ABSTRACT

Title of dissertation: IDENTITY RESOLUTION IN EMAIL COLLECTIONS

Tamer Mohamed Elsayed, Doctor of Philosophy, 2009

Dissertation directed by: Associate Professor Douglas W. Oard
Department of Computer Science
and College of Information Studies

Access to historically significant email collections poses challenges that arise less often in personal collections. Most notably, people exploring a large collection of emails, in which they were not sending or receiving, may not be very familiar with the discussions that exist in this collection. They would not only need to focus on understanding the topical content of those discussions, but would also find it useful to understand who the people sending, receiving, or mentioned in these discussions were.

In this dissertation, the problem of resolving personal identity in the context of large email collections is tackled. In such collections, a common name (e.g., John) might easily refer to any one of several hundred people; when one of these people was mentioned in an email, the question then arises: " who is that John?"

To "resolve identity" of people in an email collection, two problems need to be solved: (1) modeling the *identity* of the participants in that collection, and (2) *resolving* name-mentions (that appeared in the body of the messages) to these identities. To tackle the first problem, a simple computational model of identity, that is built on extracting unambiguous references (e.g., full names from headers, or nicknames from free-text sig-

natures) to people from the whole collection, is presented. To tackle the second problem, a generative probabilistic approach that leverages the model of identity to resolve mentions is presented. The approach is motivated by intuitions about the way people might refer to others in an email; it expands the context surrounding a mention in four directions: the message where the mention was observed, the thread that includes that message, topically-related messages, and messages sent or received by the original communicating parties. It relies on less ambiguous references (e.g., email addresses or full names) that are observed in some context of a given mention to rank potential referents of that mention.

In order to jointly resolve all mentions in the collection, a parallel implementation is presented using the MapReduce distributed-programming framework. The implementation decomposes the structure of the resolution process into subcomponents that fit the MapReduce task model well. At the heart of that implementation, a parallel algorithm for efficient computation of pairwise document similarity in large collections is proposed as a general solution that can be used for scalable context expansion of all mentions and other applications as well.

The resolution approach compares favorably with previously-reported techniques on small test collections (sets of mention-queries that were manually resolved beforehand) that were used to evaluate the task in the literature. However, the mention-queries in those collections, besides being relatively few in number, are limited in that all refer to people for whom a substantial amount of evidence would be expected to be available in the collection thus omitting the "long tail" of the identity distribution for which less evidence is available. This motivated the development of a new test collection that now is the largest and best-balanced test collection available for the task. To build this collection, a

user study was conducted that also provided some insight into the difficulty of the task and how time-consuming it is when humans perform it, and the reliability of their task performance. The study revealed that at least 80% of the 584 annotated mentions were resolvable to people who had sent or received email within the same collection.

The new test collection was used to experimentally evaluate the resolution system. The results highlight the importance of the social context (that includes messages sent or received by the original communicating parties) when resolving mentions in email. Moreover, the results show that combining evidence from multiple types of contexts yields better resolution than what can be achieved using any individual context. The one-best selection is correct 74% of the time when tested on the full set of the mention-queries, and 51% of the time when tested on the mention-queries labeled as "hard" by the annotators. Experiments run with iterative reformulation of the resolution algorithm resulted in modest gains only for the second iteration in the social context expansion.

IDENTITY RESOLUTION IN EMAIL COLLECTIONS

by

Tamer Mohamed Elsayed

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2009

Advisory Committee:
Associate Professor Douglas W. Oard, Chair
Professor Ben Shneiderman
Associate Professor Philip Resnik
Assistant Professor Jimmy Lin
Associate Professor David Kirsch

Dedication

To my mother and father
who dreamed of that moment, but could not live to see it.

♢

To my wife Shereen, daughter Nour, son Mohamed, and brother Khaled
who have constantly supported me along this long path.

♢

To the people of Gaza, Palestine
who were, are, and (by the will of Allah) will stay strong and patient.

Acknowledgments

Working on my dissertation has indeed taken a long path and there must be a list of people who played major roles in achieving such success.

First and foremost, I would like to express my profound gratitude to my advisor Doug Oard. Being new to the field of information retrieval when I started my research work with him, he has been very patient with me in every step of the path, from searching for a dissertation topic till writing this dissertation. I cannot ever forget his sincere advice, insightful feedback, encouraging comments, and friendly discussions throughout the whole process. It was my pleasure working with him.

I would like to thank my dissertation committee members Ben Shneiderman, Philip Resnik, Jimmy Lin, and David Kirsch for their valuable comments and suggestions. Specifically, I want to thank Jimmy Lin for giving me the opportunity for joining his "Cloud Computing" class, which inspired my MapReduce line of work.

