple method for modeling the thread structure. Another way to model the thread structure is by using sequential models. In this section, we apply Long Short Term Memories (LSTMs) (Hochreiter and Schmidhuber, 1997) over sequence of emails in threads to model the thread structure. The input to the LSTM models is a sequence of emails in a thread. Each email has two sets of features: lexical features, and social network features. We concatenate two Bidirectional LSTMs (BiLSTMs), one for lexical features and the other for the social network features. For lexical features, we average the FastText word embeddings for words in the email content (chapter 10). For social network features, we use the social network features presented in chapter 11. Figure 12.1 illustrates the model architecture. For the implementation of the LSTM models, We use the Keras framework (Chollet and others, 2015). We use the Adam optimizer (Kingma and Ba, 2014) with the default parameters as provided by Keras. We experiment with different settings for various hyper-parameters. For the hidden dimensions, we test values of 2, 5, 10, 20, 50. We use dropout (Srivastava *et al.*, 2014) on the model input and the recurrent input with rates: $0 - 100\%$ with an increment of $10\%$. We try different values for the two LSTMs (i.e., the lexical and the network). We use a batch size of 16 with 50 epochs. We experiment with both binary cross-entropy and hinge loss as the loss function. For the non-linearity activation on the output layer, we use the *sigmoid* function when using the binary cross-entropy loss, and *tanh* with hinge loss.

**Majority of the thread** Similar to previous models that predict labels for emails individually, we can apply the majority vote on the output of LSTMs such that all emails in a given thread are assigned a single label (the majority in the thread). We first predict emails using LSTMs, Then, we compute the majority vote of all emails in the same thread by assigning the majority label to each email in the thread, in case that there is no majority (i.e., the numbers of predicted business and personal labels are the same), we consider “Personal” to be the majority label. We apply the majority vote method on the labels predicted by the LSTM models.

Table 12.2 shows the performance of LSTMs and LSTMs with majority vote. The results show that LSTMs models perform better than models trained on individual emails on both the personal and business F-1 scores. We observe that applying majority to LSTM models increases the personal F-1 score but decreases the business F-1. The results show that sequential modeling was really helpful for Avocado as the relative improvement on Avocado is much higher than on Enron. We believe this is due to the fact that since Avocado is another corpus, a classifier that models emails individually will predict the class for many emails with less certainty on Avocado more than on Enron. Then, sequential modeling for threads will help when there is uncertainty in some emails by incorporating information from other emails in the same threads where it is easier to assign the correct class; given the fact that most emails in the same thread belong to the same class.

## 12.3 Conclusion

We have shown in this chapter two techniques for modeling the thread structure: majority vote for emails in the thread; and sequential modeling of threads using LSTMs. Both techniques improve the classification performance over models that do not incorporate the thread structure. Combining both techniques improves the performance further when considering the optimization metric, i.e., personal F-1.

# Chapter 13

# Alternative Social Network Modeling Approaches

We have presented so far in this part our approach for modeling the task of email classification into “business” and “personal”. We have proposed social network based features extracted from different graph representations for the underlying communication network for the emails.

In this chapter, we present alternative, state-of-the-art social network modeling approaches that have been proposed in the literature. Particularly, we investigate using graph neural networks for the task of email classification. Recently, graph neural networks have gained a lot of attention in the research community. Network embedding aims at representing network nodes as low-dimensional vector representations, preserving both network topology structure and node content information. GraphSAGE (Hamilton *et al.*, 2017a) is a state-of-the-art inductive model for learning node embeddings for different tasks including node classification. We use GraphSAGE in this chapter for modeling the task of email classification into “business” and “personal”. In addition to the ordinary implementation for GraphSAGE, we extend it to the bipartite graphs.

## 13.1 GraphSAGE

In this section, we experiment with alternative approaches for social network modeling. Recently, graph neural networks have gained a lot of attention in the research community. Network embedding aims at representing network nodes as low-dimensional vector representations, preserving both network topology structure and node content information. GraphSAGE (Hamilton *et al.*, 2017a) is a recent state-of-the-art inductive model for learning node embeddings for different tasks including node classification. It learns an embedding for a given node by aggregating information from its neighboring nodes and from attributes of the node. It is designed for homogeneous graphs where nodes belong to one type. Thus, we construct a graph which has only emails as nodes (we do not construct a graph with people as nodes since we also need access to the lexical content for GraphSAGE). In this graph, nodes represent emails and edges link emails if they share a certain percentage of participants – we do not distinguish between senders and recipients as participants. Then, we feed the GraphSAGE supervised model with this graph of emails with their corresponding labels, and furthermore, we use the lexical features described in section 10.1 as node attributes. Particularly, we use *FastText* embeddings for emails as the node attributes.

In addition to the GraphSAGE hyper-parameters, we try different values for the threshold of the percentage of participants. The hyper-parameters include *batch size*, *number of iterations*, *learning rate*. We use the PyTorch implementation for GraphSAGE. [1] For the *batch size*, we experiment with values $32, 64, 128, 256$. For *number of iterations* we use values between $100 - 700$ with an increment of 100. For *learning rate*, we try values of $0.5, 0.6, 0.7, 0.8$.

[1] `https://github.com/williamleif/graphsage-simple/`

## 13.2 GraphSAGE with Bipartite Graph

GraphSAGE is not designed to deal with the heterogeneous network induced by email exchange that includes emails and participants. Recall that GraphSAGE learns an embedding for a given node by aggregating information from its neighboring nodes and from the attributes of that node. Particularly, it learns $K$ different aggregator functions ($f_1, ..., f_k$) which are used to propagate information between $K$ different layers of the model or "search depths". At the first iteration, or search depth, nodes aggregate information from their immediate neighbors. At depth K, nodes aggregate information from neighbors at depth K.

We can simply extend GraphSAGE to bipartite graphs as follows. We construct a bipartite graph of users and emails as discussed in section 5.2. Then, we feed this graph to a version of GraphSAGE which we modified such that we have different aggregates for users and emails. At odd aggregator layers, we process emails and we sample their user neighbors in the bipartite graph; and at even layers, we process users and sample their neighboring emails. For emails, we use lexical features to represent them. For users, we use the network features extracted from the corresponding node in the user graphs as discussed in chapter 7. We refer to this method as *GraphSAGE-BiP*. Because of the extension to bipartite graphs and the use of our own features as attributes for the user nodes, *GraphSAGE-BiP* represents a contribution of this thesis. We set $K = 4$ in our experiments. We use the PyTorch implementation for GraphSAGE. [2] For other hyper-parameters, we use the same values discussed in the previous section.

[2]`https://github.com/williamleif/graphsage-simple/`

## 13.3 Experiments and Results

In this section, we present experiments on email classification into “business” and “personal” using GraphSAGE and our extension GraphSAGE-BiP.

We first show experiments using different threshold values for the email graph for GraphSAGE, then, we show experiments on GrphSAGE-BiP.

We are optimizing the personal F-1 score. We train on Enron and test on Enron and Avocado. We use the Enron$_T$ dataset for Enron and Avocado$_T$ for Avocado (section 4.2 and section 4.3).

### 13.3.1 Participant Threshold Values

In this section, we report the results for different threshold values with their best performance when GraphSAGE model hyper-parameters are tuned. We try threshold values with an increment of 5.

Figure 13.1 shows the Personal F-1 score for different threshold values for the number of common participants. We observe that the peak performance is when we link emails only when they share $30\%$ of participants. The personal F-1 score increases until it reaches $15\%$ as it drops when the threshold increases from $15\%$ to $20\%$, then, it keeps increasing until it reaches its peak at $30\%$, then, it starts to drop.

Table 13.1 shows the results for GraphSAGE (GS) and GraphSAGE-BiP (GS-BiP). We observe that in the intra-corpus setting, GS-BiP performs better than GS in all scores. However, in the cross-corpora setting, we observe that the personal scores (except the precision) are lower in GraphSAGE-BiP than GraphSAGE.

### 13.3.2 Number of Neighbors for GraphSAGE-BiP

We try different numbers of neighbors to sample for GraphSAGE-BiP. Note that each GraphSAGE layer has two types of aggregates: users, and emails. We have two layers (aggregators) for each

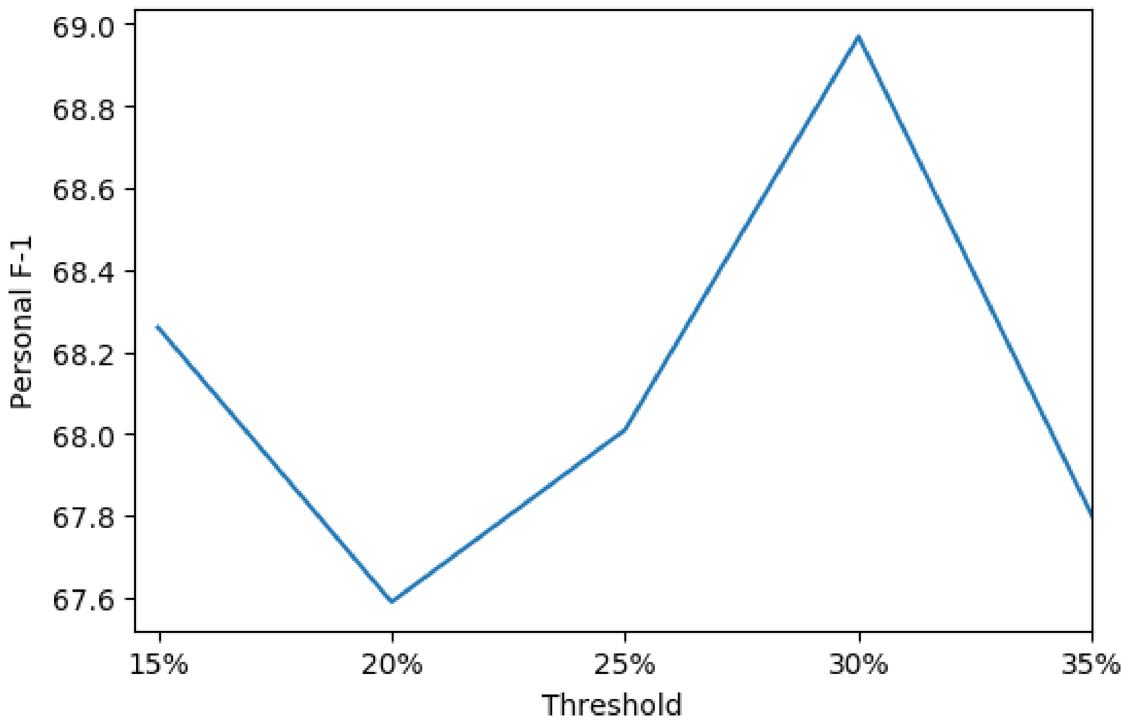


Figure 13.1: Personal F-1 scores for different threshold values for the number of common participants in the email graph for GraphSAGE

node type. Thus, in total, we have four numbers:

1. *n1emails*: number of email neighbors in the first layer to be sampled. These neighbors are sampled from the directly connected users to the emails in the bipartite graph.

2. *n1users*: number of user neighbors in the first layer. These are the emails connected to the users sampled from *n1emails*.

3. *n2emails*: number of email neighbors in the second layer to be sampled. These neighbors are sampled from the users connected to the emails sampled from *n1users*.

4. *n2users*: number of user neighbors in the second layer. These are the emails connected to the users sampled from *n2emails*.

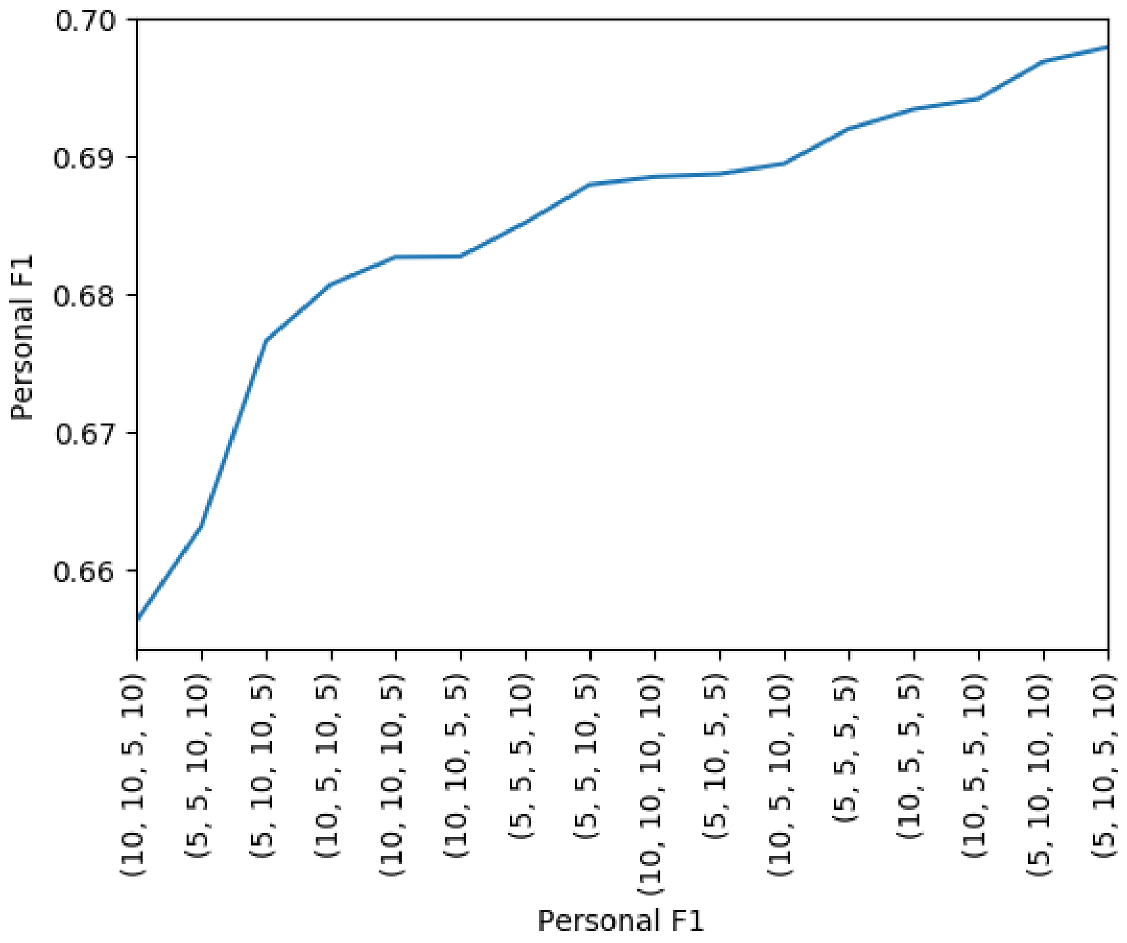


Figure 13.2: Personal F-1 scores for different values for the number of neighbors at each layer and the node type for GS-BiP on Enron$_{T\ dev}$. The x-axis labels represent the tuple (*n1emails*, *n1users*, *n2emails*, *n2users*) different values.

Figure 13.2 shows the Personal F-1 score for different values for the tuple (*n1emails*, *n1users*, *n2emails*, *n2users*). We observe that, in general, sampling more emails than users gives better performance. The best performance is when we sample 5 users and 10 emails at each layer.

### 13.3.3 Results

We select the best models from the previous sections where we train on Enron$_{T\ tr}$ and tune on Enron$_{T\ dev}$. Then, we apply them to Avocado$_{T\ dev}$. Table 13.1 shows the results for both Graph-

|  |  |  | Business |  |  | Personal |  |  |
|---|---|---|---|---|---|---|---|---|
| Test Set | Model | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| Enron$_{T\ dev}$ | GS | 91.10 | 94.81 | 93.56 | 96.09 | 68.97 | 74.92 | 63.89 |
|  | GS-BiP | 91.41 | 94.99 | 93.86 | 96.15 | 69.79 | 75.24 | 65.07 |
| Avocado$_{T\ dev}$ | GS | 90.15 | 94.41 | 91.13 | 97.95 | 58.33 | 79.82 | 45.96 |
|  | GS-BiP | 91.10 | 95.09 | 93.12 | 97.02 | 57.50 | 69.74 | 48.92 |

Table 13.1: Results for GraphSAGE and GraphSAGE with a bipartite graph (GS-BiP). In all experiments, we train on Enron.

SAGE (GS) and GraphSAGE with a bipartite graph (GP-BiP). We observe that on Enron, GS-BiP outperforms GS on all scores. In the cross-corpora setting (i.e., testing on Avocado), the Personal F-1 score for GS-BiP is worse than for GS. However, the accuracy and the Business F-1 is better for GS-BiP than for GS. Both models i.e., GS and GS-BiP, do not outperform our best models discussed in the previous chapters.

## 13.4 Conclusion

In this chapter, we have shown alternative approaches for modeling the social network of emails using a state-of-the-art graph embedding model, GraphSAGE. We also show our extension of GraphSAGE to bipartite graphs. The results show that both models improve over the performance of an approach based on textual information only. Our extension GS-BiP outperforms the ordinary GS model in the intra-corpus setting. However, both approaches: GS and GS-BiP, perform worse than our best models discussed in the previous chapters.

# Chapter 14

# Summary, Additional Evaluations, and Conclusion

In this chapter, we show the summary of results from previous chapters and we analyze the statistical significance of the best models discussed in the previous chapters. We also evaluate them on blind test sets that we do not optimize on. In addition, we conduct an analysis on the erroneous cases made by the machine learning classifier. We also show performance of our models on another Enron dataset, the Sheffield dataset (subsection 4.2.2).

We start the chapter by summarizing the results and statistical significance of models presented in the previous chapters in section 14.1. Then, we show error analysis results in section 14.2. We show the results of evaluating the best models on blind test sets in section 14.3. Then, we evaluate some of our models on the Sheffield dataset in section 14.4. Finally, we conclude the chapter and the part in section 14.5

## 14.1 Summary of the Results and Statistical Significance

In this section, we show the summary of the results for the email classification task into business and personal. We show the corresponding best models from previous chapters for each email corpus (i.e., Enron and Avocado). Also, we show results for statistical significance tests for different models. To determine whether the performance improvement of different classifiers over others is statistically significant, we use the non-parametric Wilcoxon Signed-Rank Test (Sidney, 1957) on pairs of the Personal F-1 scores of different classifiers using 10 fold-cross validation runs on Enron.

| Classifier | Accuracy | Business F-1 | Personal F-1 | Reference Table |
|---|---|---|---|---|
| Random Baseline | 76.9 | 86.7 | 13.3 | Table 9.2 page 92 |
| GraphSAGE | 91.2 | 94.8 | 69.0 | Table 13.1 page 125 |
| GraphSAGE-BiP | 91.4 | 95.0 | 69.8 | Table 13.1 page 125 |
| SVM-net | 83.5 | 90.4 | 40.8 | Table 11.1 page 106 |
| SVM-lex | 91.0 | 94.76 | 68.5 | Table 11.1 page 106 |
| SVM-all | 91.3 | 94.9 | 69.6 | Table 11.1 page 106 |
| SVM-all + Majority | 91.5 | 95.0 | 71.5 | Table 12.1 page 115 |
| LSTMs | **91.9** | **95.3** | 71.5 | Table 12.2 page 116 |
| LSTMs + Majority | 91.7 | 95.1 | **71.6** | Table 12.2 page 116 |

Table 14.1: Summary of results on Enron. All models are trained on Enron$_{T\ tr}$ and evaluated on Enron$_{T\ dev}$.

### 14.1.1 Enron

Table 14.1 shows the summary of models from previous chapters on Enron. All models are trained on Enron$_{T\ tr}$ and evaluated on Enron$_{T\ dev}$. We observe that all models beat the random baseline on both evaluation scores. In addition, models that incorporate the social network information perform better than models that use lexical or social network features alone.

To analyze the statistical significance of incorporating the social network information, we compare the performance of *SVM-all*, *GraphSAGE*, and *GraphSAGE-BiP* with *SVM-lex*. The increase in the performance of adding the social network information over the model that uses lexical features only (*SVM-lex*) is statistically significant at $p < 0.01$ for all three models. Moreover, the gain in the performance of GraphSAGE-BiP and SVM-all over GraphSAGE is statistically significant ($p < 0.01$). However, the gain of GraphSAGE-BiP over SVM-all is not statistically significant ($p > 0.05$).

We analyze the statistical significance of modeling the thread structure by studying the gain in the performance when using majority vote on SVM-all and when using sequential models (i.e., LSTMs). The analysis shows that the performance improvement of using LSTMs and the improvement by applying the majority vote on SVM-all and over SVM-all alone is statistically significant ($p < 0.01$). However, the gain of LSTMs + Majority over the ordinary LSTM and SVM-all + Majority is not statistically significant. Finally, the performance of our best model LSTMs + Majority is statistically significant ($p < 0.01$) in comparison with GraphSAGE and GraphSAGE-BiP.