I especially would like to thank Gary Kuhn who steered the first research project I worked on jointly with Doug. He was such a cheerful and dedicated person who helped me considerably in my first steps.

It is a pleasure to acknowledge individuals who have contributed to this dissertation. I want to acknowledge Galileo Namata for co-authoring a technical report and a full paper with me on the main ideas of this dissertation. He also helped me in writing an Institutional Review Board (IRB) application for building the test collection I have used to evaluate my work. I also want to thank Lise Getoor for her useful comments on an early version of my work. Special thanks go to Yejun Wu for his collaboration on an evaluation

experiment and all CLIP Lab faculty and students for their moral support. I am especially indebted to the whole family of Masjid Al-Tauba, and more specifically my dear friends Moustafa Youssef and Mohamed Hussein, for their consistent encouragement and advice.

This work would not have been possible without the support of the Joint Institute of Knowledge Discovery (JIKD) at University of Maryland and the Human Language Technology-Center of Excellence (HLT-COE) at Johns Hopkins University. Colleagues at both institutes have provided very useful feedback on different pieces of work in this dissertation. I am also so thankful to IBM and Google for giving me access to their Hadoop cluster, without which I could not feasibly run my lengthy experiments.

And before all of course, I am so much grateful to Allah, the Almighty, for the countless bounties he showered me with. Without his care, guidance, and blessings in my whole life, I could not achieve what I am up to; in no way I can thank him enough.

Table of Contents

List of Figures

Chapter 1

Introduction

Informal conversational media (such as email, instant messaging, discussion lists, and voice messaging) are now increasingly popular and an important part of daily life for many people. Software systems that enable users to communicate with text, voice, and video are widely available and significantly affecting our social experience. Because of the declining cost of long-term storage, such services have the potential to be an invaluable future resource for understanding the past. For example, the National Archives of the United States has received 32 million emails from the White House adminstration of former President Bill Clinton and hundreds of millions of emails from George W. Bush's. Making sense of collections at this scale will require new types of tools.

Unlike the traditional collections of news articles that steered the initial research in the field of information retrieval, informal textual communication media that are conversational put more emphasis on several remarkable characteristics:

- The data is naturally temporal in the sense that events, actions, and topics evolve over time. This can make it hard for searchers who are neither involved in the discussions nor familiar with the specific topics to understand what is actually meant without reconstruction of the surrounding context.

- New (non-dictionary) words may have been introduced. Moreover, some existing dictionary words may have been used non-traditional meanings that are understand-

able only by a small group of people communicating closely.

- The metadata that are associated with each unit of conversation can help people recognize who is talking to whom and when, hence understand and make sense of the conversations. While the users of more formal documents can rely on the source characteristics (e.g., journal reputation), the central role of individuals in the construction of informal conversations results in an explosive proliferation of sources (e.g., email participants) that searchers new to these conversations could have great difficulty comprehending.

- The conversational nature makes the identity of individuals a key factor for tasks such as exploratory search and social network analysis.

- Informality makes handling such data more challenging for automated natural language processing systems. It raises some non-trivial issues such as spelling mistakes, broken sentences, and interspersed text segments.

All of these characteristics make the problem of handling informal conversational text *different* from traditional documents, and thus open a new set of research problems that were not previously evident; the one addressed in this dissertation is identity resolution.

People exploring a large collection of informal conversational text in which they were not involved (by sending or receiving,) may not be very familiar with the discussions that exist in this data. They would not only need to focus on understanding the topical content of those discussions, but would also find it useful to understand who the people talking, listening, or mentioned in these discussions were. In fact, resolving (i.e.,

disambiguating) the identity of people involved in these conversations will be beneficial to better understand what could be found in this kind of data.

This problem of identity resolution is apparent in several exploratory tasks that deal with large informal collections (e.g., retained emails, printed letters, and instant messages) such as:

- a lawyer searching in retained emails, printed documents, and memos for evidence that relates to intentions, or statements of or about specific individuals in a legal case.

- an historian putting together a timeline of decisions made that led to the rise of an organization by analyzing the actions of the key individuals affecting those decisions.

- a police investigator tracking people involved in a specific event through a collection of their digitally-recorded text conversations.

- an archivist indexing the data (of the White House for example) by who was involved in each conversation.

Email is perhaps one of the most popular and mature informal communication media.[1] Research on email access has traditionally focused on tools for managing personal collections, in part because larger and more diverse collections were not available for research use. That is starting to change, most notably with the introduction of the Enron

[1] An October 2006 report by technology market research firm "The Radicati Group" (www.radicati.com) estimated that there were 1.1 billion email users and 1.4 billion active email accounts worldwide.

collection [48] to the research community. The availability of this collection motivated the work in this dissertation to focus on email as the domain in tackling the problem of identity resolution for this dissertation. The problem is concerned with modeling and resolving the identity of the involved participants (i.e., who sent email, received email, or were mentioned in email).

1.1 Identity Resolution in Email

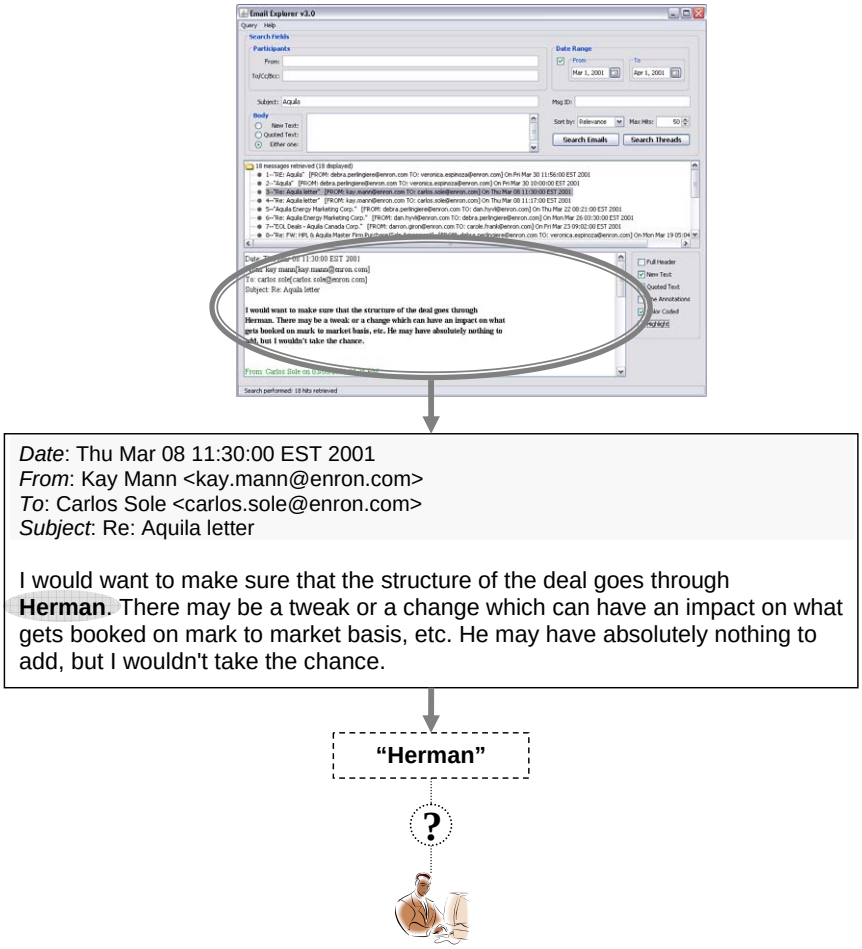


Figure 1.1: An example of an actual email message from the Enron collection that illustrates the problem of identity resolution.

As the largest (to the author's knowledge) email collection available for research purposes, the Enron collection was elected to be used in the experiments conducted in this dissertation. The collection was first released by the Federal Energy Regulatory Commission (FERC) during the investigation of the former energy trading company. The email dataset was later purchased by Leslie Kaelbling at MIT, and then a number of folks at SRI tried to address some of the email-formatting issues. The version of the collection used in this dissertation is the one finally hosted by CMU.[2] It has a large number of communicating parties and includes 517,431 messages in 150 top-level directories; each of which contains the retained emails of a former Enron employee on the date that the collection was obtained, but without attachments.

A typical email message from the Enron collection is shown in figure 1.1. For the user to understand that email, it could be useful to know who "Herman" is. There might be tens (or hundreds) of people involved in the collection whose first name is "Herman", but the question arises: "which of them is the one intended here?" A simple automatic resolution output would either be the email address (as a unique identifier) of the most-probable referent, or a ranked list of the most-probable ones by their identifiers (assuming, of course, that the referent actually has an email address in the collection.)

Solutions to the above problem can be leveraged in different applications, either for an automatic downstream process or an end-user:

- It can be used in exploratory tools for email collections by users such as historians, archivists, and lawyers, to produce online resolutions of ambiguous mentions. The resolutions can be linked to other emails in which the same people were involved

[2] http://www.cs.cmu.edu/∼enron

as well. An example of the type of tools that can benefit from this work is NetLens [45]; an iterative analysis tool that enables the exploration of content-actor network data.