### 14.1.2 Avocado

Table 14.2 shows the summary of models from the previous chapters evaluated on Avocado$_{T\ dev}$. Similar to evaluating on Enron, we observe that adding the social network features to lexical features (svm-all) improves the classification performance over using the lexical features alone (svm-lex). However, unlike Enron, GraphSAGE and GraphSAGE-BiP perform worse than SVM-lex. Additionally, the relative gain in the performance when modeling the thread structure using LSTMs on Avocado is much larger than on Enron.

To perform a statistical significance test on Avocado, we run 10 iterations in which we split the

| Classifier | Accuracy | Business F-1 | Personal F-1 | Reference Table |
|---|---|---|---|---|
| Random Baseline | 83.9 | 91.2 | 8.8 | Table 9.2 page 92 |
| GraphSAGE † | 90.2 | 94.4 | 58.3 | Table 13.1 page 125 |
| GraphSAGE-BiP † | 91.1 | 95.1 | 57.5 | Table 13.1 page 125 |
| SVM-net ‡ | 87.3 | 93.0 | 33.4 | Table 11.3 page 109 |
| SVM-lex ‡ | 91.2 | 95.1 | 59.1 | Table 11.3 page 109 |
| SVM-all ‡ | 92.0 | 95.5 | 61.6 | Table 11.3 page 109 |
| SVM-all + Majority ‡ | 91.2 | 95.0 | 62.1 | Table 12.1 page 115 |
| LSTMs † | **93.7** | **96.5** | 67.1 | Table 12.2 page 116 |
| LSTMs + Majority † | 93.6 | 96.5 | **68.1** | Table 12.2 page 116 |

Table 14.2: Summary of results on Avocado. † indicates that the model is trained on Enron$_{T\ tr}$ and ‡ indicates that it is trained on Enron$_{\cap A\ tr}$. All models are evaluated on Avocado$_{T\ dev}$.

corresponding Enron train set into 10 folds. For each iteration, we use 9 folds for training, and we leave one out (we leave a different fold at different iterations), and we use the same Avocado evaluation set in all iterations ( Avocado$_{T\ dev}$).

Similar to Enron, the performance gain by adding the social network features (SVM-all) over the lexical features alone (SVM-lex) is statistically significant ($p < 0.01$). Also, the improvement of using the majority vote with SVM-all over SVM-all is statistically significant. Unlike Enron, the gain of LSTMs + Majority over the ordinary LSTM and SVM-all + Majority is statistically significant ($p < 0.01$). Finally, the performance of our best model LSTMs + Majority is statistically significant ($p < 0.01$) in comparison with GraphSAGE and GraphSAGE-BiP.

The statistical significance analysis shows that the improvement of adding social network features to lexical features is statistically significant using both email corpora (i.e., Enron and Avocado). Also, the improvement in the personal F-1 score by incorporating the thread structure, whether by using the simple majority vote approach or using sequential models (i.e., LSTMs), is statistically significant. More importantly, the improvement of our best models over the state-of-the-art graph embedding model (i.e., GraphSAGE) is statistically significant.

## 14.2 Error Analysis

In this section, we show error analysis results by studying erroneous cases made by the classifier on the Enron corpus. We use the best model, i.e., LSTMs with the majority vote, trained on $\text{Enron}_{T\ tr}$, and evaluated on $\text{Enron}_{T\ dev}$. We randomly selected 100 emails in which the classifier predicts the wrong class on $\text{Enron}_{T\ dev}$. Then, we manually investigate these emails. The total number of wrongly predicted emails is 193 out of 2,327 (8.29%).

We first study the distribution of labels assigned by different annotators for both correctly and wrongly predicted emails. Recall that for Enron, we use Amazon Mechanical Turk to annotate emails such that each email is labeled by different annotators, and each annotator is asked to assign a numerical value between 1 and 6; with 6 being "cannot determine" and otherwise a larger number indicating that the email is more personal and a smaller number indicating that the email is more business. See section 4.1 on page 30 for more information.

| | Average Final Label | Mean | Standard Deviation | Cannot Determine |
|---|---|---|---|---|
| All | 0.132 | 1.577 | 0.353 | 144 (6.19%) |
| Correctly Predicted | 0.114 | 1.489 | 0.305 | 120 (5.62%) |
| Wrongly Predicted | 0.332 | 2.529 | 0.878 | 24 (12.44%) |

Table 14.3: Difference of final label average, mean and standard deviation of labels assigned by annotators between correctly and wrongly predicted emails. For the final label: 0 and 1 values represent business and personal classes, respectively. The last column, "cannot determine", shows the counts and percentage of emails that at least one annotator says "cannot determine".

Table 14.3 shows the average final label, mean and standard deviation for labels assigned by the annotators. For the final label, after following the procedure discussed in section 4.1, we assign 0 to the business labels and 1 to the personal labels. For the mean and standard deviation, for each email, we first calculate the mean and standard deviation for labels assigned by different annotators after excluding cannot determine (i.e., 6); for example, if three annotators assign the labels: $\{1, 4, 6\}$, we first exclude the label 6, and therefore the mean is $2.5$, and the standard deviation is $1.5$. Then, we compute the average for mean and standard deviation values for all emails. We refer to these scores as "the average annotator mean score" and "the average annotator standard deviation score", respectively. We compute these values for all emails and given if they are correctly or wrongly predicted by the classifier.

We observe that the average final label for the correctly predicted examples is much less than the wrongly predicted ones. Note that the dataset is unbalanced as that more than 80% of emails are business. In addition, the average annotator mean score is less for the correctly predicted emails than the wrongly predicted ones. This indicates that the model tends to wrongly predict the personal emails more often than the business emails. In fact, 66% of the wrongly predicted emails are business but predicted personal, which is much larger than the ratio of personal emails in the dataset (13%). In addition, the personal recall is higher than the personal precision while the opposite for the business class (Table 12.2 page 116). Moreover, Table 14.3 shows that, on average, the standard deviation of email labels is larger for the wrongly classified emails than for the correctly classified ones. This indicates that annotators tend to disagree in the labels when emails are wrongly predicted by the classifier more than when emails are correctly predicted. This is intuitive as we expect that the classifier will be more likely to predict the wrong class for the harder cases in which humans disagree more.

To perform a deeper analysis on the erroneous cases, we assign each email sampled from the wrongly predicted emails to one of the following categories:

- *Gold Standard Error*: After careful manual investigation, we found that the final label given by the annotators is wrong.

- *Hard Cases*: After careful manual investigation, we found it hard to assign a label to a given email. Some cases were not covered in the instructions provided to the annotators. Such cases include: professional social events such as happy hours that include employees of Enron and other companies in the same area; and emails that ask for contact information. Other cases include emails with a very informal tone; and personal emails with many business words/content.

- *Short Emails*: Emails in this category lack enough context to determine their category, or they include attachments that the classifier does not have access to.

- *External Content*: Emails in this category contain external material being shared with other employees, such as online articles.

- *Other business*: Cases in this category include emails exchanged between Enron employees and businesses other than Enron. For instance, an Enron employee purchasing items from another company using their Enron email.

- *Wrong majority:* Emails in this category were correctly classified by the LSTM model, but the majority vote overrides the correct prediction. The majority vote is computed using other labels in the same thread.

- *Non-English emails*: Emails in this category are written in languages other than English.

Table 14.4 shows the distribution of the sampled wrongly classified emails among the 7 categories. For computing the mean and standard deviation, we first exclude any cannot determine

| Error Type | # Cases | Mean | STD | Cannot Det | Bus | Pers |
|---|---|---|---|---|---|---|
| Gold Standard Error | 47 | 2.19 | 0.989 | 7 (14.89%) | 39 (82.98%) | 8 (17.02%) |
| Hard Cases | 15 | 2.29 | 0.685 | 6 (40.0%) | 11 (73.33%) | 4 (26.67%) |
| Short Emails | 15 | 2.95 | 0.701 | 2 (13.33)%) | 7 (46.67%) | 8 (53.33%) |
| External Content | 9 | 2.82 | 1.193 | 0 | 4 (44.44%) | 5 (55.56%) |
| Other Business | 6 | 3.33 | 0.943 | 0 | 0 | 6 (100%) |
| Wrong Majority | 5 | 2.40 | 0.798 | 0 | 3 (60%) | 2 (40%) |
| Non-English emails | 3 | 2.11 | 0.831 | 0 | 3 (100%) | 0 |
| Total | 100 | 2.45 | 0.902 | 15 | 67 | 33 |

Table 14.4: Distribution of the sampled wrongly classified emails among different categories. The mean and STD indicate the mean and standard deviation of the ordinal values for labels assigned by annotators, respectively. Note that we exclude "cannot determine" labels when computing the mean and standard deviation. The column "cannot det" shows the counts and percentage of emails that at least one annotator says "cannot determine". The last columns: "Bus" and "Pers", indicate the gold business and personal label counts (and percentages), respectively.

labels; then, we use the ordinal values for labels assigned by the annotators to compute these numbers.

The numbers show that the largest category is the gold standard error (47%). Many emails in this category have a relatively large disagreement on the labels assigned by different annotators as indicated by the standard deviation for the label values (0.989). After manual investigation, we found that many emails in this category were easily business or personal, but the annotators assigned the wrong category (according to our assessment). Note that we use Amazon Mechanical Turk as the annotation platform, which tends to have a less reliable annotation as indicated by the inter-annotator agreement scores discussed in section 4.4, and previous studies have shown that annotations using Amazon Mechanical Turk tend to be less reliable (Sappelli *et al.*, 2016). We noticed that many annotators always assign the majority class (i.e., "business") for many emails especially when they are long. However, there are some emails in this category that need extra effort to determine the category, including reading other emails in the same thread.

The second most frequent category is “hard cases”. In fact, many emails in this category were assigned “cannot determine” by at least one annotator. Note that the standard deviation is computed after excluding “cannot determine” labels. Emails in this category include emails about social events that are not clear if they take place at Enron or another place. Other examples include ambiguous emails, such as keeping-in-touch, or follow-up emails for offline activities. Also, some emails were labeled business, but the tone used in the email content is very informal, and the content is unclear.

“Short emails” is the third most frequent error category, and it has more than 13% of emails in which annotators assign “cannot determine” label. Emails in this category lack enough context to determine the correct category. They include follow-up emails for offline events and emails with attachments that are not provided in the corpus.

The “external content” category is the fourth in the number of erroneous cases. It has the largest standard deviation for labels assigned by different annotators. Emails in this category include business articles that are not really about Enron, but shared with other colleagues.

The “other business” category is the fifth most frequent case. Emails in this category are labeled “personal” by the human annotators as these emails are not related to the Enron business; examples include emails about purchasing personal items such that the tone in these emails is business, which makes the classifier wrongly predict them as business.

The sixth most frequent case is the “wrong majority” category. Emails in this category were correctly predicted before applying the majority vote (the majority of other emails in the same thread). There is a total of 5 emails in this category, and one of these emails belong to “mix threads“ in which emails have different labels (see subsection 9.2.1 on page 91). In this type of threads, the majority vote does not work as at least one email will be wrongly predicted. The other case is when the email is correctly predicted by the LSTM model, but the majority vote makes it wrong because other emails in the thread are wrongly classified.

"Non-English emails" is the least in the number of cases. It contains emails written in languages other than English. In the sampled erroneous cases, the annotators assigned business labels to all emails in this category, but some of them are not (we found that after translating them). However, we do not expect the machine classifier nor the human annotators to assign the correct class as modeling and annotating non-English emails is outside the scope of our task.

Below we show examples for each category:

**Gold standard error** annotators assigned $\{1, 1, 3\}$ labels; the final label is business.
This email is clearly personal, but the annotators wrongly labeled it business.

**Subject:** Reservation
J,
Do you think that if P & J and the kids went to Guam they would consider leaving
Jo Lynn with the kids and come on to Hong Kong and join up with us?
FYI, I still have small reservations about travelling abroad still close to our international situation.
Perhaps fear of the unknown. Sitting on the fence. Undecided.
K.

**Hard cases** annotators assigned $\{1, 3, 4\}$ labels; the final label is business.
It seems to be a follow-up email for an offline conversation.
Unclear if it is business or personal.

**Subject:** computer
I got everything hooked up , I guess.
You need to straighten out the cords this weekend.

**Short emails** annotators assigned $\{1, 3, 4\}$ labels; the final label is business.

This email includes an image attachment which is not available in the corpus.

**Subject:** OOPS

Beth Cherry

*Attachment: priceless.jpg*

**External Content** annotators assigned $\{1, 1, 4\}$ labels; the final label is business.

This email is sharing an article about a computer virus.

**Subject:** WORST EVER VIRUS - WARNING

WORST EVER VIRUS (CNN announced)

A new virus has just been discovered that has been classified by Microsoft as the most destructive ever!

This virus was discovered yesterday afternoon by McAfee and no vaccine has yet been developed. ... *the shared article continues*

**Business with other companies** annotators assigned $\{4, 4, 4\}$ labels; the final label is personal.

This email is about offering wine business, and it is clearly personal in our case.

**Subject:** wine

Greg–I know you like good wine and I am selling some of my California Cabs.

If you have any interest here is what I have:

’92 Silver Oak Napa Valley 2 bottles

’93 Silver Oak Napa Valley 17 bottles

’94 Silver Oak Napa Valley 3 bottles

*... the list of wines continues*

All of the wine listed above has been stored in a wine locker
at 55 degrees since it was purchased.

**Wrong Majority** There are two emails in this thread;

annotators assigned $\{1, 1, 2\}$ labels to both emails; the final label is business.

This thread is related to a hotel reservation;

it is not specified if the accommodation is for a business or personal trip.

**Predictions** first email is personal; second is business; the majority is personal

**First email** This email was labeled business and predicted personal (before majority);

**Subject:** (forward) Lucci 8oct Denver to Houston—ETKT

Hyatt wasn't available – she got you in the Doubletree.

*quoted second email*

**Second email** This email was labeled business and predicted business (before majority);

**Subject:** Lucci 8oct Denver to Houston—ETKT

AGENT JH/JH BOOKING REF

Y5THCI

LUCCI/PAUL

... *ticket information continues*

**Non-English emails** annotators assigned $\{1, 3, 4\}$ labels; the final label is business.

This email is written in Russian (Latin script).

**Subject:** *blank*

Privet!!!

Che takoe???

U menia vse normal'no, vchera nas evakuirovali srazu,

trseliy den' sideli doma smotreli telivizor... U tebia che?

This error analysis confirms that email classification into business and personal is a non-trivial task, as shown in section 4.4 on page 46. Also, it shows that the erroneous cases made by the classifier are not trivial, and many of these errors are caused by gold standard errors.

## 14.3 Performance on the Test Set

Finally, we select the models with the highest personal F-1 score for both settings (intra-corpus and cross-corpora), then, we test these models on Enron$_{T\ ts}$ and Avocado$_{T\ ts}$. The best models are LSTMs with majority vote from Table 12.2. Table 14.5 shows the performance of the best

| | | | Business | | | Personal | | |
|---|---|---|---|---|---|---|---|---|
| Model | Test Set | Accuracy | F-1 | Recall | Prec. | F-1 | Recall | Prec. |
| LSTMs + Maj. Vote | Enron$_{T\ ts}$ | 91.03 | 94.71 | 92.87 | 96.64 | 70.49 | 79.3 | 63.44 |
| LSTMs + Maj. Vote | Avocado$_{T\ ts}$ | 92.65 | 95.89 | 94.18 | 97.67 | 64.92 | 76.87 | 56.18 |

Table 14.5: Applying best models on test sets: Enron$_{T\ ts}$ and Avocado$_{T\ ts}$. Both models trained on Enron$_{T\ tr}$.

models on the test sets. We observe a drop in the performance for both test sets in comparison to the corresponding development set. For Enron, we expect a slight decrease in the results since that we optimize our models on the development set. However, for Avocado, we investigate the cause of the drop in the performance by using different models from Table 11.3 (with different feature sets). We find that these models always perform lower on Avocado$_{T\ ts}$ in comparison to Avocado$_{T\ dev}$. This suggests that Avocado$_{T\ ts}$ is just harder than Avocado$_{T\ dev}$. Note that the size of Avocado$_{T\ dev}$ and Avocado$_{T\ ts}$ and their ratio of personal emails are similar: $8.6\%$ and $9.1\%$, respectively.

## 14.4 Evaluation on Sheffield Data

In this subsection, we evaluate SVM classifiers on the Sheffield dataset (subsection 4.2.2). The information about the experiments described in Jabbari *et al.* (2006) is not detailed and does not mention the train and test ratios. We divide the Sheffield set into 75% and 25% for train and test, respectively. Table 14.6 shows the results of three SVM classifiers: with network features only,

|  |  | **Bus** |  |  | **Pers** |  |  |
|---|---|---|---|---|---|---|---|
| Model | Acc | F1 | Rec | Prec | F1 | Rec | Prec |
| shf | 93 | 95 | 99 | 92 | 80 | 69 | 95 |
| net | 86.2 | 90.2 | 87.4 | 93.1 | 77.2 | 83.2 | 72.0 |
| lex | 95.3 | 96.7 | 96.8 | 96.7 | 91.6 | 91.4 | 91.8 |
| all | 96.0 | 97.2 | 97.6 | 96.9 | 92.7 | 91.8 | 93.6 |

Table 14.6: Results of our models on the Sheffield dataset. We show numbers reported in (Jabbari *et al.*, 2006) as (shf); their results are not directly comparable and are only shown for rough benchmarking.

with lexical features only, and with combination of both features. Also, we report the results of the preliminary experiment reported in Jabbari *et al.* (2006) for convenience. However, the results are not directly comparable, as we do not know what their training data was. The results show that our models outperform the results presented in Jabbari *et al.* (2006). Moreover, similar to evaluation on our datasets, the results of evaluating on the Sheffield dataset confirm that incorporating social network features with the lexical features outperforms modeling emails with lexical features only. Note that we are not modeling threads here as that the Sheffield dataset does not maintain the thread structure.

## 14.5 Conclusions

In this part, we have presented various experiments for the task of email classification into "business" and "personal" by incorporating information from the social network as well as by modeling the thread structure. We focus on detecting the rare personal emails, and we evaluate our methods on two corpora, Enron and Avocado, only one of which we train on. The experimental results reveal that:

- Adding the social network information from different graph representations improves over the performance of an approach based on textual information only. The improvement by adding social network features is statistically significant.
- We have shown two techniques for modeling the thread structures: a simple majority vote, and sequential modeling of threads using LSTMs. The results also show that considering the thread structure of emails improves the performance further. Both techniques help in improving the classification performance, and combining them improves the performance further.
- We also compare our models with GraphSAGE, a state-of-the-art graph embedding model, and our models outperform it in this task.
- Furthermore, we extend the GraphSAGE model to a bipartite graph of users and emails. The results show that it performs better than the ordinarily model with a homogeneous graph of emails as nodes. However, our proposed social network features outperform both GraphSAGE models.

# Part III

# Other Applications

# Chapter 15

# Overt Display of Power

In this chapter, we show another application on emails, Overt Display of Power (ODP). We extend previous studies (Prabhakaran *et al.*, 2012b; Prabhakaran, 2015) that mainly focus on data annotation and detecting overt display power in emails at the utterance level using linguistic features. We study the underlying social networks of emails for the task of ODP detection. Our extension of the studies mentioned above is in:

- Exploiting the social network for the task of ODP.
- Studying ODP at an email-level rather than an utterance-level.

As in personal and business emails, the same pair of people might exchange both emails with and without ODP. However, the intuition is that unlike the case with personal and business email classification where a pair of users might exchange both types of emails; ODP classes of emails exchanged between a pair of users is asymmetric with a few exceptions. In other words, we expect that when the user $u$ sends an ODP email to the user $v$ there would not be an email from $u$ to $v$ with ODP. We are interested in incorporating social network features discussed in chapter 7 for the task of detecting ODP instances.

In section 15.1, We briefly review definitions introduced in (Prabhakaran *et al.*, 2012b) and state the motivation of modeling ODP using our proposed methods. Then, we discuss the dataset and our usage of it in section 15.2. We show some analysis of the dataset and the underlying social network of interaction in section 15.3. After that, we discuss methods and features used for ODP detection in emails in section 15.4. We show the experiments and results in section 15.5. We conclude the chapter in section 15.6.

## 15.1 Definitions and Motivation

Prabhakaran *et al.* (2012b) introduced the notion of "Overt Display of Power" (ODP) in written dialogues, particularly emails, to capture utterances in dialogs that display the exercise of power in an overt way. In other words, ODP is the situation when a person explicitly shows power signals over the other interlocutor.

**Utterance-level ODP** Prabhakaran *et al.* (2012b) study ODP detection at an utterance level within an email. The task they are interested in is to automatically tag utterances in a given email as ODP or not.

**Email level ODP** We introduce another task which is different from the one in Prabhakaran *et al.* (2012b); Prabhakaran (2015): rather than detecting ODP instances at an utterance level, we detect ODP instances at an email level.

As this task involves social interaction, which can induce a social network of communication, we are interested here in incorporating and studying the induced social networks for the ODP at the email level task. Note that, at an utterance level, all utterances within the same email will have the same social network information, while for emails, emails with different senders and recipients differ in their social network information. In this thesis, we are interested in incorporating the social network of interaction for text classification. We limit our study in this chapter to the email level ODP detection.

| | | |
|---|---|---|
| Threads | | 122 |
| Emails | Total | 357 (3 duplicates are discarded) |
| | Email with at least one ODP instance (POS) | 73 (20.4%) |
| | Email with no ODP instances (NEG) | 284 (79.6%) |
| Utterances | Total | 1734 |
| | Utterances with ODP (POS) | 86 (95.04%) |
| | Utterances without ODP (NEG) | 1648 (4.96%) |

Table 15.1: Summary of the ODP dataset annotations.

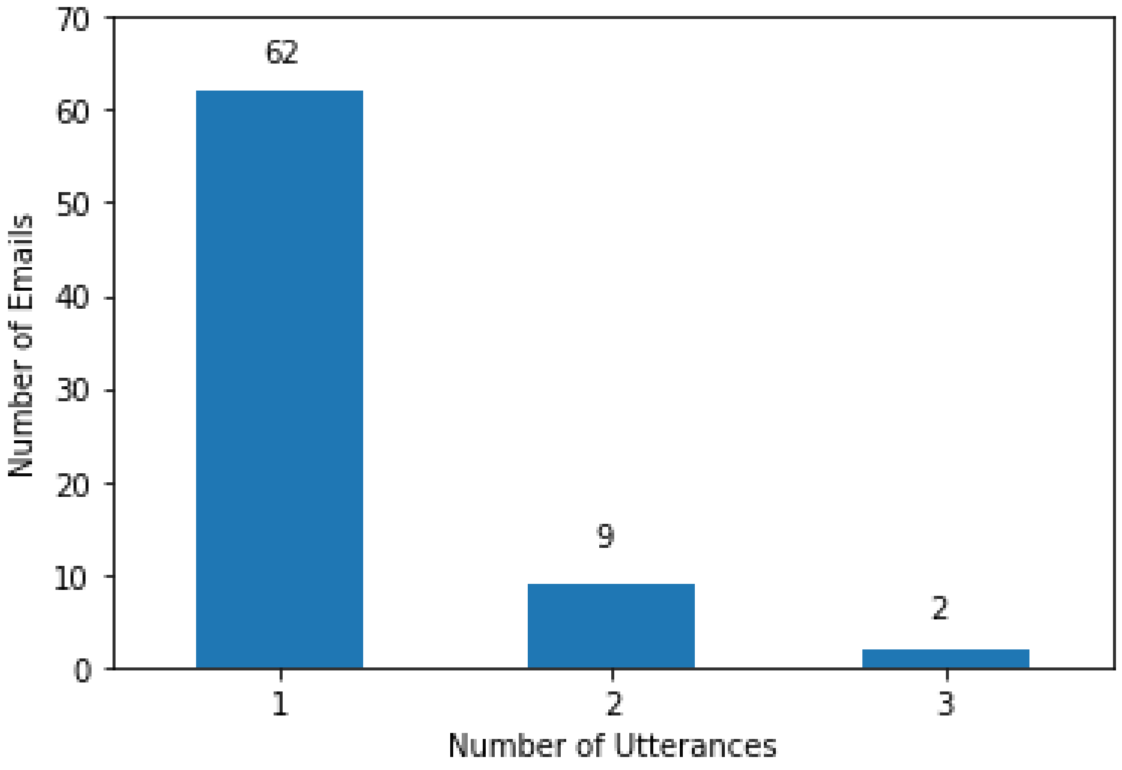


Figure 15.1: Distribution of the number of ODP utterances in emails. There are 284 emails without any ODP utterance.

## 15.2 Dataset

In this section, we discuss the dataset for ODP we use in our experiments. We use the ODP-UTTERANCE corpus introduced by Prabhakaran *et al.* (2012b) (discussed in subsection 3.1.3). It consists of 122 email threads manually annotated with instances of overt display of power at the utterance level such that each message is segmented into dialog functional units, and each dialog functional unit is further split into utterances. As we are interested in detecting ODP occurrences at the email level, we create a new dataset of emails labeled with ODP induced from ODP-UTTERANCE. Given an email message, we label it as a positive instance of ODP if there is at least one occurrence of ODP utterances in this email. Otherwise, we label it as a negative instance. Our annotation here is automatic, and we depend on the annotations in ODP-UTTERANCE.

We extend and clean up the data set for overt display of power ODP-UTTERANCE introduced in subsection 3.1.3. We manually reviewed some emails and found that there are 3 duplicates that we discard. Then, we label the whole email as a positive ODP instance if it has at least a single ODP utterance; a negative instance otherwise. We call this dataset ODP-EMAIL. Table 15.1 summarizes the datasets.

Figure 15.2 below shows two example emails in the dataset. The first email does not contain ODP utterance. In the second email, the sender is overtly displaying power by saying, *"can you do so **immediately**"*.

## 15.3 ODP and Social Network: Statistical Analysis

In this section, we conduct an analysis on the ODP-EMAIL dataset. We analyze email labels for each pair of sender and recipient and the dataset. We use emails for all pairs of sender and recipient in the dataset. We are interested in analyzing how ODP emails are exchanged among pairs. Here,

**Example 1** An email without any ODP instance

Sara,

I need to report time. Were you out in the last two weeks?

Becky

**Example 2** An email with an ODP instance

Did you get this reserve cleared through Lavorato?

If you have not, can you do so immediately.

Figure 15.2: Examples of emails with and without ODP.

the pair of participants ($p_1$, $p_2$) is formed such that the sender is the first user in the pair ($p_1$) and the recipient is the second ($p_2$). Note that the pair with the opposite order ($p_2$, $p_1$) is different and might appear in other emails. We call the pair ($p_2$, $p_1$) the "opposite" pair for ($p_1$, $p_2$). Table 15.2 shows the statistics for pairs in the dataset. We observe that around 27% of pairs in the dataset having at least one ODP email sent from the sender to the recipient. From the dataset, only 13% of pairs of sender and recipient appear in other emails in the opposite direction (the recipient sends an email to the sender). There are only 10 pairs such that both users exchange ODP emails. This is expected as the intuition is that people who overtly display power usually are in a higher rank.

| | |
|---|---|
| Total number of unique pairs | 1,130 |
| - pairs with at least 1 ODP email | 306 |
| Total number of opposite pairs | 150 |
| - both pair and opposite having 1 ODP email | 10 |

Table 15.2: Statistical analysis on the participant pairs in ODP-EMAIL dataset. Opposite pair means that for an email in the dataset, it has a pair of participant ($p_1$,$p_2$) such that $p_1$ is the sender and $p_2$ is a recipient, there exists another email with the pair ($p_2$,$p_1$) such that $p_2$ is the sender and $p_1$ is the recipient.

## 15.4 Methods

In this section, we show methods we are using for ODP detection at the email level. We first discuss lexical features for modeling the email content. Then, we discuss methods for incorporating social network information for the ODP detection task.

**Sentence:** *"Please resend to me at the university."*

**Dialog Act:** *"Request-Action"*

**Tokens:** 'Please', 'resend', 'to', 'me', 'at', 'the', 'university', '.'

**POS tokens:** 'UH', 'VB', 'TO', u'PRP', 'IN', 'DT', 'NN', '.'

**Lemma Tokens:** 'please', 'resend', 'to', 'I', 'at', 'the', 'university', '.'

| Feature Set | Example |
|---|---|
| LemmaNGram | {please, resend, please resend, to I ...} |
| POSNgram | {UH, VB, UH VB, ...} |
| MixedNgram | {please VB to I, ...} |

Table 15.3: Features adopted from Prabhakaran *et al.* (2012b) for ODP detection.

### 15.4.1 Lexical Modeling

We adopt features from (Prabhakaran *et al.*, 2012b) as well as averaged FASTTEXT word embeddings for the email content (section 8.4). Prabhakaran *et al.* (2012b) propose different types of features to capture linguistic and syntactic patterns. These features include n-grams of part-of-speech (POS), Lemma n-grams, mixed n-grams, and Dialog Act Tags. Below, we briefly describe each feature. For POS tagging and lemmatization, we use the Stanford CoreNLP toolkit (Manning *et al.*, 2014).

Table 15.3 shows an example of a sentence: *"Please resend to me at the university."* with its extracted features.

**POSNGRAM (PN)** N-grams of part-of-speech. We first tag texts with their POS tags. Then, we extract n-grams for POS tags.

**LEMMANGRAM (LM)** Sequences of word lemmas of length n or smaller.

**MIXEDNGRAMS (MN)** A mixed n-gram is a special case of lemma n-grams where words belonging to open classes are replaced with their POS tags.

**DIALOGACT Tags (DA)** Another set of features we use in this task is the dialog act tags. The corpus is already tagged with dialog act by Hu *et al.* (2009) in which each message is segmented into dialog functional units (DFUs), and each dialog functional unit is further split into utterances. Below, we give definitions for each DA tag in the corpus.

**CONVENTIONAL** These are greeting, introductions, expression of thanks, etc.

**INFORM** This DFU conveys information. This covers many different types of information that can be conveyed, including answers to questions, elaborations, reporting completion of a requested action, and so on.

**REQUEST-ACTION** This DFU obliges the hearer/reader, or opens an option to the hearer/reader, to perform some non-communicative action, i.e., an action that cannot be part of the dialog.

**REQUEST-INFORMATION** This DFU obliges the hearer/reader, or opens an option to the hearer/reader, to provide information (either facts or opinion), either in the dialog or through another form of communication.

Prabhakaran *et al.* (2012b) report that the best performance of ODP detection at an utterance level is the combination of POSNGRAM, MIXEDNGRAMS, and DIALOGACT Tags (PN, MN, DA).

We chose this subset of features with the best performance reported in Prabhakaran *et al.* (2012b). We combine these features with our social network features as well as the average word embeddings for emails.

### 15.4.2 Social Network Modeling

For social network modeling, we use both the social network features (**net**) discussed in chapter 7 and GraphSAGE Hamilton *et al.* (2017a) discussed in section 8.5. For **net** features, we extract them from the graph constructed using the whole Enron corpus, not just the labeled examples. This overcomes the issue of having a small dataset with only 357 emails. For GraphSAGE, we construct a document graph (i.e., email graph) (section 5.4; page 58) such that nodes represent emails, and we link emails if they share common participants. We use the whole Enron corpus when constructing this graph. Note that GraphSAGE aggregates information from the neighboring nodes which do not have to be labeled. For the lexical features for the email nodes, we use the average FASTTEXT word embeddings for the email content (section 8.4). We use the default hyper-parameter values for the PyTorch implementation for GraphSAGE .[1]

## 15.5 Experiments

In this section, we present experiments for ODP detection at the email level. We evaluate the performance using k-fold cross-validation. Following Prabhakaran *et al.* (2012b), we choose $k = 5$. The folds do not cross thread boundaries, such that no two folds contain emails from the same thread, and all emails from the same thread are contained in and only in a single fold. We report performance using the average scores of the cross-validation. We optimize the F-1 score for the positive class. Note that the dataset is tiny and imbalanced as there are 357 emails and only 20%

[1] `https://github.com/williamleif/graphsage-simple/`

| | | | Positive | | | Negative | | |
|---|---|---|---|---|---|---|---|---|
| | | Accuracy | F1 | Recall | Prec | F1 | Recall | Prec |
| Baseline | All-True | 20.45 | 33.95 | **100.0** | 20.45 | 0.0 | 0.0 | 0.0 |
| | All-False | 79.55 | 0.0 | 0.0 | 0.0 | **88.61** | **100.0** | 79.55 |
| | Random | 65.7 | 21.45 | 18.94 | 24.73 | 78.06 | 81.06 | 75.27 |
| | PN, MN, DA † | 79.15 | 52.2 | 55.13 | 50.61 | 86.15 | 84.97 | 87.49 |
| | GraphSAGE | 76.34 | 52.17 | 52.17 | 52.17 | 84.29 | 84.29 | 84.29 |
| | net only | 71.33 | 32.05 | 35.99 | 31.5 | 80.75 | 79.2 | 83.05 |
| | FastText | 74.79 | 54.55 | 73.97 | 43.2 | 82.56 | 75.0 | 91.81 |
| | FastText + net | 75.63 | 56.28 | **76.71** | 44.44 | 83.11 | 75.35 | **92.64** |
| | PN,MN,DA + FastText | 81.69 | 58.65 | 64.89 | 53.64 | 87.72 | 85.34 | 90.29 |
| | PN,MN,DA + FastText+net | **82.81** | **60.72** | 65.8 | **56.79** | **88.43** | **86.34** | 90.75 |

Table 15.4: Results of ODP tagging at an email level. †: best features reported in Prabhakaran *et al.* (2012b) . PN: POS N-grams; MN: Mixed N-grams; DA: Dialog Acts. net: using social network features (chapter 7). FastText: using average FastText embedding for the email (section 8.4).

labeled ODP positive. Table 15.4 summarizes the results.

In Table 15.4, we first show the performance of three weak baselines: All-True, All-False, and Random. The first baseline predicts all emails as positive instances, the second predicts all emails as negative instances, and the third predicts email labels randomly by respecting the class distribution. Then, we show the performance of a linear SVM classifier trained with the best features from Prabhakaran *et al.* (2012b) (i.e., **PN, MN, DA**). Note that the results here are not directly comparable with the results reported in Prabhakaran *et al.* (2012b) as they model the task at an utterance level while we model it at an email level. In the third box, we show the performance of GraphSAGE, which is our strong baseline. We feed the GraphSAGE model with an undirected email network such that nodes represent emails, and we link emails if they share participants (i.e., sender or recipients); we use the email average FastText word embedding for the node features. In the last box in Table 15.4, we present the results of the social network features (**net**) and lexical modeling of emails using FastText word embeddings as features instead of hand-crafted linguistic features. We also show combinations of different features. We use an SVM classifier with a linear kernel in all experiments except for the GraphSAGE.

We observe that the model with network features only (**net**) retrieves more than 35% of the ODP positive emails even without exploiting the email content. This indicates that the email communication network is an indicator of power relations. However, modeling the lexical content of emails gives much higher performance than modeling the social network. Also, using word embeddings as lexical features instead of the hand-crafted linguistic features (i.e., **PN, MN, DA**) improves the F-1 score for the positive class but hurting the F-1 score for the negative class. In particular, the classifier trained using the FastText embeddings retrieves much more ODP positive emails than the model trained with the hand-crafted linguistic features. For GraphSAGE, we get a slightly lower performance than the lexical models. This is expected given the size of the dataset as GraphSAGE is a neural network model that requires a relatively large dataset. Additionally, adding the social network features to the FastText embeddings improves the classification performance for both the positive and negative scores over using the FastText embeddings alone. However, the hand-crafted linguistic features (i.e., **PN, MN, DA**) perform better on the negative score. Combining the hand-crafted linguistic features with the FastText embeddings improves the performance on both the positive and negative scores over other models. Our best model uses the combination of the linguistic features from Prabhakaran *et al.* (2012b) (i.e., **PN, MN, DA**), average FastText embeddings for the email content, and social network features (net). Given that the size of the dataset is tiny (less than 360 emails), we do not perform statistical significance analysis for this task.

## 15.6 Conclusion

In this chapter, we have shown another task on email, Overt Display of Power (ODP). We extended a previous study introducing features to capture linguistic and syntactic patterns (Prabhakaran *et al.*, 2012b). Unlike the previous work, we model the task at an email level (instead of an utterance level), and we present social network modeling methods for this task. We found that incorporating social network information extracted from the underlying communication network with lexical content of emails achieves the best performance among other models that use lexical only. Our best model uses the best hand-crafted linguistic features from previous research (Prabhakaran *et al.*, 2012b) and FastText word embeddings as lexical features in addition to our proposed social network features. Additionally, our proposed social network features outperform the state-of-the-art model for graph modeling, GraphSAGE.

# Chapter 16

# Hierarchical Power Prediction

In this chapter, we show another application of incorporating the social network for text classification on the email genre: Hierarchical Power Prediction. The task here is to predict whether there is a hierarchical power relation between a pair of people or not given a set of emails exchanged between this pair. In other words, For a given pair of people $a$ and $b$ who exchange emails, we want to predict if there is a hierarchical power relation between $a$ and $b$ or not, regardless of the direction. We define hierarchical power as follows: a pair of people are in a hierarchical power relation if they are in a chain of command in the company's hierarchy. We use the Enron corpus for this task. Although the final task is not mainly a text classification one, we apply different methods and techniques for text classification in the context of social networks discussed so far in this thesis. We formulate the problem as a text classification problem such that we use emails exchanged between a given pair of people for modeling the hierarchical power relation.

We first discuss related work in section 16.1. Then, we present the dataset used for this task in section 16.2. We show methods for modeling the task in section 16.3. Then, we present experiments and results in section 16.4. Finally, we conclude the chapter in section 16.6

## 16.1 Related Work

Since the introduction of the Enron email corpus, there has been a large body of research for studying hierarchical power relations in the Enron corpus for a wide variety of applications: the effect of relevant power on the sender's choice of formality Peterson *et al.* (2011), and predicting hierarchical power (Prabhakaran and Rambow, 2014; Agarwal *et al.*, 2012). Early studies on predicting the Enron hierarchy have been hampered by a lack of data about the organizational hierarchy. As one of the earliest studies on extracting the Enron hierarchy, Shetty and Adibi (2004) assembled a list of job titles for the core 158 Enron employees whose complete mailboxes were released. Other researchers have attempted to predict the relative ranking of two people's job titles using this list (Rowe *et al.*, 2007; Palus *et al.*, 2011).

Agarwal *et al.* (2012) released a gold Enron, which they extracted manually by studying the original Enron organizational charts in Enron email attachments. We use this gold-standard in this chapter. In addition, they predict organizational hierarchy using SNA. Particularly, they use the degree centrality of every node in the email exchange network, and then, rank the nodes by their degree centrality. In our work, we use emails exchanged between a pair of people to predict whether they are in the same managerial lineage or not, regardless of the direction of power. We use a wide variety of social network features in addition to email content.

## 16.2 Dataset

For the task of predicting hierarchical power relations, we use the Enron power annotation dataset ENRON-POWER from Agarwal *et al.* (2012) (subsection 3.1.4) to obtain the hierarchical labels. As mentioned in section 3.1, the Enron email corpus is a small subset of all Enron emails. The corpus has all of the mailboxes for the **core** Enron employees; while for the **non-core** employees, we only have emails in which the **core** employees appear as senders or recipients. The corpus does not have emails exchanged between the **non-core** employees unless there are email participants in the **core** group.

ENRON-POWER is a MongoDB database that contains hierarchical relation information of Enron's employees as well as departments. The entries are stored as a MongoDB collection named ”Entities”. Of which, there are 3,187 entries with the hierarchy information. These entries represent both employees and departments. For hierarchical power modeling, we are interested in extracting the power relations between employees. Some of these entries representing people have multiple nodes distinguishing between various positions for people at different points in time. Among these entries, there are 1,518 entries that have user ids such that we can map it to the emails (senders or recipients).

The ENRON-POWER dataset does not explicitly distinguish between employee nodes and department nodes. However, we can infer most of the node types by using some information. Particularly, for some nodes, we can directly infer their type. Those nodes have values in the “uid” field, and some of them have multiple position nodes distinguishing between various positions for employees at different points in time. Only employee nodes have this property. In addition, there are three types of edges in the annotated dataset:

- “contains” edge: from a department node to any other node.
- “manages” edge: from an employee node to a department node.
- “supervises” edge: from an employee node to another employee node.

From these edge types, we can infer most of the node types except for nodes that have only incoming edges with type “contains”; as this type of edges cannot be used to distinguish the node type for the target nodes (because “contains” can go to either an employee or department).