- It can be used to enrich social networks inferred from email communications. Resolving mentions in the body of emails has the effect of adding an indirect relation, labeled "mentioned by", to the social network in addition to the traditional "sent to" and "received from" relations.

- It can be used to replace the ambiguous mentions by the true referent's full name or email address, in order to improve the performance of automatic email retrieval systems. This could be done at indexing time.

- It can be used as a component in generating biographies of participants using summaries of emails in which they were mentioned in addition to the emails they sent or received.

- It can be used in an expert finding task by an automated system that can jointly model both content and people to suggest experts for specific topics.

1.2 Structure of the Problem

The task of identity resolution in email is defined as follows. Given a collection of raw emails, the goal is to build a system that seeks to resolve (i.e., disambiguate) the identity of the persons who were involved (i.e., participated) in that collection, whenever references to them are observed. In the resolution process, the system infers which per-

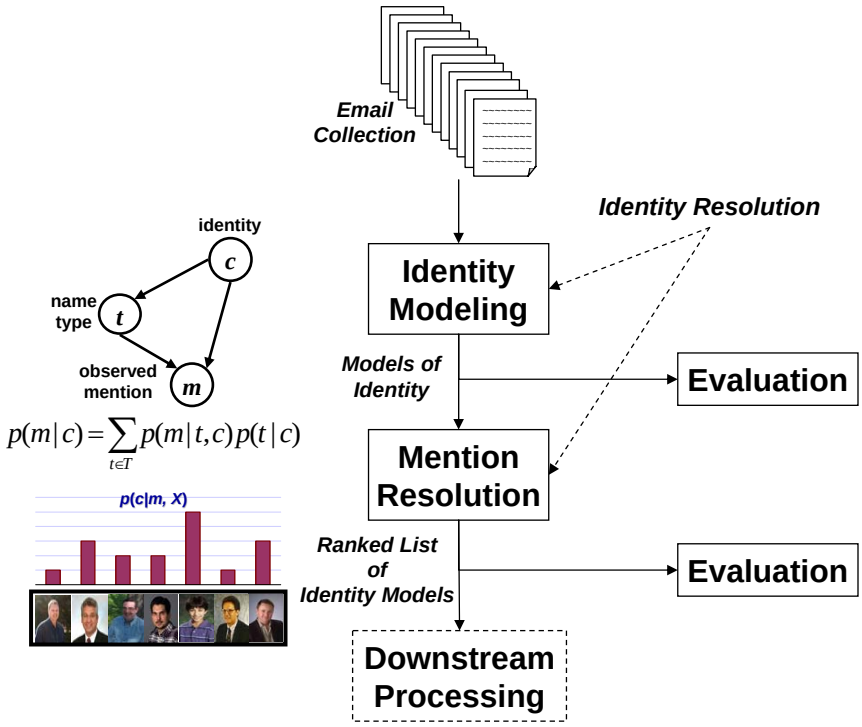


Figure 1.2: Basic components of identity resolution system.

sonal identity sent, received or was mentioned in an email. Since the senders and receivers of an email are easily identified by their email addresses, this dissertation focuses on resolving ambiguous names that are mentioned in the body of an email. Collectively, the general goal is to determine for each participant, all the emails that refer to that participant, either in the email headers or in the body. No side information about the participants (e.g., their names, job titles, relationships) is assumed to be available, in order to focus the research on the general problem.

The problem can conceptually be considered a special case of the more general problem of "Word Sense Disambiguation" (WSD) which is well-studied in the field of natural language processing [80]. WSD is concerned with labeling an ambiguous word (e.g., bank) with the appropriate sense (either river or financial institution) according to

the context in which the word appears. In our problem, the ambiguous word is a name-mention (e.g., Alan) and the potential senses are people who can be called by that name. The domain is also specific; email. The biggest difference that makes our special case much more challenging is that the cardinality of the potential candidate set of referents for a name-mention in email data (which sometimes reaches hundreds in Enron collection as shown in Chapter 7) is much larger than the case in typical WSD problems.

The task can also be formulated as a classification task, where the references are the objects being classified and the personal identities are the actual classes. In fact, there is a closely-related problem that is called "cross-document co-reference resolution" in which all mentions that refer to the same person are clustered together, without the need of a personal identifier that plays the role of a class label.

There is a basic set of sub-problems that are inherent in the problem of identity resolution in email. Three sub-problems, illustrated in figure 1.2, are to be addressed:

1. **How to model identity**: In order to resolve references to people, a computational model of each participant in the email collection is needed. The specific structure of the identity model, that supports the use of the whole system, should be provided and how the model is populated and then leveraged in the resolution process should be determined. The accuracy of the resulting models should be evaluated before being used in the resolution process.

2. **How to resolve name-mentions**: Ambiguous name-mentions should be resolved to the true referent, or computationally to the referent's unique identifier (e.g., email address.) A mention resolution system that can accurately resolve name-mentions

in the collection should be designed. The resolution output of one reference can be either one identity (selected from a set of potential candidates) or a ranked list of candidates, each assigned a score (e.g., an estimated probability of being the true referent.) This depends on how the downstream process will use that output. The process of both identity modeling and mention resolution is called *identity resolution*.

3. **How to evaluate the resolution system in a repeatable yet affordable way**: Automated resolution of mentions should be evaluated to measure the effectiveness of the system. For an affordable evaluation of variants of the system, an appropriate test collection should be used. A test collection generally consists of three components: a set of documents, a set of queries that represent actual user needs, and true answers to that set of queries [86]. The test collection in our case consists of an email collection, a set of ambiguous name-mentions, and manual resolutions of these mentions.

The three sub-problems impose the structure of the proposed solution to the main problem. First an identity representation that captures the main personal attributes such as email addresses, full names, and nicknames that can identify participants, is designed. The representation forms the basis of a computational model that is used in the resolution process. An automated resolution algorithm is then designed to resolve mentions in the whole collection, using a model that is motivated by the way an email author might mention a person in an email. The resolution system should be accurate (for a reliable downstream reasoning) and probably scalable (for downstream applications that require

resolving all mentions in the collection.) Finally, a representative test collection and evaluation measures are specifically identified for this task to evaluate the performance of the resolution system.

1.3 Contributions

This dissertation makes the following contributions:

- A computational model of identity that can be used for resolving mentions in email is designed and implemented. The model associates names (including nicknames) and email addresses for participants in the email collection.

- A generative probabilistic model for mention resolution is proposed. A complete resolution system based on that model is implemented and evaluated on every test collection suitable for that task that the author knows of. The system achieves accuracy that is at least as good as, and often better than, previously-reported results.

- A parallel algorithm for resolving all mentions in an email collection using the MapReduce framework [30] is designed and implemented. This scalable solution resolves approximately 1.3 million mentions in about 3 hours.

- A parallel algorithm for efficient computation of pairwise document similarity in large collections using MapReduce is designed and implemented. This algorithm is a general solution that has been shown to be useful in the implemented resolution system, and for other applications (e.g., query-by-example retrieval [53].)

- A new large and balanced test collection for mention resolution in email, that is now the best available test collection for that task, is developed.

1.4 Dissertation Outline

The rest of this dissertation is organized as follows. Chapter 2 reviews related research. Chapter 3 introduces the computational model of identity designed for the resolution task. The mention resolution algorithm is discussed in Chapter 4 and its MapReduce implementation is detailed in Chapter 5. The new test collection developed specifically for the task is presented in Chapter 6, followed by an experimental evaluation of the resolution system in Chapter 7. Finally, Chapter 8 concludes the dissertation with some remarks about future work.

Chapter 2

Related Work

This chapter discusses research related to the work of this dissertation from several perspectives. Research on the general problem of entity resolution is first introduced, followed by an overview of the related work on the dissertation main problem, identity resolution in email, in its two major sub-problems: mention resolution and identity modeling. There is also a large body of work on other problems in the email domain; this chapter gives some pointers to it. The chapter concludes by a brief introduction to MapReduce, the tool used in implementing the proposed resolution approach, and other related alternative frameworks that were recently presented.

2.1 Entity Resolution

The problem of identity resolution in email is a special case of a more general problem referred to as "entity resolution." Entity resolution is generically defined as the process of determining a mapping from references observed in data to real-world entities. Examples of references are names, words, and phrases. Data can be structured (e.g., a relational database) or unstructured (e.g., news articles). Real-world entities can, for example, be people, organizations, or locations. The general problem is also known as "entity disambiguation," "co-reference resolution," "anaphora resolution," or "deduplication" in different domains. The specific problem tackled in this dissertation is an instance

of entity resolution that is concerned with resolving references to *people* in *unstructured* data, specifically *emails*. In this particular domain, *mentions* refer to references, and *identities* refer to people.

2.1.1 Resolution in Structured Data (Relational Databases)

Most of the research on entity resolution has been focused on structured data. Here a few recent approaches are cited.