To obtain the hierarchical information for Enron, we construct a graph of people with their hierarchical information in a three-phase graph construction: Big Graph $G_B$, Combined Graph $G_C$, and Final Graph $G_F$. Table 16.1 shows statistics for these graphs. For node types, we use “emp” for employee nodes, “dept” for department nodes, and “else” for nodes with an unknown type.

| | $G_B$ | $G_C$ | $G_F$ |
|---|---|---|---|
| Number of nodes | 3,479 | 3,187 | 1,518 |
| “emp” | 2,236 | 1,944 | 1,518 |
| “dep” | 948 | 948 | 0 |
| “else” | 295 | 295 | 0 |
| | | | |
| Number of edges | 3,458 | 3,446 | 1,717 |
| “contains” | 1,860 | 1,857 | 0 |
| “manages” | 549 | 549 | 0 |
| “supervises” | 1,049 | 1,040 | 1717 |
| | | | |
| Weekly connected components | 21 | 10 | 10 |

Table 16.1: Statistics for graphs used to induce hierarchical power relations in Enron.

We start by constructing a graph named $G_B$ using all nodes and edges in the MongoDB entries with hierarchical information. Each node in $G_B$ represents a unique position (as some employees have different positions at different times). We assign the “emp” type to each node having a user id, as these nodes represent actual individuals. For each node with outgoing or incoming edges with type “supervises”, we assign the ‘emp” type. For edges with the “manages” type, we assign the “emp” type to the source node and “dept” to the target node.

Then, we construct another graph from $G_B$ in which we combine nodes representing different positions for the same person (i.e., having the same uid) into one node. We call this graph the combined graph $G_C$. Finally, we construct the final graph $G_F$, which has only employee nodes by removing all non-employee nodes.

From $G_C$, we induce the final graph $G_F$ that has only nodes with uids (employee nodes) by deleting every other node. For each node to be deleted, we replace all of its outgoing edges with all incoming edges. We recursively delete nodes that do not have a uid until no more nodes are left. We delete self-loops from the final graph $G_F$. It has one cycle after deleting self-loops. This is due to the fact that some employees have different position nodes, each for a different time point. Each position node can supervise and be supervised by different people, which results in a cycle in the hierarchical power relation graph. For instance, an employee $a$ has two position nodes: $a_1$ and $a_2$ can be both supervising and being supervised by $b$ such that the position node $a_1$ is supervised by $b$ and $a_2$ supervises $b$. We randomly deleted an edge from the cycle as we do not have information about which edge is the most recent one, i.e., which position is the most recent.

| | | |
|---|---|---|
| Number of pairs | 4,459 | |
| Positive | 706 | (15.83%) |
| Negative | 3,753 | (84.17%) |

Table 16.2: Summary of the Enron hierarchical power relation dataset (ENRONPOWER).

Table 16.2 shows the summary of the Enron hierarchical power pairs. Note that pairs who have never exchanged any emails are excluded.

## 16.3 Methods

In this section, we present methods for modeling the task of hierarchical power prediction. Recall that the task here is to predict whether or not a pair of people are in a hierarchical relation, regardless of the direction. Emails exchanged between a pair of people can be good indicators for power

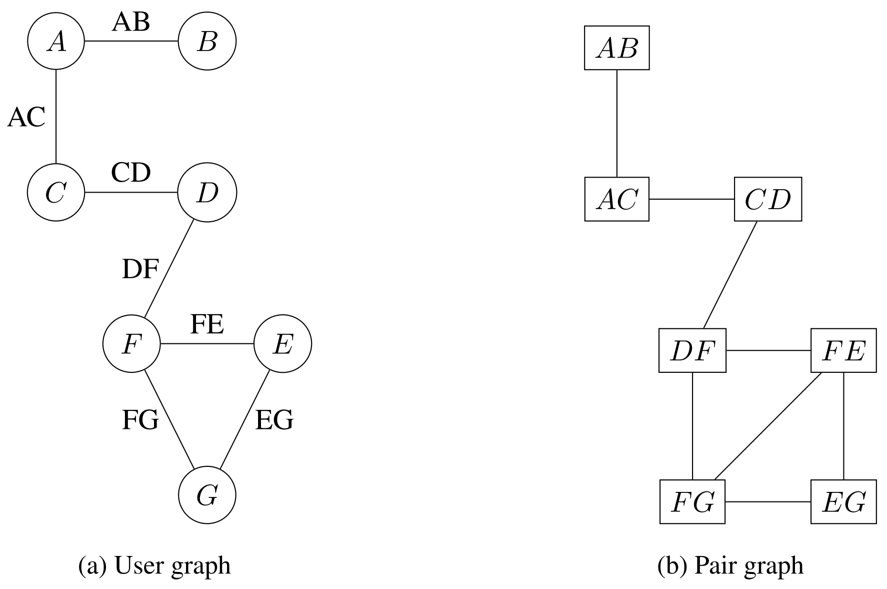


Figure 16.1: Construction of the Pair Graph. Figure 16.1a (left) shows the user graph induced from the email exchange network: nodes are linked if the two ends (users) exchanged at least one email. Figure 16.1b (right) shows the pair graph, the line graph for the user graph. Nodes represent a pair of users (edges in the user graph) and linked if their corresponding edges in the user graph share a common node (user).

information; assuming that organizational hierarchy is unknown for the predictor, we are interested in exploiting information from emails exchanged between people in order to predict if there is hierarchical power relation or not. We formulate the task of hierarchical power prediction as a text classification task such that given a set of emails exchanged between a pair of people, we are interested in classifying whether all of these emails belong to people in a hierarchical power or not. This makes the task not on a single email level but on the level of all emails exchanged between the pair. We exploit both the lexical content of the emails and the social network that can be induced from the email exchange.

Given a pair of people, we first extract emails exchanged between them – we do not distinguish between the sender and recipient in this case. Then, we use these emails to predict whether there exists a hierarchical relation between this pair or not.

We incorporate both the lexical content of the emails and the social networks induced by these emails. We apply different techniques discussed in this thesis so far. For lexical features, we use the average word embeddings for the email content. We include both the subject and the body of the email. We use FastText to obtain the word embeddings pre-trained on the whole Enron corpus.

For social network modeling, we use the social network features for email, as discussed earlier in chapter 7. Then, we average the features for all emails exchanged between a pair of people regardless of the communication direction. Also, we experiment with GraphSAGE for predicting whether a hierarchical relation exists or not. For GraphSAGE, it is not obvious how to represent pairs for the GraphSAGE model, given that it is designed for homogeneous graphs. Accordingly, we investigate using a graph representation for a pair of users in which nodes represent pairs (instead of single users). The line graph $L(G)$ of an undirected graph $G$ is another graph $L$ that represents the adjacencies between edges of $G$. It is defined to have as its nodes, the edges of $G$, with two being adjacent if the corresponding edges share a node in $G$. In other words, edges in $G$ become nodes in $L$ and the nodes in $L$ are linked if the corresponding edges in $G$ share a node. Accordingly, to construct a pair graph, we can simply use the line graph of the user graph induced from the email exchange network. The resulting graph $G_p$ has pairs as nodes, and they are linked if the pair (represented as a node) share a common user.

To construct the pair graph, we first construct the user graph $G_u$ section 5.3 such that users (nodes) are linked if they have exchanged at least one email. Then, for each edge in $G_u$, we form a pair of the two ends. Finally, we construct the pair graph $G_p$ by construing the line graph of $G_u$. Figure 16.1 shows an example of a user graph and its corresponding pair graph. We use this graph as the input for the GraphSAGE model. The nodes have the average email embeddings as features.

| | | Positive | | | Negative | | |
|---|---|---|---|---|---|---|---|
| | Accuracy | F-1 | Recall | Precision | F-1 | Recall | Precision |
| All-NEG | 82.33 | 0 | 0 | 0 | 90.31 | 100 | 82.33 |
| Random | 72.49 | 16.35 | 15.22 | 17.67 | 83.54 | 84.78 | 82.33 |
| lex | 83.23 | 35.29 | 25.89 | 55.43 | 90.37 | **95.53** | 85.73 |
| net | 82.51 | 39.63 | 32.49 | 50.79 | 89.77 | 93.25 | 86.55 |
| lex+net | **86.28** | **54.60** | 46.70 | **65.71** | **91.92** | 94.77 | 89.23 |
| GS-pair | 85.65 | 47.02 | **67.62** | 36.04 | 91.70 | 87.52 | **96.30** |

Table 16.3: Enron hierarchical power prediction results. Positive means a pair of users is in a hierarchical power relation (in a chain of command); negative means they are not.

## 16.4 Experiments and Results

In this section, we present experiments and results for the task of predicting hierarchical power relations in the Enron corpus. We divide ENRON-POWER into train and test sets with 75% and 25% of the pairs, respectively. We use SVM classifiers with linear kernels with different feature sets:

- **lex**: average of word embeddings of all emails exchanged between the pairs.
- **net**: social network features for pairs as the average social network features of emails exchanged between the pairs' users.
- **lex+net**: concatenation of **lex** and **lex+net**.

For lexical features **lex**, we use the average FastText embeddings (section 8.4) for the email exchanged between all pairs. Each email is represented using a 100-dimensional vector. Then, we compute the average of all email vectors exchanged between the users in a pair. For the social network features **net**, we use features discussed in chapter 7. We also experiment with GraphSAGE fed with the pair graph $G_p$ (**GS-pair**). We use the default hyper-parameter values as in the PyTorch implementation for GraphSAGE .[1] We report the performance for different feature sets, and we optimize the F-1 score for the minority class (i.e., positive instances). We show the performance of two baselines: All-NEG and Random classier. The All-NEG classifier always predicts the majority class while the Random classifier randomly predicts labels respecting the training set's class distribution.

Table 16.3 summarizes the results. The first line presents a baseline classifier that always predicts the majority class (negative). The second line shows the expected scores for a random classifier that makes predictions by respecting the class distribution.

We observe that a lexical classifier retrieves 25% of the positive instances, while a classifier that is trained on network features retrieves about a third of the positive instances. Also, unlike the task of business and personal email classification, we observe that **net** performs better than **lex** on the positive F-1 measure. This can be explained by the fact that while the same pair of people can exchange both personal and business emails. This is not the case with the hierarchical power relations, as different emails involving the same participant have different lexical content, but the same social network information.

In addition, combining the two feature sets boosts the performance much further. Interestingly, given the recall scores for each feature set and the combination of them, we observe that the two feature sets **lex** and **net** retrieve almost two different sets of positive instances. This suggests that these two sets of features capture different things. Similarly, we observe that **GS-pair** performs much better than the classifier trained only on the lexical content of emails. However, combining our proposed social features with the lexical content **lex+net** gives a better performance than **GS-pair** on the F-1 score. Note that **GS-pair** has the highest recall score for the positive class among other classifiers but with a low precision score.

To determine whether the performance improvement of different classifiers over others is statis-

[1] `https://github.com/williamleif/graphsage-simple/`

tically significant, we use the non-parametric Wilcoxon Signed-Rank Test (Sidney, 1957) on pairs of the positive F-1 scores of different classifiers using 10 fold-cross validation runs. The analysis shows that the improvement in the performance of all other classifiers over both the ALL-NEG and the random classifiers are statistically significant ($p < 0.01$). Additionally, the improvement in the performance by combining the lexical features and the social network features (**lex+net**) over using either **lex** or **net** is statistically significant ($p < 0.01$). More importantly, the improvement in the positive F-1 score of our best model (i.e., **lex+net**) over the state-of-the-art graph embedding model (GS-pair) is statistically significant ($p < 0.01$).

## 16.5 Post-hoc Analysis

In this section, similar to the analysis we have shown in section 11.4, we perform a post-hoc analysis by inspecting the feature weights of the classifier trained using the **net** features only.

In a linear kernel SVM such as the one we used in this task, the feature weight assigned in the model for each feature is an indicator of how that feature correlates with the class being predicted. Table 16.4 shows the top weights for an SVM classifier with a linear kernel trained only on the **net** features.

Similar to business and personal email classification, we observe that most of the top features are extracted from the user graphs (both directed and undirected), while only a few features are extracted from the bipartite user-email graph. Additionally, similar to business and personal email classification, business and personal email classification are recipient-related features. However, the proportions are different as there are more sender-related features in the top social network features for the hierarchical power detection task. Note that for features involving the sender and recipient pair, we compute these numbers for every pair of sender and recipient; then, we average the numbers.

| Top Positive Social Network Features | | Top Negative Social Network Features | |
|---|---|---|---|
| D_recipient_eigenvector_centrality_max | 11.56 | D_sender_degree_centrality | -18.87 |
| D_recipient_degree_centrality_min | 11.54 | D_recipient_out_degree_min | -14.90 |
| U_#common_neighbors | 10.66 | D_recipient_degree_centrality_max | -14.62 |
| D_recipient_w_out_degree_min | 9.82 | U_sender_w_degree | -13.88 |
| D_recipient_hub_score_max | 8.23 | U_recipient_w_between_centrality_avg | -11.31 |
| U_sender_#triangles | 8.19 | D_recipient_eigenvector_centrality_avg | -11.14 |
| D_recipient_w_in_degree_min | 8.00 | D_recipient_degree_centrality_avg | -9.82 |
| D_sender_w_in_degree | 7.88 | #recipients | -9.59 |
| U_recipient_w_between_centrality_max | 7.55 | D_recipient_hub_score_avg | -7.52 |
| U_sender_betweeness_centrality | 7.23 | U_recipient_w_degree_min | -7.28 |
| U_sender_degree_centrality | 7.19 | D_recipient_eigenvector_centrality_min | -5.93 |
| U_sender_degree | 7.19 | D_recipient_between_centrality_max | -4.77 |
| D_sender_w_out_degree | 7.12 | U_sender_w_betweeness_centrality | -4.48 |
| U_recipient_degree_centrality_max | 6.60 | U_#common_neighbors_norm_triangle | -4.19 |
| U_recipient_degree_max | 6.60 | U_recipient_degree_centrality_min | -4.18 |
| D_recipient_out_degree_centrality_min | 4.56 | U_recipient_degree_min | -4.18 |
| U_recipient_between_centrality_max | 4.04 | U_recipient_eigenvector_centrality_max | -3.23 |
| D_recipient_w_in_degree_max | 4.03 | D_recipient_between_centrality_avg | -3.18 |
| U_recipient_degree_avg | 3.97 | U_Jaccard | -2.61 |
| U_recipient_degree_centrality_avg | 3.97 | D_B_email_hub | -2.50 |

Table 16.4: Post-hoc analysis of social network features for hierarchical power prediction in email. We show the Top 20 **net** feature weights for the positive and negative classes. "U/D" denotes that the feature is extracted from the undirected/directed user graphs, respectively. "W" denotes that the graph is weighted. "B" denotes that the feature is extracted from the bipartite user-email graph. Absence of "B" denotes that the feature is extracted from the user graph.

The top positive **net** features include *U_#common_neighbors*, which indicates that when the pairs of sender and recipient in an email share many neighbors with whom they both exchange emails, it is likely that they are in a hierarchical power relation. The intuition here is that people who are in the same managerial lineage share neighbors with whom they exchange emails. In addition, the top positive **net** features include *U_sender_#triangles*, which indicates that senders who tend to cluster more often than others are more likely to be in a hierarchical relation. However, the top negative features include *U_#common_neighbors_norm_triangle* and *U_Jaccard*. This indicates that when normalizing the number of common neighbors either by the number of triangles for the sender or the total number of neighbors for both the sender and recipients, we capture different information. For instance, if "A" and "B" work in the same department and they are both supervised by C, they are not in a hierarchical relation, and they are likely to have very similar neighbors (other people in the same department) with whom they both exchange emails. On the other hand, their manager "C" shares many neighbors with "A" and "B", but "C" also has other neighbors from other departments (e.g., other managers).

Interestingly, the features *#recipients* and *D_B_email_hub* being in the top negative features indicates that if an email is sent to many and more influential recipients, it is highly likely that the pairs of sender and recipient in that email are not in a hierarchical power relation.

We observe that different centrality scores capture different things. Sometimes they are indicators for opposite classes. For instance, the top positive features include the max recipient eigenvector centrality score, and the top negative features include the max recipient degree centrality score – both from the directed user graph. Also, the top negative features include features indicating that the recipient is influential in the network, such as *D_recipient_eigenvector_centrality_avg* and *D_recipient_eigenvector_centrality_min*. However, the top positive features include *D_recipient_eigenvector_centrality*. We believe this is because when many recipients are influential, it is likely that the email is sent to many influential people in a different department (email between mangers), but when only one recipient has a high eigenvector centrality, the email is sent from someone who works under the supervision of that person.

## 16.6 Conclusion

We present another application in which we incorporate both lexical content and the social network information induced from the underlying social network of interaction. In addition, we make use of GraphSAGE for modeling this task. As this model is designed for homogeneous graphs, we introduce a homogeneous graph representation containing pair communication information. The results show that combining social network information and lexical features for emails improve the classification performance. Also, we report that our proposed features outperform the state-of-the-art model GraphSAGE in this task. For future work, we are interested in exploring other graph representations for pairs of users to be used with GraphSAGE or another generic graph embeddings model. We are also interested in predicting hierarchical power for pairs who never exchanged emails; we can use features from the corresponding user graph induced from the email exchange (with other users).

# Chapter 17

# Reddit Posts Classification

We have shown so far in this thesis work on incorporating social network information for various email classification tasks. In this chapter, we investigate another genre and application, Reddit posts classification. The task is to predict to which community different Reddit posts belong. Reddit [1] is a large online discussion platform where users post and comment in different communities called "subreddits". In this task, we have a set of posts from different subreddits which we are interested in classifying them into their corresponding community. The task here is different than the other tasks that have been presented so far in this thesis on the following:

- The genre is different; we have shown so far in this thesis tasks on email. Here we show work on Reddit posts.
- In email, documents are sent to a designated set of recipients defined by the sender. In Reddit, the author of a post does not specify a set of recipients, but rather a community (subreddit).
- In the previous tasks: "business" and "personal" email classification (Part II), overt display of power (ODP) detection (chapter 15), and hierarchical power classification (chapter 16), the

[1] `http://www.reddit.com`

problems were binary classification. Here, it is a multi-class classification task as we have more than two communities for posts.

In this chapter, we present different techniques for modeling Reddit posts. We apply the social network modeling methods proposed earlier in this thesis. In addition, we present other methods for modeling the communication network for the Reddit post classification task. We first discuss the dataset we use for this task in section 17.1. Then, we show different methods for modeling the task in section 17.2. After that, we show the experiments and results in section 17.3. Finally, we conclude the chapter.

| | |
|---|---|
| Number of posts | 232,965 |
| Number of comments | 7,317,217 |
| Number of users | 716,195 |
| Number of communities (classes) | 41 |

Table 17.1: the REDDIT-NEW post classification dataset statistics.

## 17.1 Data

Reddit is an online discussion platform where people can submit contents as posts or comment on other users' posts such that posts are represented as a threaded discussion. Reddit is split out into sub-communities, or "subreddits" which can be created by any user. Subreddits cover a wide range of topics and interests; they can be about a broad subject such as "science" or "news" or a specific one such as the video game "DotA".

In this section, we show the dataset we use for the task of Reddit post classification. We create a new dataset induced from REDDIT-FULL and GRAPHSAGE-REDDIT (section 3.3). We refer to this new dataset as REDDIT-NEW. GRAPHSAGE-REDDIT is a sample of $232,965$ posts in the month of September 2014. This sample was extracted from the REDDIT-FULL dataset. Hamilton *et al.* (2017a) use the GRAPHSAGE-REDDIT dataset to evaluate the GraphSAGE model for the Reddit post classification task. In particular, they use it for constructing a post-to-post graph, connecting posts if the same user comments on both, with word embeddings of post titles and comments as lexical features for the nodes (i.e., posts). In this chapter, we are interested in modeling the communication differently such that we incorporate information not only from posts but also from users. Note that GRAPHSAGE-REDDIT is missing some information about the social network. For instance, it does not specify who made a post or a comment. However, we can recover information about users from the original collection (REDDIT-FULL) as follows. For each post in GRAPHSAGE-REDDIT, we match the corresponding post as well as all comments made to that post in REDDIT-FULL. We use the post id, which is unique for each post. For comments, we match the post id with the *link_id* field, which stores the post id to which a comment was made. This allows us to retrieve data of users who made or commented on the posts.

Finally, for each comment and post, we extract the user id and the community id on which the post was posted. Table 17.1 shows the statistics for the REDDIT-NEW dataset. Figure 17.1 shows the distribution of posts in communities. Below, we show the description of the top 10 communities, extracted from each community about page.

**DestinyTheGame** This subreddit is for discussing Destiny 2 and its predecessor, Destiny, an online-only multiplayer first-person shooter video game.