A generic perspective to approach the entity resolution problem in relational data was taken by Benjelloun et al. [13]. Their model was based on two black-box functions, "match" and "merge," provided as input to the resolution engine. Given such black-boxes, an efficient resolution strategy was developed to minimize the number of invocations to these potentially expensive black-boxes. A set of properties that the black-boxes should have were identified that would lead to a well-defined single "answer" to the problem, as well as to efficient algorithms.

An underlying assumption in that model was that local pairwise matching decisions could be made. While this simplifies the solution from a generic perspective, other studies have shown that leveraging global relational information between groups of references can yield better results.

In one such study, Malin [56] adopted a social network-based disambiguation approach that leverages community similarity, rather that lexical similarity, to resolve references in relational databases.

Another study was conducted by Bhattacharya and Getoor [15]. They formalized

the problem as a clustering problem and proposed similarity measures for clustering references, taking into account the different relations that are observed among them in addition to their attribute (i.e., local) similarities. They also developed a probabilistic generative model [14] that extends Latent Dirichlet Allocation. They used the notion of collaborating groups of entities and developed an unsupervised sampling algorithm for doing inference in that model. The algorithm jointly determines the most likely number of entities and the mapping of references to entities. Experiments on two different citations datasets showed improvements over traditional attribute-based models when relations were leveraged. Although, these approaches have been applied to relational data, they can also be applied to resolve references in email headers. However, it is not clear how conversational, topical, or time aspects could be leveraged in their model.

Reuther [79] offered some insights into the importance of leveraging temporal and topical aspects to match personal names in co-authorship networks, but those aspects were not reflected in his approach.

2.1.2 Resolution in Unstructured Data (Text Documents)

Several techniques have been proposed to resolve references in text collections, mostly on collections of web pages and news articles.

An intuitive way to resolve mentions is to extract unique personal information that co-occurs with each mention and then cluster that information into groups that refer to the same underlying person. This approach was adopted by Mann et al. [57] to distinguish personal names with multiple real referents in Web pages, based on little or no

supervision. The approach utilizes a clustering technique over a rich feature space of biographic facts (e.g., date of birth, location of birth), which were automatically extracted via a language-independent bootstrapping process.

The traditional structure of news articles that usually refers to unambiguous (e.g., full names) before ambiguous references are used, has motivated a two-step approach for resolving mentions in news text [18]. First, within-document co-reference resolution is performed to aggregate information about each entity mentioned in each document. Then this information was used along with other features found in documents to determine which documents mention the same entity. Within-document resolution may, of course, be less useful in the context of conversational text because of the phenomenon of early negotiation, in which a private vocabulary is "negotiated" early in the conversational interaction.

Probabilistic models have been successfully applied to tackle the entity resolution problem. McCallum and Wellner [61] introduced a probabilistic model for proper noun co-reference resolution across text documents. The model defines a conditional probability distribution over partitions of mentions, given all observed mentions.

Lie et al. [52] presented two learning approaches to the entity resolution problem in the context of news articles. The first was a supervised discriminative approach where a pairwise local classifier was trained, followed by a global clustering algorithm that used the classifier to produce a similarity measure. Their second approach developed an unsupervised generative model that aimed to better exploit the natural generation process of documents and the process of how names are scattered into them, taking into account dependencies among entities. While these performed well, they have only been applied

on small datasets, and they have not been demonstrated to be scalable.

The most similar approach to the one adopted in this dissertation (in the sense that both try to leverage the context of a reference) was adopted by Pedersen et al. [73]. They proposed an unsupervised approach that resolves name ambiguity by clustering the instances (passages of text) of a given name into groups, each of which is associated with a distinct underlying entity. A vector for each instance is built using features of words that co-occur in the set of all instances with the words that appear in a window surrounding the given names in that specific instance (i.e., second order co-occurrence). The general intuition behind the second order representation is that it captures indirect relationships between words. The method was applied to a newswire corpus with pairs of manually conflated names (pseudo-names). This approach was first proposed by Schutze in the context of word sense discrimination [82]. Results of that approach showed good performance for a sample of natural and artificial ambiguous words.

Bagga and Baldwin [7] suggested a Vector Space Model solution of that problem and described a scoring algorithm for evaluating the coreference chains (i.e., clusters). A comparison of different statistical methods in the task of cross-document coreference resolution was conducted by Gooi and Allan [41]. The study showed that the agglomerative vector space clustering algorithm consistently yields acceptable performance.

To link references of same entity in different languages, Aktolga et al. [3] used entity language models to capture the contextual language around a given alias, which aids in finding new aliases to the same entity in another language. This is very similar to the approach adopted in [81] to detect coreferences in Arabic documents using English training data.

More recently, Wick et al. [89] proposed a discriminatively-trained model that reasons over the entities not the references, and thus jointly performed coreference resolution and canonicalization (i.e., extracting attributes of the entities). Their approach achieved an error reduction of up to 62% when compared to a method that reasons about references only.

2.2 Mention Resolution in Email

In contrast to the other types of text, the nature of email as a conversational medium has suggested different approaches to the problem. Email messages are often short, so within-document clues for mention resolution are typically rare. Moreover, individual messages are often not self-contained, so complementary context might be found outside the collection. Furthermore, email is personal, so terms might be specific to the communicating parties. By contrast, news articles and Web pages are often one-shot, self contained, and public, so a mention can most probably be resolved from earlier mentions in the same document. Therefore, it is tempting to exploit a broad range of collection resources in searching for evidence that can help with resolving mentions found in email.

All of the research cited above has focused on the task of resolving *all* references in a collection collectively. This task is referred to as "joint resolution." Since performing this task is costly and time-consuming in practice, especially in growing collections, there is a related task that is concerned only with resolution of a specific reference. This task is referred to as "query-based" resolution. One way of query-based resolution is to reduce the problem to joint resolution, but for only a subset of references that are related to the

original reference. An example that adopted that reduction approach is the adaptive strategy introduced in [16] for extracting the set of most relevant references for collectively resolving a query.

Entity (or identity) resolution in email has not received much research attention until the recent introduction of fairly large publicly available email collections, most notably the Enron collection. Two approaches have been tried by Diehl et al. [31] and by Minkov et al. [63].

Diehl et al. used temporal models of email traffic to resolve name references in a subset of Enron email collection. The candidates were restricted to individuals who had communicated with the sender of the message in which the reference to be resolved was found. The candidates were scored based on the temporal characteristics of their interaction with the participants of the message in which the mention to be resolved was found. Two averaging filters (autoregressive and moving average) were applied to the past and future interaction patterns. Experiments with varying time periods showed that long-term patterns exhibited the greatest utility for mention resolution. In contrast to this dissertation's work with the entire collection, they focused only on Enron-domain email addresses. Their test collection and queries have been used in the experiments in this dissertation as one reference test collection.

Minkov et al. adopted a different approach using a graphical framework to represent all of the specific objects in an email collection. Their defined objects were: email messages, persons, email addresses, terms, and dates. Similarity between these objects were estimated using a multi-step lazy graph walk (where there is a fixed probability of halting the walk at each step.) Each object was represented as a node in that graph and

relations between pairs of objects were represented by the edges. Related objects that are not directly linked to a query could be reached via a multi-step graph walk. Among the applications of this framework is the task of resolving references in the emails.

Minkov et al. automatically generated two small test collections by simulating a non-trivial matching of references to individuals whose names appeared in the "Cc" header field. While they showed that a resolution based on that framework was effective on two small subsets of the Enron collection, no results were reported for larger collections, leaving the scalability of the technique uncertain. Their test collections have been used in the experiments conduced in this dissertation for comparison.

It is worth noting that joint resolution in email has not yet been researched. This might be because (1) the task is complex, and (2) evaluation of that task requires a significant human annotation effort (and thus understanding) of a large collection. This dissertation tackles both problems. An efficient and scalable solution for joint resolution is implemented, and a new test collection is developed for the task. The new test collection is much larger than existing collections and spans a broader range of mentions than has been investigated in previous work.

2.3 Identity Modeling in Email

Research on modeling identity in email collections can draw on a substantial amount of prior work.

Carvalho and Cohen [21] applied machine learning methods to effectively detect signature blocks and quoted text in the Enron collection. The approach adopted in this

dissertation uses a simpler unsupervised technique and extends it by detecting salutations and nicknames as well. Studies have also used the Web as an external source.

Culotta et al. [27] used the Web to extract contact information for people whose names and email addresses were extracted from email headers.

Holzer et al. [43] proposed a graph-proximity-based technique to determine which email addresses correspond to the same entity. This is exactly the problem of extracting "address-address associations" discussed in Section 3.3. They approached the problem by analyzing the relational network of addresses extracted from Web pages. The proposed solution in this dissertation is restricted to the email collection.

Exploiting name repetition in the email collection, Minkov et al. [64] proposed a recall enhancing technique for name recognition in email collections. They used name dictionaries to help train their models.

There are a wide range of research problems that take some form of an identity model as a starting point, the most deeply explored of which is social network analysis. McArthur and Bruza [59] found explicit and implicit connections between people by mining semantic associations inferred from their email communications. McCallum et al. [60] proposed a Bayesian network that learns a topic distribution for communication between two entities based on the content of the messages sent between them. Finally, Keila and Skillicorn [46] applied a model based on patterns of word usage to detect deceptive emails in Enron collection.