**friendsafari** A place to exchange 3DS Friend Codes for the Pokémon X/Y Friend Safari!

**pcmasterrace** A subreddit of the PC Master Race. the PC Master Race is an internet subculture, internet community, and a term of superiority for PC gaming used among gamers to compare PC gaming to other gaming platforms (i.e., console gaming).

**buildapc** This is a community-driven subreddit dedicated to custom PC assembly. Users in this community share their experiences and seek help in PC assembly.

**DotA2** It is a competitive multiplayer video game. This subreddit is for any topic related to the game as well as the competitive scene surrounding it.

**trees** A subreddit for anything and everything cannabis. It is described in the about community section as "the casual cannabis community".

**fantasyfootball** A subreddit where football fans reacting to National Football League (NFL) news and trade fantasy tips.

**explainlikeimfive** A subreddit where users post questions about various topics and seek simple explanations.

**aww** A subreddit for cute and cuddly pictures such as puppies, bunnies, and babies.

**news** A subreddit for news articles, primarily but not exclusively, news relating to the United States and the rest of the World.

We divide the REDDIT-NEW dataset into training, validation, and test sets using the same division in Hamilton *et al.* (2017a). We do so because this allows us to have a direct comparison of our methods with the models presented in their research.

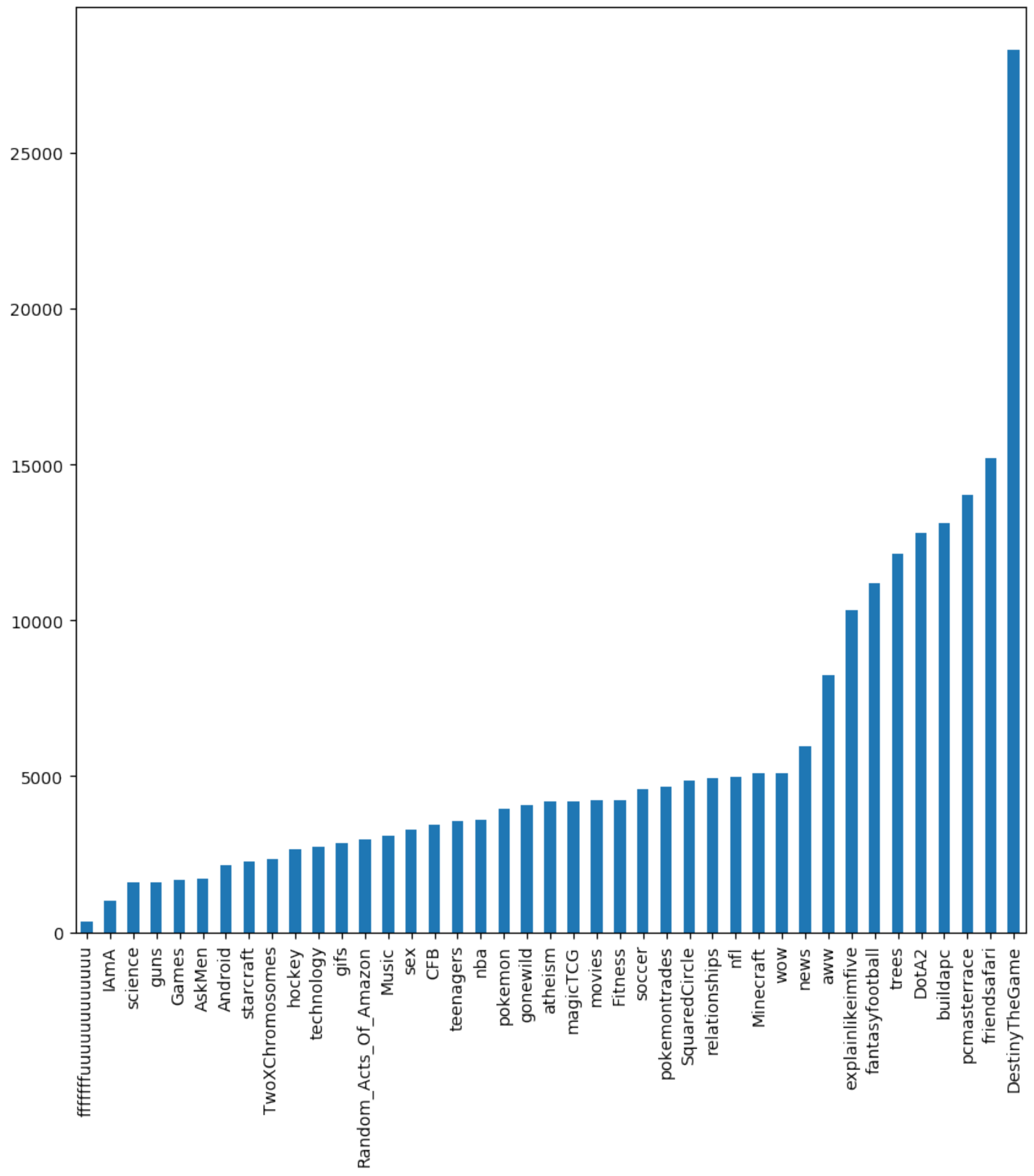


Figure 17.1: Distribution of posts among different communities in Reddit.

## 17.2 Methods

In this section, we present different methods for modeling the task of Reddit post classification. We divide methods into three parts: first, lexical modeling for the post content; second, network modeling of the underlying communication graphs using techniques discussed earlier in this thesis; third, propagation methods for the post labels.

### 17.2.1 Lexical Modeling

For post lexical features, we use the same lexical embeddings in the GRAPHSAGE-REDDIT dataset (subsection 3.3.2). Each post has a lexical feature of a concatenation of (i) the average embedding of the post title, and (ii) the average embedding of all the post's comments. Both are of-the-shelf 300-dimensional GloVe CommonCrawl word vectors (Pennington *et al.*, 2014). We use the same lexical features from GRAPHSAGE-REDDIT as we are interested in exploring different social network modeling techniques. This allows us to have a direct comparison with the performance of GraphSAGE reported in (Hamilton *et al.*, 2017a).

### 17.2.2 Network Modeling

In this section, we present network modeling methods. We use GraphSAGE (Hamilton *et al.*, 2017a) as a baseline since it is the study that proposed this task of post classification.

For a different social network modeling method, we use network features similar to those proposed in chapter 7. For this task, we construct two graphs: a bipartite graph of documents (posts) and users (section 5.2); and a user graph (section 5.3) such that we link users if they participated in a certain number of posts. Then, we extract social network features from these graphs. Note that, unlike email classification, the notion of sender and recipient is not applicable here in this task. We use a slightly different notation for the features discussed in chapter 7 as we do not have the distinction between senders and recipients here. We treat all users who participated in a post similarly. Below, we give more details about each graph and the social network features extracted from these graphs.

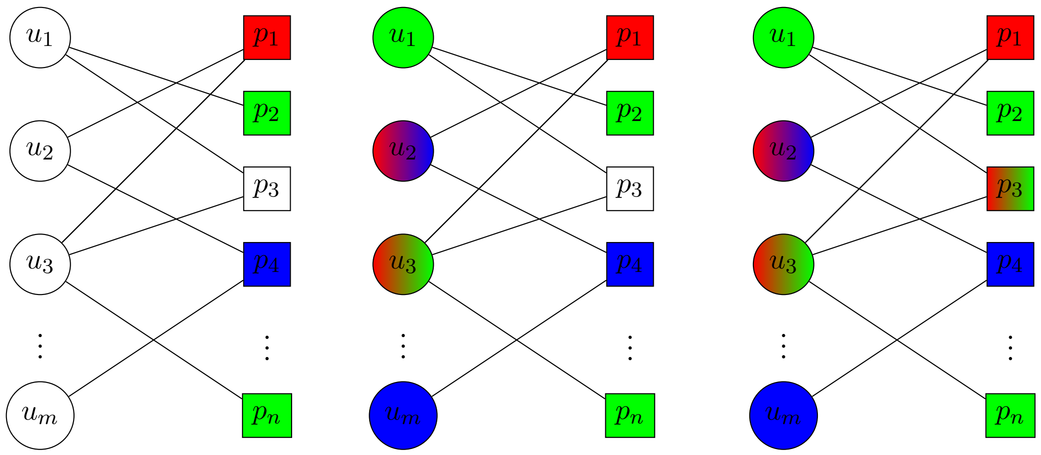
**Bipartite user-post graph** We construct a bipartite of users and posts such that the user $u$ is linked to the post $p$ if $u$ is the author of $p$ or has commented on $p$. Here, we do not distinguish between the post authors and other users who commented on that post. Table 17.2 below shows statistics for the bipartite user-post graph.

| | |
|---|---|
| Number of post nodes | 232,965 |
| Number of user nodes | 716,195 |
| Number edges | 4,067,211 |

Table 17.2: Bipartite user-post graph statistics.

**User graph** We construct a user graph (section 5.3) for Reddit post participants. We link users if they participate in the same posts – as the post author or comment on the post. However, since this does not reflect direct communication between users, linking all users if they participate in a single post might add noise, especially when there are posts with a large number of participants. Therefore, we set a threshold value $t$ for the number of participants in posts, and we include only posts with the number of participants below $t$.

**Social network features (net)** As we are interested in classifying posts (not users), we extract features for posts from both the user graph and the bipartite user-post graph as follows. From the bipartite graph, we extract features from the corresponding post nodes and the users who participate in the posts; and from the user graph, we extract features from users who participate in the posts. For features extracted from user nodes (in both the bipartite graph and user graph), we compute the average, max, and min for each user who is a participant in the post. Table 7.1 on page 74 shows the features we extract from the user graph and the bipartite graph. Note that, for features involving the sender, we instead compute them for all participants. For instance, when we compute *Jaccard's coefficient* score, we compute it for all pairs of participants in a given post; then, we compute max, min, and average. Additionally, we use only undirected graphs for this task.



(a) Initial step.

(b) Propagating labels from posts to users.

(c) Propagating labels from user to posts

Figure 17.2: The label propagation algorithm on the Reddit bipartite graph of posts and users. Colors represent different subreddits (communities). Each iteration performs the second and third steps.

### 17.2.3 Post Label Propagation Method

The users and the posts in which they participate (as authors or as they comment on) can be represented as a bipartite graph of posts and users such that the users and posts form two disjoint sets of nodes, and we link the user $u_i$ to the post $p_j$ if $u$ is the author of or commented on $p_j$.

We propose a simple approach adapted from Filippova (2012). The basic idea here is simply to propagate the multinomial distribution of communities from posts to users, then back to posts such that eventually, each post and user will have a multinomial distribution for communities: $\{c_1, c_2, \ldots c_n\}$ where $n$ denotes the number of communities in the dataset. Figure 17.2 illustrates the process.

We first construct a bipartite graph of users and posts including every post in the dataset such that a user is linked to a post if the user has posted a comment on that post, or the user is the author of that post. The idea here is to assign a multinomial distribution to every user and post in the bipartite graph.

Initially, labeled posts (in the train set) are assigned 1 for the corresponding community to which they belong; and 0 for other communities. Users and unlabeled posts start with 0 probabilities for the multinomial distribution of all communities.

Then, we propagate the labels from posts to users, then, from users to posts in $k$ iteration. At each iteration, we first propagate the distribution of communities to users through the edges in the user-post bipartite graph. For all posts linked to user $u_i$, the multinomial distribution for users and communities is the normalized probabilities propagated from the posts linked to users. Then, similarly, we propagate back the multinomial distributions from users to posts. We use two modes for the propagation process: reset and no-reset.

**reset mode** With reset mode, we reinitialize the community distribution for labeled examples by re-assigning them their initial true values at the begging of the next iteration.

**no-reset mode** Without reset (no-reset), we do not reset the labels of the labeled examples. So they have their propagated values instead of their true ones for the next iteration.

After $k$ iterations, we have a multinomial distribution for each post; we can use these probabilities in two ways: first, to predict the post community, we assign the community with the highest propagated probability to a post; second, we can use the probability distribution values as features for a standard machine learning classifier.

#### 17.2.3.1 Probabilistic Prediction

Following the approach presented in Filippova (2012), given the propagated distributions for unlabeled posts, we can predict their labels by simply finding the community with the highest propagated probability.

#### 17.2.3.2 Propagated Distributions as Features for ML Classifiers

Instated of predicting the post's class (community) directly by using the highest probability of the multinomial distribution described above, we propose a novel method for using the propagated distributions of posts as features for a machine learning classifier. Recall that after the propagation process, a post $p_i$ will have probability values for each community $\{p_{i,1}, p_{i,2}, \ldots, p_{i,m}\}$. We use these propagated probabilities as features for standard machine learning models. Also, we also do an experiment in which we add lexical features to these propagated probabilities for a standard machine learning classifier. Note that, for the *reset mode*, we reset the labels for the training posts only during intermediate iterations, but not for the final iteration. This will give us a probability distribution for the training example instead of 1 for the true class and 0 for other classes.

## 17.3 Experiments and Results

In this section, we present results for different experiments. We first start with showing results for the propagation method in subsection 17.2.3. Then, we show results for different classifiers with different features and settings. We are interested in optimizing the micro F-1 score, and we use it here to report the performance for different experiments. We use the test set for the evaluation. Table 17.3 summarizes the results.

### 17.3.1 Number of Iterations for Propagation

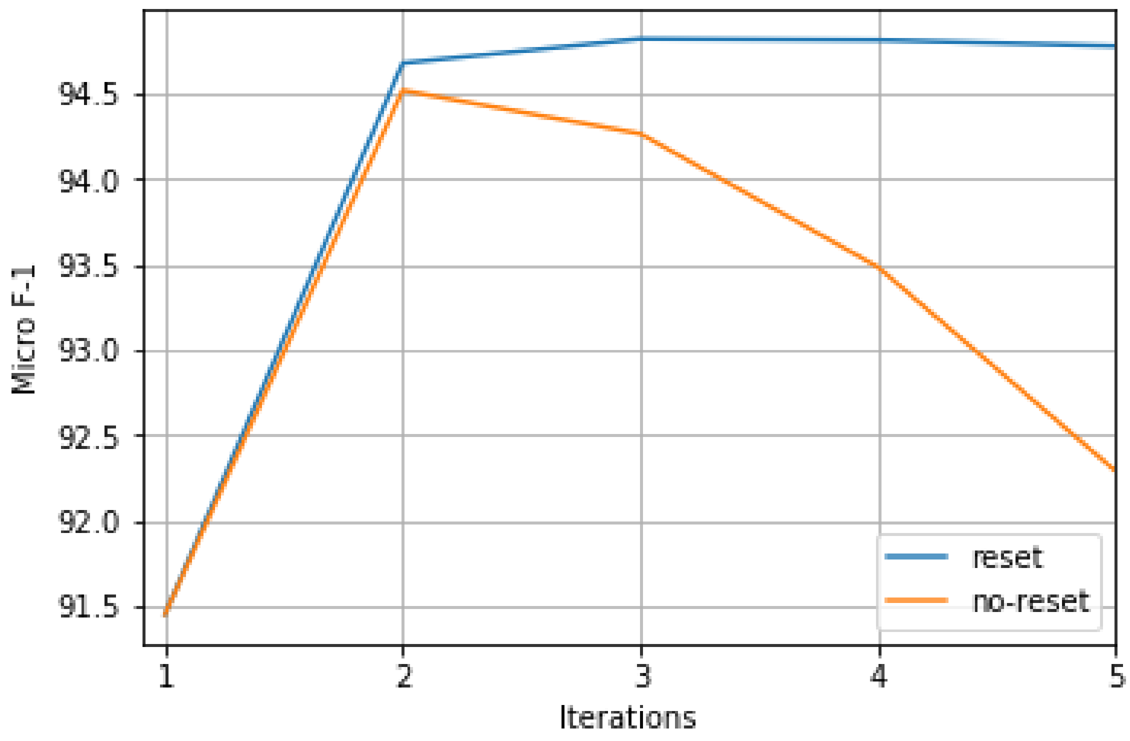


Figure 17.3: Micro F-1 score for the simple propagation algorithm with different number of iterations. We show the performance using two modes: reset, where we assign the train examples their actual distribution; no-reset, where we do not.

In the first set of experiments, we try different numbers of iterations $t$ for the community distribution propagation algorithms described in subsection 17.2.3. Figure 17.3 shows the results for different iterations using the two modes: **reset** and **no-reset**. We observe that when using the reset mode, the performance stabilizes after the second iteration and peaks at the third iteration. While when using the non-reset mode, the performance starts to drop after the second iteration. For further experiments, we use our best propagation setting (i.e., 3 iterations with rest).

### 17.3.2 Machine Learning Approach

In this section, we discuss experiments on the post classification task using machine learning classifiers. For lexical modeling, we experiment with different classifiers: logistic regression (LogReg), support vector machines (SVMs), and feedforward neural network (NN). For SVMs, we experiment with both a linear and an RBF kernel; we use $C \in \{0.1, 1, 5, 10, 100\}$ for both SVMs and LogReg classifiers. We use the implementation of SVMs and LogReg as provided in sklearn. For neural networks, we use a two-layer MLP with *ReLu* activation function in the hidden layers; for the non-linearity activation on the output layer, we use the sigmoid function and the binary cross-entropy as the loss function. We use the Keras framework (Chollet and others, 2015) for the implementation of neural networks, and we use the Adam optimizer (Kingma and Ba, 2014) with the default parameters as provided by Keras. We also show results for GraphSAGE, where we feed it with a graph without any edges (GraphSAGE-lex). This allows us to see the effect of GraphSAGE network modeling over its lexical modeling. Note that, without the network structure, GraphSAGE behaves as a feed-forward neural network. We use the GraphSAGE tensorflow implementation with the default hyper-parameter values for Reddit. [2,3] Table 17.3 shows a summary of different experiments with different methods.

We first replicate the best result of the GraphSAGE model on this task reported in Hamilton *et al.* (2017a); we observe that we have slightly different numbers than those reported there. Then, for GraphSAGE-lex, we observe that it performs almost the same as other lexical classifiers that are trained on the lexical features only. The best lexical classifier is SVM with an RBF kernel.

We observe that adding our network features (section 17.2.2) to the SVM-lex classifier boosts

[2] `https://github.com/williamleif/GraphSAGE`

[3] `https://github.com/williamleif/GraphSAGE/blob/master/eval_scripts/reddit_eval.py`

| Method | Micro F-1 |
|---|---|
| GraphSAGE | 94.85 † |
| GraphSAGE-lex | 71.45 |
| SVM-lex | 72.03 |
| Feedforward NN-lex | 71.51 |
| | |
| SVM-lex+net | 78.78 |
| **Simple Propagation (reset)** | |
| PROP-1 iteration | 91.45 |
| PROP-2 iterations | 94.63 |
| PROP-3 iterations | 94.82 |
| PROP with Classifier | |
| LogReg+PROP-3 | 95.30 |
| LogReg+PROP-3+Lex | **96.07** |

Table 17.3: The micro F-1 scores for different methods on Reddit post classification. †: we replicate best results reported in Hamilton *et al.* (2017a), it is slightly different than the number they reported. PROP refers to the propagation method.

the performance, but it has much worse performance than GraphSAGE (with the post-to-post network).

In the second box in Table 17.3, we show results for a probabilistic classifier that predicts the class for a given post using the propagated distributions (subsubsection 17.2.3.1).

In the third box, we show results for a simple machine learning model (i.e., LogReg) fed with the probability scores propagated using the algorithm discussed in subsection 17.2.3. The last line of Table 17.3 shows the result of a logistic regression classifier trained on the propagated distribution as features in concatenation with the lexical features.

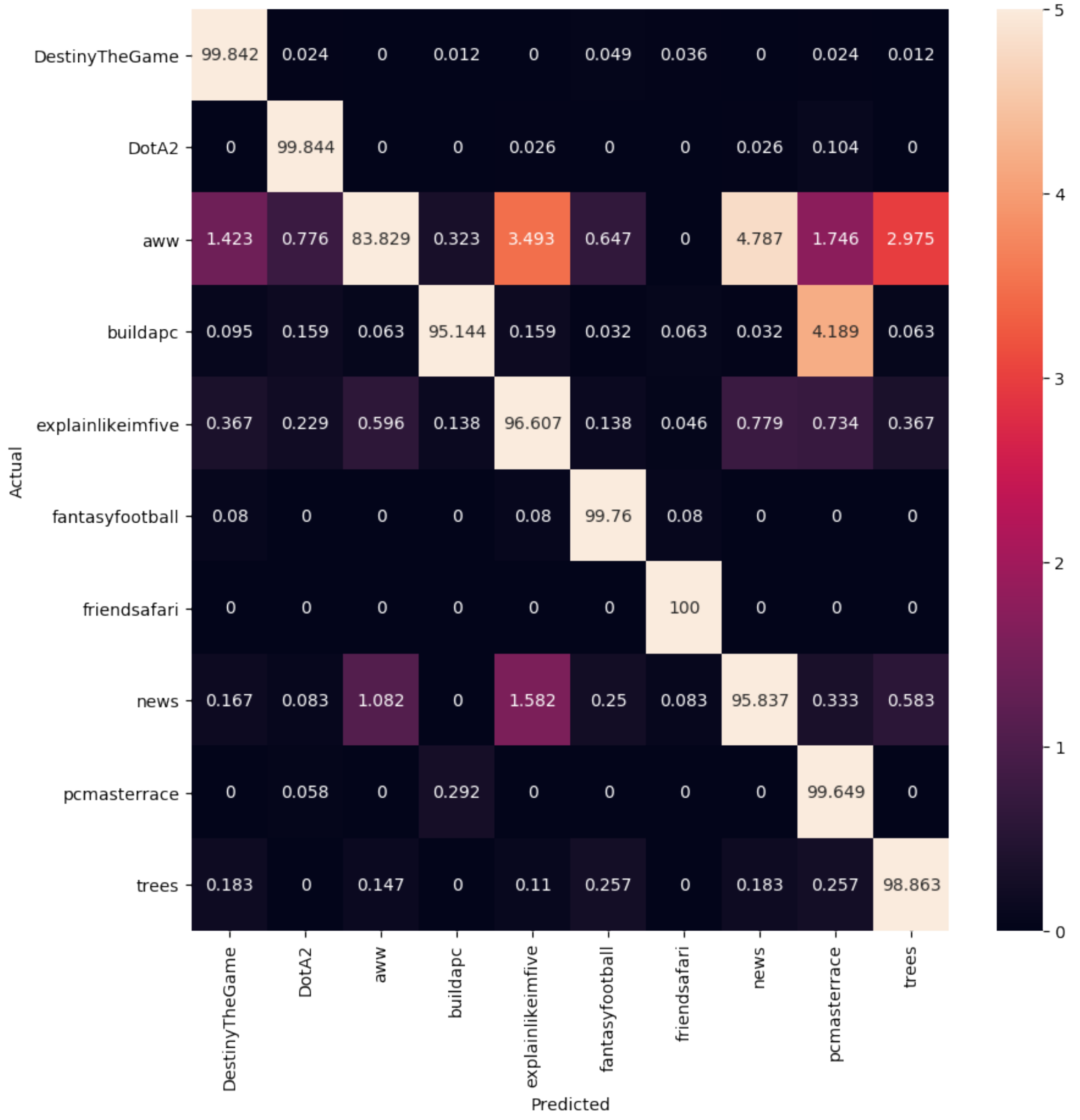
The results show that using the propagation method with a simple probabilistic classifier predicting the community with the highest probability performs better than any previous model (except GraphSAGE). Note that this method is performed without any lexical content. Additionally, when we train a simple machine learning classifier (i.e., Logistic Regression) on the propagated probability distribution, we get a further boost on the performance (95.30 for micro F-1). Finally, our best model is combining lexical features with the learned propagated distribution as features.

To determine whether the performance improvement of our best model (i.e., LogReg+PROP+-lex) over GraphSAGE is statistically significant, we use the non-parametric Wilcoxon Signed-Rank Test (Sidney, 1957) on pairs of the micro F-1 scores of different runs using 10-fold cross-validation. The improvement of LogReg+PROP+lex is indeed statistically significant at $p < 0.01$.

Although our proposed social network features (section 17.2.2) help in improving the classification performance, they still do not help as much as in other tasks. Also, for this task, Reddit post classification, GraphSAGE model performs much better than our proposed social network features. Our explanation is that these features capture dyadic relations, which are not helpful for this particular task (Reddit post classification), as this task involves community of users rather than dyadic relations. However, using a simple approach (i.e., label propagation) performs better than a state-of-art model for this task.

### 17.3.3 Error Analysis

In this section, we perform some error analysis on the predictions. Although our best model has a high performance on most classes, it still underperforms on some classes. Figure 17.4 shows the confusion matrix for the top 10 communities. The confusion matrix shows that the subreddit "aww" is the most wrongly predicted class among other communities. This might be due to the nature of the contents for the community, as users usually post pictures with a few or no lexical content.



| | DestinyTheGame | DotA2 | aww | buildapc | explainlikeimfive | fantasyfootball | friendsafari | news | pcmasterrace | trees |
|---|---|---|---|---|---|---|---|---|---|---|
| DestinyTheGame | 99.842 | 0.024 | 0 | 0.012 | 0 | 0.049 | 0.036 | 0 | 0.024 | 0.012 |
| DotA2 | 0 | 99.844 | 0 | 0 | 0.026 | 0 | 0 | 0.026 | 0.104 | 0 |
| aww | 1.423 | 0.776 | 83.829 | 0.323 | 3.493 | 0.647 | 0 | 4.787 | 1.746 | 2.975 |
| buildapc | 0.095 | 0.159 | 0.063 | 95.144 | 0.159 | 0.032 | 0.063 | 0.032 | 4.189 | 0.063 |
| explainlikeimfive | 0.367 | 0.229 | 0.596 | 0.138 | 96.607 | 0.138 | 0.046 | 0.779 | 0.734 | 0.367 |
| fantasyfootball | 0.08 | 0 | 0 | 0 | 0.08 | 99.76 | 0.08 | 0 | 0 | 0 |
| friendsafari | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 0 | 0 |
| news | 0.167 | 0.083 | 1.082 | 0 | 1.582 | 0.25 | 0.083 | 95.837 | 0.333 | 0.583 |
| pcmasterrace | 0 | 0.058 | 0 | 0.292 | 0 | 0 | 0 | 0 | 99.649 | 0 |
| trees | 0.183 | 0 | 0.147 | 0 | 0.11 | 0.257 | 0 | 0.183 | 0.257 | 98.863 |

Figure 17.4: Confusion matrix for the 10 largest communities for Reddit. For the communities' description, refer to section 17.1 on page 167. We show numbers as percentages.

**General communities** We observe that more general communities such as the "news" community and "explainlikeamfive" are the most wrongly predicted label.

**Similar communities** We observe that for some communities, the classifier predicts wrong communities that have similar topics. For instance, “buildapc” and “pcmasterrace” are similar communities, and we observe that the classifier confuses these two classes most frequently.

## 17.4 Conclusion

We have shown in this chapter different methods for Reddit post classification. We focus on modeling the Reddit posts and users as a social network using various models and different graph structures. We found that lexical features for posts do not perform well for this task. Additionally, our proposed social network features, which help in other tasks discussed in this thesis, improve the classification over lexical features only, but they underperform other network models such as GraphSAGE by a large margin.

We found that a simple approach such as the label propagation method in subsection 17.2.3 outperforms sophisticated models such as GraphSAGE. Even without looking at the lexical content for the posts, we can achieve a very high performance using this simple approach.

We conclude that for some applications, a simpler but task-specific method can perform better than complex and general-purpose graph models such as GraphSAGE. Additionally, our proposed social network features can capture dyadic relations, but they fail to capture communities as this task (i.e., Reddit post classification) is defined by communities rather than dyadic relations. For future work, a number of extensions and potential improvements are possible, such as extracting social network features that capture community information rather than dyadic relations.

# Chapter 18

# Conclusion

The main contribution of this thesis is the introduction of social network techniques for improving text classification tasks. Specifically, we proposed social network features extracted from different graph representations for the communication networks. We used these features with different machine learning models on different text classification tasks. As our main task, we choose email classification into business and personal. In addition, we apply our methods to three other applications:

- Overt Display of Power in email detection.
- Predicting Hierarchical Power in email.
- Reddit post classification.

In this chapter, we first summarize the contributions presented in this thesis in section 18.1, and we discuss limitations and future work in section 18.2.

## 18.1 Summary of Contributions

- We have collected large datasets of emails and annotated them with fine-grained business and personal labels. These datasets are based on two widely available email corpora: Enron and Avocado. We presented these datasets with the details of annotations in chapter 4.

- We proposed different graph structures to represent the communication network for documents and users. We presented these graph structures in chapter 5. We used these graphs for various text classification tasks.

- We conducted social network analysis on graphs induced from the datasets annotated with business and personal labels. We analyzed the induced personal and business sub-networks using different SNA measures, and we showed in chapter 6 that the two networks have different properties using these measures.

- We proposed a variety of social network features extracted from different graph structures representing the underlying social network of communication for both users and documents. This way, we can use social network information for document classification without having to explicitly model different graphs separately using graph models. We discussed these social network features in chapter 7.

- We applied our methods on different applications and showed that adding social network information to machine learning models improves the classification performance over models that have access only to the textual content of documents. We compared our proposed hand-engineered social network features with a state-of-the-art graph embedding model, and our model outperforms it on three tasks. Particularly, our proposed features outperform GraphSAGE on the email tasks: classification into business and personal (Part II), overt display of power detection (chapter 15), and hierarchical power detection (chapter 16). The fourth task (chapter 17) turns out to be a different kind of problem than the other tasks as it does not involve "dyadic" relations. However, for this task, we proposed a different and relatively simple method, and it outperforms GraphSAGE.

- We also proposed an extension of GraphSAGE to heterogeneous bipartite graphs that outperforms the ordinary GraphSAGE for the task of email classification into business and personal (section 13.2).

The experiments we have conducted in this dissertation show that our proposed models improve the classification tasks.

## 18.2 Limitations and Future Work

There are many future directions to take further the research presented in this thesis. We summarize some of the major directions below:

### 18.2.1 Exploring New Genres, Domains, and Applications

In this thesis, we studied two genres: email and Reddit posts. For email, we used two corpora: Enron and Avocado. We applied our models on three email applications: email classification into business and personal; overt display of power detection; and hierarchical power detection. For Reddit, we showed models that incorporate information from the social network for Reddit post classification. Our proposed social network features improve the email classification tasks, and they outperform graph embedding models. For Reddit, although our proposed social network features improve the classification performance over models that have access only to the textual content of Reddit posts, their performance is relatively low in comparison with the other proposed methods in chapter 17.

We concluded that the Reddit task turned out to be a different kind of problem than the other tasks as it does not involve “dyadic” relations. It is not clear whether the findings from our study carries over to other genres in which they involve dyadic relations.

One way to extend the work presented in this thesis is by applying the presented models to other genres and applications. An interesting application for future work would be the task of document-level sentiment analysis of tweets on Twitter, similar to work by Tan *et al.* (2011). On Twitter, there are variations of underlying social networks, for example: the follower/followee network, and the mention network. These variations allow us to exploit social network information as different networks carry different types of social interactions.

Another direction for future work is to apply methods presented in this thesis on another language – other than English, especially morphologically complex languages such as Arabic. Previous work on sentiment analysis has shown that language families and morphological complexity play an important role in the performance of cross-lingual sentiment transfer models Farra (2019). Future work should consider incorporating social network information for cross-lingual sentiment analysis. We have shown in Part II that incorporating social network information helps when we train on emails from one company (i.e., Enron) and evaluate on emails from another company (i.e., Avocado). An interesting future direction is to study if incorporating social networks improves the performance when we train on one language and evaluate on another. We have worked on Arabic dialect morphological analysis (Alshargi *et al.*, 2019; Habash *et al.*, 2018), and we plan to apply methods presented in this thesis to Arabic sentiment analysis on Twitter. Specifically, we plan to study the effect of cross-lingual and cross-network transfer learning for sentiment analysis on different Arabic dialects.

### 18.2.2 Applying New Methodologies

Another way to take the work presented in this thesis further is by applying new methods for incorporating social network information. We presented models using hand-engineered social network features and compared them with a state-of-the-art graph embedding model, namely, GraphSAGE. We showed that our social network features outperform GraphSAGE on the three email tasks, while on Reddit, another simple method that propagates labels on graphs outperforms GraphSAGE. We showed a simple way to extend GraphSAGE to bipartite graphs by constructing different aggregates and encoders for different node types (i.e., user nodes and document nodes).

A possible future direction is to explore new network embedding models by adapting techniques from previous work for incorporating information from different networks. We have shown in section 6.1 that the business and personal networks have different properties, and we believe that incorporating information from these sub-networks can improve the classification performance. Ni *et al.* (2018) propose to use multiple related social networks to learn embeddings through a joint learning schema such that similar nodes in different networks have similar embeddings. They simply add pairwise regularizers to the single-network embedding objective function. Another study, (Xu *et al.*, 2017), introduces a harmonious embedding matrix to transform the latent features from one space into another space.

Also, we showed in section 6.2 that the distribution of business and personal emails differs among clusters. However, in this thesis, we did not investigate incorporating information from clusters in the experimental studies. One possible direction is to use techniques from the literature for incorporating information from clusters for better social network representations. Rozemberczki *et al.* (2018) propose a $k$-means like cost function to enforce nodes with high neighborhood overlap to have similar representations. We plan to investigate these techniques to improve our models.

In this thesis, we have focused on models that incorporate social network information for different text classification tasks. However, one way to improve lexical modeling is by using Contextual Word Representations instead of the context-free word embedding models we have used in this thesis. Recent developments in Contextual Word Representations models such as BERT (Devlin *et al.*, 2018) have led to significant improvements for several natural language processing tasks. Future work should consider improving the lexical features by using pre-training BERT (or other Contextual Word Representation models) and fine-tuning in our models.

# Bibliography

Amjad Abu-Jbara, Ben King, Mona Diab, and Dragomir Radev. Identifying opinion subgroups in Arabic online discussions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 829–835, 2013.

Apoorv Agarwal, Adinoyi Omuya, Aaron Harnly, and Owen Rambow. A comprehensive gold standard for the Enron organizational hierarchy. In *50th Annual Meeting of the Association for Computational Linguistics*, page 161, 2012.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. *ICWSM*, 270:2012, 2012.

Nikolaos Aletras and Benjamin Paul Chamberlain. Predicting Twitter user socioeconomic attributes with network and language information. *arXiv preprint arXiv:1804.04095*, 2018.

Sakhar Alkhereyf and Owen Rambow. Email classification incorporating social networks and thread structure. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1336–1345, 2020.

Faisal Alshargi, Shahd Dibas, Sakhar Alkhereyf, Reem Faraj, Basmah Abdulkareem, Sane Yagi, Ouafaa Kacha, Nizar Habash, and Owen Rambow. Morphologically annotated corpora for seven Arabic dialects: Taizi, Sanaani, Najdi, Jordanian, Syrian, Iraqi and Moroccan. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 137–147, 2019.

Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Harold Borko and Myrna Bernick. Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162, 1963.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv preprint physics/0608255*, 2006.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On finding graph clusterings with maximum modularity. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 121–132. Springer, 2007.

Hongyun Cai, Vincent W Zheng, and Kevin Chang. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

Kathleen M Carley, Dave Columbus, Matt DeReno, Jeff Reminga, and Il-Chul Moon. Ora user's guide 2008. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2008.

Dorwin Cartwright and Frank Harary. Structural balance: a generalization of Heider's theory. *Psychological review*, 63(5):277, 1956.

Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386. ACM, 2017.

Francois Chollet et al. Keras, 2015.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. TopicRNN: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016.

Jana Diesner and Kathleen M Carley. Exploration of communication networks from the Enron email corpus. In *SIAM International Conference on Data Mining: Workshop on Link Analysis, Counterterrorism and Security, Newport Beach, CA*, pages 3–14. Citeseer, 2005.

John Dobson. Enron: The collapse of corporate culture. In *Enron and World Finance*, pages 193–205. Springer, 2006.

Noura Farra. *Cross-Lingual and Low-Resource Sentiment Analysis*. PhD thesis, Columbia University, 2019.

BJ Field. Towards automatic indexing: Automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation*, 1975.

Katja Filippova. User demographics and language in an implicit social network. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1478–1488. Association for Computational Linguistics, 2012.

Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

David Graus, David Van Dijk, Manos Tsagkias, Wouter Weerkamp, and Maarten De Rijke. 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*, pages 1079–1082. ACM, 2014.

WA Gray and AJ Harley. Computer assisted indexing. *Information Storage and Retrieval*, 7(4):167–174, 1971.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

Wojciech Gryc and Karo Moilanen. Leveraging textual sentiment analysis with social network modelling. *From Text to Political Positions: Text analysis across disciplines*, 55:47, 2014.

Lin Gui, Yu Zhou, Ruifeng Xu, Yulan He, and Qin Lu. Learning representations from heterogeneous network for sentiment classification of product reviews. *Knowledge-Based Systems*, 124:34–45, 2017.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, et al. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

William L Hamilton. *Representation Learning Methods for Computational Social Science*. PhD thesis, Stanford University, 2018.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Ahmed Hassan, Amjad Abu-Jbara, and Dragomir Radev. Detecting subgroups in online discussions by modeling positive and negative relations among participants. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 59–70. Association for Computational Linguistics, 2012.

Swapnil Hingmire, Sandeep Chougule, Girish K Palshikar, and Sutanu Chakraborti. Document classification by topic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 877–880, 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jun Hu, Rebecca J Passonneau, and Owen Rambow. Contrasting the interaction structure of an email and a telephone corpus: A machine learning approach to annotation of dialogue function units. In *Proceedings of the SIGDIAL 2009 Conference*, pages 357–366, 2009.

Sanaz Jabbari, Ben Allison, David Guthrie, and Louise Guthrie. Towards the Orwellian nightmare: separation of business and personal emails. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 407–411. Association for Computational Linguistics, 2006.

Ling Jian, Jundong Li, and Huan Liu. Toward online node classification on streaming networks. *Data Mining and Knowledge Discovery*, 32(1):231–257, 2018.

Suqi Jiang, Jason Lewris, Michael Voltmer, and Hongning Wang. Integrating rich document representations for text classification. In *2016 IEEE Systems and Information Engineering Design Symposium (SIEDS)*, pages 303–308. IEEE, 2016.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, pages 301–312. IBM Corp., 2011.

Jon M Kleinberg. Hubs, authorities, and communities. *ACM computing surveys (CSUR)*, 31(4es):5, 1999.

Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *CEAS*, 2004.

Aleksander Kolcz. Local sparsity control for naive Bayes with extreme misclassification costs. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 128–137, 2005.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

Vinodh Krishnan and Jacob Eisenstein. ”You’re Mr. Lebowski, I’m the Dude”: Inducing address term formality in signed social networks. *arXiv preprint arXiv:1411.4351*, 2014.

Andrew Lampert, Robert Dale, and Cécile Paris. Requests and commitments in email are more complex than you think: Eight reasons to be cautious. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 64–72, 2008.

J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE, 2015.

Jiangming Liu and Yue Zhang. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577, 2017.

Edward Loper and Steven Bird. NLTK: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

Yukun Ma, Haiyun Peng, and Erik Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *Aaai*, pages 5876–5883, 2018.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

Melvin Earl Maron. Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417, 1961.

Justin Christopher Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. In *Third international AAAI conference on weblogs and social media*, 2009.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Tanushree Mitra and Eric Gilbert. Have you heard?: How gossip flows through workplace email. In *ICWSM*, 2012.

Mark EJ Newman. *Networks: an introduction*, page 224. Oxford University Press, 2011.

Jingchao Ni, Shiyu Chang, Xiao Liu, Wei Cheng, Haifeng Chen, Dongkuan Xu, and Xiang Zhang. Co-regularized deep multi-network embedding. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 469–478. International World Wide Web Conferences Steering Committee, 2018.

Douglas Oard, William Webber, David Kirsch, and Sergey Golitsynskiy. Avocado research email collection. *Philadelphia: Linguistic Data Consortium*, 2015.

Benjamin Pachev and Benjamin Webb. Fast link prediction for large networks using spectral embedding. *Journal of Complex Networks*, 6(1):79–94, 2017.

Sebastian Palus, Piotr Brodka, and Przemyslaw Kazienko. Evaluation of organization structure based on email interactions. *International Journal of Knowledge Society Research (IJKSR)*, 2(1):1–13, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Bryan Perozzi and Steven Skiena. Exact age prediction in social networks. In *Proceedings of the 24th International Conference on World Wide Web*, pages 91–92. ACM, 2015.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Kelly Peterson, Matt Hohensee, and Fei Xia. Email formality in the workplace: A case study on the Enron corpus. In *Proceedings of the Workshop on Languages in Social Media*, pages 86–95. Association for Computational Linguistics, 2011.

Vinodkumar Prabhakaran and Owen Rambow. Predicting power relations between participants in written dialog from a single thread. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 339–344, 2014.

Vinodkumar Prabhakaran, Huzaifa Neralwala, Owen Rambow, and Mona T Diab. Annotations for power relations on email threads. In *LREC*, pages 806–811, 2012.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. Predicting overt display of power in written dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 518–522, 2012.

Vinodkumar Gourinivas Prabhakaran. *Social Power in Interactions: Computational Analysis and Detection of Power Relations*. PhD thesis, Columbia University, 2015.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

Karthika D Renuka and P Visalakshi. Latent semantic indexing based svm model for email spam classification. 2014.

Ryan Rowe, German Creamer, Shlomo Hershkop, and Salvatore J Stolfo. Automated social hierarchy detection through email network analysis. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 109–117, 2007.

Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. *arXiv preprint arXiv:1802.03997*, 2018.

Ilya Safro. Lecture 11: Clustering, transitivity, spectral methods i. Lecture Notes, February 2014.

Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. Technical report, Cornell University, 1973.

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Maya Sappelli, Gabriella Pasi, Suzan Verberne, Maaike de Boer, and Wessel Kraaij. Assessing e-mail intent and tasks in e-mail messages. *Information Sciences*, 358:1–17, 2016.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

Jitesh Shetty and Jafar Adibi. The Enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California*, 4(1):120–128, 2004.

SIEGEL Sidney. Nonparametric statistics for the behavioral sciences. *The Journal of Nervous and Mental Disease*, 125(3):497, 1957.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1397–1405. ACM, 2011.

Takenobu Tokunaga and Iwayama Makoto. Text categorization based on weighted inverse document frequency. In *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ*. Citeseer, 1994.

Bart Van Looy and Tom Magerman. Using text mining algorithms for patent documents and publications. In *Springer Handbook of Science and Technology Indicators*, pages 929–956. Springer, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. Inferring user political preferences from streaming communications. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 186–196, 2014.

Min-Feng Wang, Meng-Feng Tsai, Sie-Long Jheng, and Cheng-Hsien Tang. Social feature-based enterprise email classification without examining email contents. *Journal of Network and Computer Applications*, 35(2):770–777, 2012.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, pages 203–209, 2017.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. SHINE: signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 592–600. ACM, 2018.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. Sentiment analysis by capsules. In *Proceedings of the 2018 world wide web conference*, pages 1165–1174, 2018.

Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S Yu. Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1611–1619. International World Wide Web Conferences Steering Committee, 2017.

Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. Exploiting social network structure for person-to-person sentiment analysis. *arXiv preprint arXiv:1409.2450*, 2014.

Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 741–749. ACM, 2017.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

Jen-Yuan Yeh and Aaron Harnly. Email thread reassembly using similarity matching. 2006.

Shinjae Yoo, Yiming Yang, Frank Lin, and Il-Chul Moon. Mining social networks for personalized email prioritization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 967–976. ACM, 2009.

Bei Yu, Stefan Kaufmann, and Daniel Diermeier. *Available at SSRN 1026925*, 2007.

# Appendix A

# List of Business and Personal Words

**Top business words in both Enron and Avocado** *changes*, *information*, *issue*, *meeting*, *please*, *review*, *thanks*

**Top business words in Enron but not in Avocado** *access*, *agreement*, *agreements*, *america*, *any*, *approval*, *assignment*, *attached*, *bank*, *call*, *comments*, *committee*, *conference*, *confidentiality*, *confirm*, *contract*, *contracts*, *copy*, *corp*, *counterparty*, *credit*, *deal*, *deals*, *document*, *documents*, *draft*, *ena*, *energy*, *enron*, *enrononline*, *eol*, *executed*, *fax*, *ferc*, *final*, *form*, *fyi*, *gas*, *guaranty*, *inc*, *isda*, *issues*, *language*, *letter*, *market*, *master*, *memo*, *notice*, *of*, *online*, *options*, *order*, *per*, *physical*, *pira*, *power*, *presentation*, *price*, *product*, *project*, *questions*, *report*, *request*, *responses*, *revised*, *sara*, *the*, *these*, *trading*, *transaction*, *update*, *version*, *weekly*

**Top business words in Avocado but not in Enron** *account*, *application*, *avocadoit*, *bug*, *build*, *customer*, *demo*, *file*, *files*, *hp*, *problem*, *ravi*, *release*, *server*, *siebel*, *support*, *test*, *testing*, *training*, *wireless*,

**Top personal words in both Enron and Avocado** *adorable*, *always*, *anyway*, *apartment*, *apt*, *around*, *arriving*, *ave*, *baby*, *bad*, *band*, *basketball*, *beach*, *beautiful*, *bed*, *beer*, *big*, *bigger*, *birthday*, *boat*, *born*, *boy*, *boys*, *bro*, *brochures*, *brother*, *brothers*, *buildings*, *bye*, *calender*, *car*, *cleaning*, *club*, *congratulations*, *corner*, *cute*, *dad*, *daughter*, *dinner*, *downtown*, *drink*, *drinking*, *drinks*, *family*, *farewell*, *feeling*, *food*, *forget*, *fort*, *friend*, *friends*, *fun*, *funny*, *game*, *gate*, *giants*, *gift*, *girlfriend*, *girls*, *god*, *golf*, *gonna*, *great*, *gretchen*, *ha*, *hang*, *happy*, *hell*, *hello*, *hey*, *holiday*, *home*, *hope*, *hotels*, *house*, *how*, *huh*, *inn*, *instant*, *kids*, *kitty*, *lesson*, *life*, *little*, *lives*, *long*, *lots*, *love*, *lunch*, *man*, *married*, *maybe*, *mom*, *my*, *nap*, *nice*, *night*, *nights*, *parents*, *party*, *pictures*, *play*, *pool*, *pot*, *pray*, *prayer*, *really*, *reservations*, *ride*, *roommate*, *sandy*, *sat*, *saturday*, *say*, *she*, *sister*, *sisters*, *ski*, *sleep*, *smoke*, *so*, *son*, *sounds*, *stay*, *staying*, *surgery*, *sweet*, *tahoe*, *tail*, *tell*, *thai*, *thought*, *tournament*, *trip*, *tv*, *unbelievable*, *uncle*, *vegas*, *way*, *wedding*, *weekend*, *well*, *whereabouts*, *wine*, *wish*, *wonderful*, *worse*, *xmas*, *ya*, *you*

**Top personal words in Enron but not in Avocado** *abel*, *about*, *absolutely*, *accenture*, *acquainted*, *actually*, *ad*, *adage*, *address*, *adios*, *after*, *afterwork*, *ahoy*, *ain*, *alabama*, *alice*, *alive*, *all*, *along*, *alpha*, *alright*, *am*, *amazing*, *amy*, *antonio*, *anybody*, *anyday*, *appetizer*, *appointment*, *appt*, *arms*, *arrive*, *arrives*, *artistid*, *asbury*, *assault*, *atcha*, *atl*, *att*, *aw*, *away*, *awesome*, *baaad*, *babes*, *back*, *background*, *backyard*, *balled*, *bamboozled*, *baseball*, *bash*, *batting*, *bb*, *bbq*, *bedroom*, *beds*, *bedtime*, *beg*, *bell*, *ben*, *berkeleyan*, *bern*, *bet*, *betas*, *beth*, *better*, *bible*, *birthdays*, *bistro*, *bitches*, *bkb*, *blackjack*, *bles*, *blockbuster*, *blocked*, *blowing*, *bluegrass*, *blues*, *bn*, *books*, *boone*, *booze*, *boring*, *bosnia*, *bounced*, *bout*, *bq*, *bragg*, *breakfast*, *brennan*, *briant*, *bring*, *brock*, *brokering*, *bub*, *buddy*, *buffalo*, *bugging*, *bum*, *bums*, *burial*, *burned*, *bus*, *busy*, *but*, *buttons*, *byington*, *cab*, *cabo*, *caddy*, *cages*, *cake*, *camel*, *cameron*, *capri*, *capstone*, *cara*, *card*, *carribean*, *cartoon*, *cashish*, *cass*, *caterer*, *cbciii*, *cda*, *celebratory*, *cgi*, *character*, *chicken*, *chicks*, *children*, *china*, *chinese*, *christa*, *christmas*, *chron*, *cigarettes*, *cigars*, *cinderella*, *cleone*, *clobber*, *clock*, *clothes*, *coburn*, *coffee*, *coke*, *collaborated*, *college*, *colors*, *come*, *coming*, *comm*, *comparison*, *complaining*, *concert*, *conn*, *contest*, *contestants*, *continent*, *cook*, *cooking*, *corrupt*, *cotton*, *couch*, *count*, *crash*, *crazy*, *creative*, *cruise*, *cruises*, *cutie*, *daddy*, *dadisms*, *danetta*, *dannetta*, *darn*, *dating*, *davies*, *dc*, *deblah*, *decent*, *decides*, *dee*, *definitely*, *dental*, *departs*, *deserved*, *desperately*, *destination*, *dianne*, *didn*, *didnt*, *died*, *dierdre*, *directions*, *dirtbag*, *dirty*, *displays*, *diva*, *do*, *doin*, *doing*, *donald*, *doreen*, *down*, *dr*, *dressed*, *drinkers*, *driving*, *drow*, *drum*, *drunken*, *dry*, *dryspell*, *dual*, *dub*, *dying*, *eating*, *edan*, *ehud*, *eit*, *eklavya*, *elway*, *emily*, *en*, *enjoyed*, *entertainment*, *entex*, *envelope*, *equant*, *erica*, *errol*, *evan*, *evening*, *ever*, *everyone*, *everything*, *excited*, *excuse*, *exodus*, *facts*, *famous*, *fedex*, *feel*, *film*, *films*, *fishing*, *fla*, *flights*, *flowers*, *fly*, *flying*, *football*, *formation*, *fredianmichaela*, *friday*, *fro*, *fwd*, *gallop*, *ganjoo*, *garden*, *gardens*, *gates*, *gator*, *gators*, *gessner*, *get*, *getting*, *gioffre*, *glad*, *gleason*, *gm*, *go*, *goin*, *going*, *good*, *gore*, *got*, *grandma*, *gravy*, *gray*, *grayness*, *grill*, *grow*, *grumps*, *gt*, *guess*, *guffaw*, *gymnasium*, *handsome*, *hanging*, *hannaandersson*, *happily*, *hasta*, *he*, *hear*, *heard*, *hee*, *her*, *here*, *herself*, *hi*, *highwater*, *hilarious*, *hillel*, *hint*, *hip*, *holidays*, *hollis*, *holmes*, *honeymoon*, *hong*, *hoop*, *hop*, *hoping*, *hornswoggled*, *horse*, *hotel*, *hotmail*, *hour*, *houses*, *howard*, *hq*, *hts*, *hug*, *hurt*, *hwy*, *hydro*, *iaia*, *idc*, *ideosyncracies*, *impressive*, *indoors*, *inefficient*, *inheritance*, *inorcatid*, *interfirst*, *interx*, *ishtar*, *islands*, *ismail*, *it*, *jahnke*, *jam*, *jamming*, *jana*, *janette*, *janna*, *jaycreek*, *jealous*, *jeffs*, *jerseys*, *jewish*, *job*, *johnelle*, *jones*, *joni*, *jonny*, *jordan*, *just*, *kael*, *kai*, *kampy*, *kari*, *katie*, *kelley*, *kerouac*, *kick*, *kidding*, *kidnapped*, *kimzey*, *kisses*, *kitten*, *knee*, *know*, *kofi*, *kohli*, *kong*, *kyle*, *lacy*, *lake*, *lamar*, *lane*, *lari*, *laughed*, *lean*, *leandro*, *leather*, *leauge*, *lehmanium*, *lick*, *likes*, *limo*, *liz*, *lizzie*, *ll*, *lodge*, *longest*, *longhorns*, *lord*, *loud*, *lovely*, *low*, *lsu*, *lu*, *luncheon*, *lunching*, *lymph*, *macedonia*, *madeleine*, *madonna*, *magdalinweiss*, *majorcatid*, *mak*, *mandy*, *manicure*, *mansion*, *marie*, *maruti*, *massages*, *mathewsmith*, *maureen*, *maurice*, *mayo*, *mb*, *mccormick*, *meadow*, *meal*, *megan*, *meghan*, *mel*, *memorial*, *memories*, *mendocino*, *menina*, *mercruiser*, *merry*, *mess*, *messenger*, *metropolitan*, *miami*, *michaela*, *migrate*, *min*, *miquel*, *missouri*, *mistaken*, *mo*, *modest*, *molson*, *montrose*, *morn*, *morning*, *mother*, *motor*, *movies*, *much*, *mum*, *murli*, *museum*, *mx*, *myrtle*, *nagwani*, *nancy*, *napa*, *ncaa*, *neat*, *neglect*, *nelly*, *never*, *newest*, *next*, *nfl*, *nigeria*, *ninfa*, *nite*, *nodes*, *noe*, *nofx*, *norway*, *now*, *ob*, *occassion*, *octel*, *off*, *officializing*, *okc*, *okies*, *oklahoman*, *ol*, *old*, *oscar*, *ou*, *outfits*, *outing*, *overgrown*, *overrated*, *packed*, *pain*, *painful*, *palacios*, *pale*, *parent*, *partying*, *passover*, *pastor*, *pat*, *patandnora*, *patient*, *patio*, *paula*, *pauline*, *pbieraugel*, *penned*, *pepper*, *perfect*, *pet*, *philly*, *photos*, *pick*, *picking*, *picks*, *pics*, *picture*, *piggy*, *piss*, *pissed*, *pl*, *plane*, *planet*, *planning*, *playboy*, *playing*, *plaza*, *plz*, *poli*, *powell*, *powerful*, *ppl*, *ppp*, *practicing*, *prentice*, *presentable*, *presents*, *presidential*, *pretty*, *probably*, *problemo*, *professional*, *prosthetics*, *psychotic*, *punk*, *qb*, *quebec*, *race*, *racecarclub*, *races*, *radianz*, *rage*, *ragin*, *rain*, *ranch*, *randalls*, *rcr*, *rcrs*, *rebels*, *recovering*, *refaxing*, *references*, *remember*, *reminding*, *reminds*, *retired*, *right*, *ro*, *rolls*, *ronn*, *room*, *rooms*, *roscoe*, *rounds*, *route*, *rsvps*, *rug*, *ruined*, *ruppie*, *rustic*, *rusty*, *sailing*, *sam*, *san*, *sandeep*, *sc*, *schmidt*, *school*, *schwartz*, *scotsman*, *screened*, *seasons*, *seattle*, *see*, *sendoff*, *sept*, *serious*, *shalesh*, *shanna*, *shannon*, *sharing*, *sheas*, *sheryl*, *shopping*, *shortest*, *sick*, *sienna*, *skiiillz*, *skiing*, *sleeping*, *slow*, *smile*, *smithweb*, *smoking*, *smu*, *snowboard*, *society*, *someplace*, *something*, *somewhere*, *songs*, *soon*, *sooner*, *sounded*, *spaces*, *spare*, *speedway*, *spirit*, *spitting*, *springsteen*, *sprint*, *srm*, *stalin*, *star*, *steak*, *stefanie*, *sterndrive*, *stolen*, *story*, *streets*, *study*, *style*, *subject*, *subtract*, *sukran*, *sunday*, *supervised*, *supervising*, *supposed*, *surprisingly*, *survivor*, *sus*, *suz*, *swallowed*, *swamped*, *tables*, *taco*, *taker*, *talking*, *tall*, *tampa*, *tasting*, *tc*, *tee*, *televised*, *temptation*, *thanksgiving*, *there*, *thing*, *things*, *think*, *thinking*, *though*, *thoughtfulness*, *three*, *thur*, *thurs*, *thursdays*, *ticketmaster*, *tickets*, *tidying*, *time*, *tix*, *toasts*, *toe*, *together*, *tongs*, *too*, *took*, *tourney*, *trail*, *train*, *tree*, *treebeards*, *triem*, *true*, *trying*, *tuesdays*, *tulane*, *tunnel*, *turkey*, *turns*, *tux*, *tweed*, *u2*, *ughhh*, *undefeated*, *unsus-pecting*, *unveiling*, *up*, *upstream*, *val*, *valerius*, *vines*, *virginboy*, *vista*, *vmail*, *volleyball*, *vote*, *wabo*, *wags*, *wallet*, *wanderer*, *warning*, *was*, *wassup*, *watch*, *watching*, *went*, *what*, *whats*, *where*, *willy*, *winding*, *winner*, *women*, *wonder*, *work*, *worth*, *wouldn*, *xmsap*, *yah*, *yall*, *yao*, *yard*, *yeah*, *yipeeee*, *yo*, *yours*, *yourself*, *yr*, *zoo*

**Top personal words in Avocado but not in Enron** *aai*, *aboout*, *accomodation*, *accurate*, *acta*, *adithi*, *adjust*, *administrative*, *adult*, *afghanistan*, *age*, *aint*, *alexander*, *ali*, *alumni*, *aman*, *amma*, *ana*, *andale*, *anesthesia*, *angie*, *anguish*, *anil*, *anil_kumar*, *anna*, *announcement*, *annual*, *anun-ciando*, *apeice*, *appa*, *arkansas*, *arrived*, *arrrive*, *asian*, *asr*, *asst*, *asx*, *attacks*, *attorney*, *auc-tions*, *auden*, *aurora*, *aussie*, *avaiable*, *availble*, *babar*, *babe*, *babyshower*, *badfinger*, *bahadur*, *bail*, *baker*, *balcony*, *ball*, *banana*, *bar*, *barcelona*, *bartending*, *bball*, *beagle*, *beauty*, *bebe*, *becky*, *beers*, *beforehand*, *believed*, *bells*, *bhopal*, *birks*, *bitching*, *bleak*, *blocks*, *blonde*, *blow*, *blowfish*, *blue*, *bmw*, *bomb*, *bonfire*, *boradway*, *bottle*, *bought*, *boyfriend*, *braganza*, *brainstorming*, *bridges*, *brochure*, *bruno*, *bryce*, *bs*, *bummin*, *bunny*, *cabaret*, *calderon*, *camping*, *campout*, *cancer*, *caps*, *captain*, *care*, *carts*, *cathedral*, *celebrate*, *celebration*, *cellphone*, *chad*, *chal*, *chaplain*, *cheese*, *chef*, *chente*, *cheryl*, *chi*, *chill*, *chillin*, *chow*, *chrissie*, *christ*, *church*, *ciao*, *circle*, *city*, *closes*, *closet*, *cmkllp*, *cokes*, *cold*, *comic*, *comp*, *condo*, *confrm*, *congrats*, *congressman*, *conscience*, *contracting*, *cool*, *coop*, *cops*, *corresponded*, *cough*, *cousin*, *cracked*, *craddle*, *cramps*, *creim*, *cristian*, *cross*, *ctive*, *cub*, *curry*, *damn*, *damnit*, *dana*, *danbaca*, *darda*, *darren*, *dasar*, *davinwheels*, *day*, *daycare*, *dear*, *del*, *delayed*, *delhi*, *denver*, *depressing*, *deserve*, *destined*, *devout*, *dgreetings*, *diane*, *dick*, *dicks*, *diego*, *dining*, *diolette*, *dm*, *dmv*, *doctors*, *doe*, *dog*, *dokie*, *dole*, *dollar*, *don*, *dork*, *double-tree*, *doz*, *dreaming*, *dress*, *dude*, *dudes*, *dystrophy*, *ear*, *eat*, *eater*, *ebay*, *ecenasia*, *eileen*, *elisabeth*, *emanuelsf*, *endorsing*, *enjoy*, *enjoying*, *equals*, *equinox*, *eren*, *err*, *euro*, *evelyn*, *evil*, *exciting*, *ex-plosion*, *fag*, *fan*, *fancy*, *fart*, *father*, *faultline*, *faye*, *feast*, *federalist*, *felt*, *fiction*, *filling*, *fitness*, *flex*, *flores*, *flown*, *frail*, *freedom*, *fremont*, *fri*, *friendship*, *frightening*, *frogs*, *fuck*, *fuckers*, *fullcrm*, *fundraisers*, *gabriel*, *gala*, *ganeshchaturthi*, *gear*, *geocities*, *gifts*, *gimme*, *girl*, *girr*, *girrrrrl*, *golfing*, *google*, *goood*, *gov*, *government*, *grab*, *grade*, *graduation*, *gramercy*, *granada*, *grand*, *grandmother*, *greatness*, *greer*, *greeting*, *greetings*, *grumet*, *gujarati*, *gym*, *hadn*, *hair*, *halfway*, *halloween*, *handcuff*, *harumphhhh*, *harvard*, *hashanah*, *hayley*, *heart*, *heathen*, *hectic*, *hehe*, *helens*, *helloooooo*, *highway*, *hit*, *hmos*, *hoes*, *hofstra*, *home_fan*, *honey*, *hooptie*, *horoscope*, *hsneiderman*, *hsu*, *hugo*, *hugos*, *humor*, *hungry*, *hunting*, *husband*, *husky*, *iana*, *ibmin*, *idiots*, *im*, *image888*, *industrial*, *inexperienced*, *infection*, *inhouse*, *innings*, *innocent*, *insurance*, *intensive*, *inter*, *intermediate*, *invitation*, *invite*, *isaac*, *isbn*, *israel*, *itchy*, *jail*, *jazz*, *jazzy*, *jeez*, *jerky*, *jesus*, *jillian*, *jims*, *jj*, *johnny*, *joyce*, *joys*, *jpeg*, *jyotsna*, *k888*, *kane*, *katya*, *kings*, *kittu*, *kobza*, *koenig*, *kristin*, *kuzhambu*, *kya*, *ladell*, *laden*, *ladies*, *lahore*, *laid*, *lakeview*, *lam*, *lamborghini*, *larissa*, *later*, *lavin*, *lawrence*, *layoffs*, *lbl*, *lbs*, *ldschurch*, *leaking*, *leaks*, *learned*, *leet*, *leftfield*, *legacypartners*, *leland*, *lights*, *lil*, *lined*, *linscott*, *lippy*, *llegada*, *llp*, *lng*, *lobby*, *loyal*, *lsiegel*, *luddite*, *udditehome*, *luddites*, *luis*, *luisa*, *lundh*, *ma*, *macarthur*, *macias*, *madrid*, *maeve*, *mahesh*, *marg*, *margot*, *markus*, *marriage*, *marsteller*, *massage*, *mat*, *mckinsey*, *mdt*, *meant*, *megamix*, *mellon*, *men*, *mental*, *mercado*, *microshit*, *mile*, *milk*, *milpitas*, *mimeole*, *miss*, *misspell*, *misunderstand*, *mitre*, *mix*, *mmmmmmmuuuuuuuuuuuuaaa*, *moments*, *monica*, *monstro*, *mountains*, *movie*, *mpeg*, *muff*, *muoi*, *muscular*, *nabe*, *nah*, *nail*, *nalini*, *nals*, *natalie*, *nation*, *neck*, *neetu*, *neil*, *neilster*, *neimoller*, *neither*, *nephew*, *nest*, *nests*, *nextaxiom*, *nigga*, *nightline*, *nos*, *nostradamus*, *nowadays*, *nudist*, *nuevo*, *num*, *nursing*, *oh*, *oil*, *okay*, *okie*, *oktoberfest*, *ole_obj*, *olive*, *olsat*, *olympus*, *omg*, *onproject*, *oooppss*, *oops*, *opened*, *opentable*, *ordered*, *oregon*, *others*, *overdisclosure*, *owe*, *oxford*, *oz*, *pad*, *padma*, *pampered*, *papa*, *papers*, *paranoid*, *parenthood*, *paris*, *parking*, *pary*, *passionup*, *pasta*, *pathetic*, *paypal*, *pcexpo*, *pedro*, *penninsula*, *penny*, *persian*, *phenomenon*, *phish*, *phonitis*, *photographs*, *physics*, *pi*, *picnic*, *pilams*, *pink*, *pit*, *pitty*, *pkwy*, *playa*, *pm*, *poon*, *popo*, *porky*, *pots*, *pramod*, *pravalika*, *pravalikaphotos*, *prayers*, *premium*, *printable*, *prisoners*, *promise*, *props*, *providence*, *prozak*, *punjabi*, *purple*, *qaeda_telemarketing*, *ra*, *racing*, *raft*, *ramirez*, *rangamani*, *rare*, *ratboy*, *rats*, *reactive*, *rebuilding*, *refunds*, *regent*, *rematch*, *remembering*, *rememember*, *repaired*, *reservation*, *restaurant*, *restaurants*, *ret*, *returns*, *reunion*, *ric*, *ricky*, *rif*, *ring*, *ritu*, *rnalini*, *robin*, *robinm*, *robinson*, *rochelle*, *rock*, *rofl*, *ronnie*, *ros*, *rosemarie*, *rosh*, *routine*, *rs*, *rstaurants*, *rsvping*, *rtc*, *rtn*, *ru*, *rush*, *rv*, *saints*, *salamanca*, *salt*, *sandra*, *saratoga*, *savepower*, *scary*, *scene*, *scouts*, *scratch*, *screenshot*, *seafood*, *secret*, *seen*, *sera*, *serv*, *seville*, *sf*, *shah*, *shawn*, *shed*, *sheep*, *shield*, *shoreline*, *shower*, *shtm*, *siegel*, *silbo*, *sill*, *sillies*, *sin-*

*gles*, *sir*, *sista*, *skippy*, *sky*, *slim*, *sm*, *smell*, *smiles*, *sneiderman*, *soccer*, *sometime*, *sony*, *soooo*, *soul*, *southwest*, *spanish*, *spearheading*, *speedbump*, *spelled*, *spinozzi*, *sponsorship*, *sports*, *spouses*, *spray*, *sprewell*, *sprewellracing*, *springing*, *srilatha*, *sriram*, *ssrilu*, *sswanson*, *st*, *statement*, *stinkiest*, *stripes*, *strips*, *stuck*, *stupendous*, *su*, *subha*, *subsitute*, *sucking*, *sucks*, *sukhwinder*, *sulking*, *sum*, *sushi*, *suspected*, *suzanne*, *sweetie*, *sws*, *sydney*, *syed*, *tang*, *tantravahi*, *taper*, *taught*, *tavern*, *teacher*, *teachers*, *teaching*, *ted*, *telemarketing*, *telling*, *temperature*, *temple*, *temples*, *tenerife*, *tennis*, *terrible*, *terrorist*, *theonion*, *therapy*, *thicker*, *thingie*, *thks*, *thon*, *tie*, *tilex*, *tired*, *tommorrow*, *tooooo*, *touche*, *town*, *townsville*, *transport*, *travels*, *tuesday*, *tully*, *turbotax*, *udp*, *ugly*, *ulrike*, *ulrike_eder*, *ultrasound*, *umbrella*, *undergoing*, *unemployment*, *unplugged*, *unwell*, *upset*, *ur*, *usu*, *vaction*, *valle*, *vampire*, *vatha*, *veg*, *vegetarian*, *velly*, *venue*, *viahardware*, *vting*, *wa*, *walnut*, *wasp*, *wasps*, *watanabe*, *wear*, *weekends*, *weird*, *whaley*, *whaleyrh*, *whatcha*, *wheelchair*, *whenever*, *who*, *whom*, *wierd*, *wierder*, *wig*, *wild*, *willies*, *winning*, *wiring*, *wisenut*, *wishing*, *wives*, *wks*, *woman*, *won*, *wood*, *worried*, *wow*, *wretched*, *write*, *wrote*, *wtc*, *xp*, *yahoo*, *yay*, *yea*, *yeeeeeeeeeeeeeeeeeah*, *yeh*, *yell*, *yelling*, *yob*, *yoga*, *yorker*, *yoshikazu*, *young*, *younger*, *yup*, *zach*, *zafar*

Appendix B

# List of Enron Mailboxes

In this appendix, we show Enron mailboxes from different releases of Enron: EDO,[1] CALO (CMU), FREC, and Columbia (MongoDB). Note that the EDO has the same mailboxes as the ISI dataset, which is no longer online. In different releases, there are some mailboxes that have been merged and assigned to a single person. In most cases, mailboxes were merged because they belong to the same person or a person with a similar name. In some cases, mailboxes were merged for other reasons. We are particularly interested in the Columbia release (MongoDB collection) as we are using it in this thesis.

In the following (long) table, we present the Enron core mailboxes with 6 columns:

- mailbox: The name of the "mailbox"; one of the 158 released mailboxes by FREC.
- MongoDB: Mailboxes assigned to the entry representing the person in the MongoDB in the Columbia Enron release. Note that some mailboxes were merged and assigned to a single entity (person) in the MongoDB database.
- uid: A unique identifier for the MongoDB entry (person) containing this mailbox.
- EDO: The EDO id for the mailbox.
- CALO: the CALO id for the mailbox.
- FERC: the FREC id for the mailbox.

[1] `https://enrondata.readthedocs.io/en/latest/`

| Mailbox | MongoDB | uid | EDO | CALO | FERC |
|---|---|---|---|---|---|
| allen-p | allen-p | 2937 | 1 | 1 | 1 |
| arnold-j | arnold-j | 1170 | 2 | 2 | 2 |
| arora-h | arora-h | 10142 | 3 | 3 | 3 |
| badeer-r | badeer-r | 2512 | 4 | 4 | 4 |
| bailey-s | bailey-s | 2414 | 5 | 5 | 5 |
| bass-e | bass-e | 33554 | 6 | 6 | 6 |
| baughman-d | baughman-d baughman-e | 6010 | 7 | 7 | 7 |
| baughman-e | | | - | - | 8 |
| beck-s | beck-s | 46153 | 8 | 8 | 9 |
| benson-r | benson-r | 212 | 9 | 9 | 10 |
| blair-l | blair-l | 2243 | 10 | 10 | 11 |
| brawner-s | brawner-s | 2167 | 11 | 11 | 12 |
| buy-r | buy-r | 47799 | 12 | 12 | 13 |
| campbell-l | campbell-l | 1681 | 13 | 13 | 14 |
| carson-m | carson-m | 12613 | 14 | 14 | 15 |
| cash-m | cash-m | 1650 | 15 | 15 | 16 |
| causholli-m | causholli-m | 78305 | 16 | 16 | 17 |
| corman-s | corman-s | 102777 | 17 | 17 | 18 |
| crandall-s | | | 18 | - | - |
| crandell-s | crandell-s | 26954 | - | 18 | 19 |
| cuilla-m | cuilla-m | 76805 | 19 | 19 | 20 |
| dasovich-j | dasovich-j | 27095 | 20 | 20 | 21 |
| davis-d | davis-d | 5075 | 21 | 21 | 22 |
| dean-c | dean-c | 2983 | 22 | 22 | 23 |
| delainey-d | delainey-d | 755 | 23 | 23 | 24 |
| derrick-j | derrick-j | 51608 | 24 | 24 | 25 |
| dickson-s | dickson-s | 103367 | 25 | 25 | 26 |
| donoho-l | donoho-l | 70596 | 26 | 26 | 27 |
| donohoe-t | donohoe-t | 33924 | 27 | 27 | 28 |

| Mailbox | MongoDB | uid | EDO | CALO | FERC |
|---|---|---|---|---|---|
| dorland-c | dorland-c | 2132 | 28 | 28 | 29 |
| ermis-f | ermis-f | 5090 | 29 | 29 | 30 |
| farmer-d | farmer-d | 28042 | 30 | 30 | 31 |
| fischer-m | fischer-m | 12778 | 31 | 31 | 32 |
| forney-j | forney-j | 6066 | 32 | 32 | 33 |
| fossum-d | fossum-d | 3394 | 33 | 33 | 34 |
| gang-l | gang-l | 42668 | 34 | 34 | 35 |
| gay-r | gay-r | 93942 | 35 | 35 | 36 |
| geaccone-t | geaccone-t | 112633 | 36 | 36 | 37 |
| germany-c | germany-c | 3471 | 37 | 37 | 38 |
| gilbertsmith-d | gilbertsmith-d | 805 | 38 | 38 | 39 |
| giron-d | giron-d | 28206 | 39 | 39 | 40 |
| griffith-j | griffith-j | 2844 | 40 | 40 | 41 |
| grigsby-m | grigsby-m | 2587 | 41 | 41 | 42 |
| guzman-m | guzman-m | 46676 | 42 | 42 | 43 |
| haedicke-m | haedicke-m | 19235 | 43 | 43 | 44 |
| hain-m | hain-m | 4306 | 44 | 44 | 45 |
| harris-s | harris-s | 2333 | 45 | 45 | 46 |
| hayslett-r | hayslett-r | 5425 | 46 | 46 | 47 |
| heard-m | heard-m | 40104 | 47 | 47 | 48 |
| hendrickson-s | hendrickson-s | 20865 | 48 | 48 | 49 |
| hernandez-j | hernandez-j | 31382 | 49 | 49 | 50 |
| hodge-j | hodge-j | 732 | 50 | 50 | 51 |
| holst-k | holst-k | 2877 | 51 | 51 | 52 |
| horton-s | horton-s | 701 | 52 | 52 | 53 |
| hyatt-k | hyatt-k | 66073 | 53 | 53 | 54 |
| hyvl-d | hyvl-d | 1005 | 54 | 54 | 55 |
| jones-t | jones-t | 111168 | 55 | 55 | 56 |
| kaminski-v | kaminski-v smith-m | 63574 | 56 | 56 | 57 |

| Mailbox | MongoDB | uid | EDO | CALO | FERC |
|---|---|---|---|---|---|
| kean-s | kean-s | 18327 | 57 | 57 | 58 |
| keavey-p | keavey-p | 541 | 58 | 58 | 59 |
| keiser-k | keiser-k | 63556 | 59 | 59 | 60 |
| king-j | king-j | 20805 | 60 | 60 | 61 |
| kitchen-l | kitchen-l | 14758 | 61 | 61 | 62 |
| kuykendall-t | kuykendall-t | 1581 | 62 | 62 | 63 |
| lavorado-j | lavorato-j lavorado-j | 2273 | - | - | 64 |
| lavorato-j | | | 63 | 63 | 65 |
| lay-k | lay-k | 40264 | 64 | 64 | 66 |
| lenhart-m | lenhart-m | 40160 | 65 | 65 | 67 |
| lewis-a | lewis-a mckay-b | 6428 | 66 | 66 | 68 |
| linder-e | linder-e | 37545 | 67 | 67 | 69 |
| lokay-m | lokay-m | 81014 | 68 | 68 | 70 |
| lokey-t | lokey-t | 42499 | 69 | 69 | 71 |
| love-p | love-p | 91168 | 70 | 70 | 72 |
| lucci-p | lucci-p luchi-p | 73955 | 71 | 71 | 73 |
| luchi-p | | | - | - | 74 |
| maggi-m | maggi-m | 3774 | 72 | 72 | 75 |
| mann-k | mann-k | 239 | 73 | 73 | 76 |
| martin-t | martin-t | 71768 | 74 | 74 | 77 |
| may-l | may-l | 5970 | 75 | 75 | 78 |
| mccarty-d | mccarty-d | 304 | 76 | 76 | 79 |
| mcconnell-m | mcconnell-m | 18533 | 77 | 77 | 80 |
| mckay-b | | | 78 | 78 | 81 |
| mckay-j | mckay-j | 40294 | 79 | 79 | 82 |
| mclaughlin-e | mclaughlin-e quigley-d | 11370 | 80 | 80 | 83 |
| merriss-s | merriss-s | 106450 | 81 | 81 | 84 |
| meyers-a | meyers-a | 6971 | 82 | 82 | 85 |
| mims-p | mims-p mims-thurston-p | 88994 | - | - | 86 |

| Mailbox | MongoDB | uid | EDO | CALO | FERC |
|---|---|---|---|---|---|
| mims-thurston-p | | | 83 | 83 | 87 |
| motley-m | motley-m | 29545 | 84 | 84 | 88 |
| neal-s | neal-s | 85818 | 85 | 85 | 89 |
| nemec-g | nemec-g | 1191 | 86 | 86 | 90 |
| panus-s | panus-s phanis-s | 1235 | 87 | 87 | 91 |
| parks-j | parks-j | 58825 | 88 | 88 | 92 |
| pereira-s | pereira-s | 107148 | 89 | 89 | 93 |
| perlingiere-d | perlingiere-d | 30637 | 90 | 90 | 94 |
| phanis-s | | | - | 91 | 95 |
| pimenov-v | pimenov-v | 4769 | 91 | 92 | 96 |
| platter-p | platter-p | 91174 | 92 | 93 | 97 |
| presto-k | presto-k | 19920 | 93 | 94 | 98 |
| quenet-j | quenet-j | 29542 | 94 | 95 | 99 |
| quigley-d | | | 95 | 96 | 100 |
| rapp-b | rapp-b | 1934 | 96 | 97 | 101 |
| reitmeyer-j | reitmeyer-j | 18492 | 97 | 98 | 102 |
| richey-c | richey-c | 1559 | 98 | 99 | 103 |
| ring-a | ring-a | 3679 | 99 | 100 | 104 |
| ring-r | ring-r | 18447 | 100 | 101 | 105 |
| rodrigue-r | | | 101 | - | - |
| rodrique-r | rodrique-r | 40281 | - | 102 | 106 |
| rogers-b | rogers-b | 13147 | 102 | 103 | 107 |
| ruscitti-k | ruscitti-k | 66165 | 103 | 104 | 108 |
| sager-e | sager-e | 7319 | 104 | 105 | 109 |
| saibi-e | saibi-e | 29551 | 105 | 106 | 110 |
| salisbury-h | salisbury-h | 2269 | 106 | 107 | 111 |
| sanchez-m | sanchez-m | 2834 | 107 | 108 | 112 |
| sanders-r | sanders-r | 40537 | 108 | 109 | 113 |
| scholtes-d | scholtes-d | 32017 | 109 | 110 | 114 |

| Mailbox | MongoDB | uid | EDO | CALO | FERC |
|---|---|---|---|---|---|
| schoolcraft-d | schoolcraft-d | 3231 | 110 | 111 | 115 |
| schwieger-j | schwieger-j | 5622 | 111 | 112 | 116 |
| scott-s | scott-s | 16810 | 112 | 113 | 117 |
| semperger-c | semperger-c | 18911 | 113 | 114 | 118 |
| shackleton-s | shackleton-s | 64528 | 114 | 115 | 119 |
| shankman-j | shankman-j | 28265 | 115 | 116 | 120 |
| shapiro-r | shapiro-r | 2482 | 116 | 117 | 121 |
| shively-h | shively-h storey-g | 43716 | 117 | 118 | 122 |
| skilling-j | skilling-j williams-j | 7016 | 118 | 119 | 123 |
| slinger-r | slinger-r | 100052 | 119 | 120 | 124 |
| smith-m | | | 120 | 121 | 125 |
| solberg-g | solberg-g | 43566 | 121 | 122 | 126 |
| south-s | south-s | 107737 | 122 | 123 | 127 |
| staab-t | staab-t | 1521 | 123 | 124 | 128 |
| stclair-c | stclair-c | 3773 | 124 | 125 | 129 |
| steffes-j | steffes-j | 6087 | 125 | 126 | 130 |
| stepenovitch-j | stepenovitch-j | 11224 | 126 | 127 | 131 |
| stokley-c | stokley-c | 3680 | 127 | 128 | 132 |
| storey-g | | | 128 | 129 | 133 |
| sturm-f | sturm-f | 827 | 129 | 130 | 134 |
| swerzbin-m | swerzbin-m | 49232 | 130 | 131 | 135 |
| symes-k | symes-k | 11426 | 131 | 132 | 136 |
| taylor-m | taylor-m | 2378 | 132 | 133 | 137 |
| tholt-j | tholt-j | 52172 | 133 | 134 | 138 |
| thomas-p | thomas-p | 309 | 134 | 135 | 139 |
| townsend-j | townsend-j | 2859 | 135 | 136 | 140 |
| tycholiz-b | tycholiz-b | 899 | 136 | 137 | 141 |
| ward-k | ward-k | 18442 | 137 | 138 | 142 |
| watson-k | watson-k | 66827 | 138 | 139 | 143 |

| Mailbox | MongoDB | uid | EDO | CALO | FERC |
|---|---|---|---|---|---|
| weldon-c | weldon-v weldon-c wheldon-c | 11307 | 139 | 140 | 144 |
| weldon-v | | | - | - | 145 |
| whalley-g | whalley-g whalley-l | 27104 | 140 | 141 | 146 |
| whalley-l | | | - | 142 | 147 |
| wheldon-c | | | - | - | 148 |
| white-s | white-s | 2310 | 141 | 143 | 149 |
| whitt-m | whitt-m | 1141 | 142 | 144 | 150 |
| williams-b | williams-w3 williams-b | 14326 | - | - | 151 |
| williams-j | | | 143 | 145 | 152 |
| williams-w3 | | | 144 | 146 | 153 |
| wolfe-j | wolfe-j | 617 | 145 | 147 | 154 |
| ybarbo-p | ybarbo-p | 270 | 146 | 148 | 155 |
| zipper-a | zipper-a | 5063 | 147 | 149 | 156 |
| zufferli-j | zufferli-j zufferlie-j | 36183 | 148 | 150 | 157 |
| zufferlie-j | | | - | - | 158 |