Finding experts in social networks has also been studied using a variety of techniques [83, 96]. The Text Retrieval Conference (TREC) also introduced the expert finding task using W3C mailing list emails [87] as part of the Enterprise Search track [26].

Dome et al. [32] used graph-based ranking measures to rank people on their expertise of a given topic. The nodes of the graph represent people, and edges represent on-topic emails between them.

Two algorithms for determining expertise from email have been proposed by Campell et al. [20]: the first considers the content of the messages communicated between candidates and the other considers the communication graph as well. Results show that considering both the graph and the content performs better than the content-only algorithm. This conceptually matches the "combination of evidence" approach adopted in this dissertation.

Balog and de Rijke [9] adopted an approach that locates messages on a topic, and then finds the associated experts using a standard language modeling for IR approach; their approach combined evidence from both metadata and content of the email messages. Balog at al. [8] later generalized the approach into two methods: The first directly models an experts knowledge based on the documents that they are associated with, whilst the second locates documents on topic, and then finds the associated expert. Experiments show that the second strategy consistently outperforms the first. Finally, Balog and de Rijke [10] complemented both document and proximity-based approaches by importing global evidence of expertise (i.e., evidence obtained using information that is not available in the immediate proximity of a candidate expert's name occurrence or even on the same page on which the name occurs.) Examples include candidate priors, query models, as well as other documents a candidate expert is associated with. Results show that the refined models significantly outperform existing state-of-the-art approaches.

2.4 Research on Email-Related Problems

There has been extensive work that tackles other different problems on email genre, such as sentiment analysis [91], name matching [40], event extraction[22], discussion search [88], question answering [84], author identification [6, 28, 29], expert search [32, 20, 9, 8, 10], email summarization [94, 33], and email clustering [47, 12, 58].

This section briefly touches some work done on two of these problems, where similar ideas to the work proposed in this dissertation have been adopted.

2.4.1 Email Classification

The problem of email classification is somewhat different from the standard problem of topic classification due to the nature of email; the task is subjective and the user may apply different criteria that are difficult to detect [12].

Klimt et al. [47] used both structured email fields such as "From" and "To" and unstructured fields such as "Subject" and "Body" as classification features. They also studied how to use the threads for the task of email classification, but time information was left out of their experiments. Bekkerman et al. [12] used a new evaluation method based on the email timeline and found that the classification accuracy (using different classification methods) was relatively low due to difficulty of the task. Martin et al. [58] used a collection of behavioral features of a users email traffic to rapidly detect abnormal email activity. They specifically analyzed the outgoing email behavior for virus detection.

2.4.2 Email Summarization

Emails are often short and hard to understand in isolation; therefore approaches to the task of email summarization tend to use contextual information that can be observed in threads and other related emails.

Zajic et al. [94] suggested two approaches to email thread summarization: collective message summarization (CMS) that considers a multidocument approach, while individual message summarization (IMS) treats the problem as a sequence of single-document summarization tasks. Both approaches involved selecting important sentences from email messages and compressing them (i.e., removing unimportant fragments). Experimental results on the Enron collection revealed that CMS performs better. Dredze et. al [33] developed an unsupervised learning framework for selecting summary keywords from emails using latent representations of the underlying topics in a users mailbox, rather than simply selecting keywords based on a single message in isolation. They tested the proposed approach extrinsically using two email tasks over Enron collection: automated foldering and recipient prediction. The results obtained demonstrate that summary keywords do indeed serve as a good approximation of message content.

2.5 MapReduce: Distributed Programming Framework

Over the past few years, there has been a remarkable interest in mapping a broad range of applications into MapReduce framework. MapReduce [30] is a distributed programming framework that allows developers to easily form their data processing tasks into a 2-stage model: map and reduce; the map stage runs in parallel on multiple process-

ing nodes that each handles a unit of data and produces a list of intermediate key-value pairs that are transparently sorted and finally processed by the reduce stage, which in turn runs on multiple processing nodes as well, to produce the final output. The run-time environment is responsible for all the rest, such as load balancing, scheduling, and fault tolerance, which makes it very convenient for developers to only focus on actual problem solving, rather than handling of low-level system issues. It also opens the door to process very large amounts of data on a cluster of commodity hardware.

The MapReduce pattern has been shown to be useful for many applications, e.g., those from machine learning [24, 72], relational data processing [92], semantic annotation [49], machine translation [34], and optimization problems [62].

There was recently work on efficiently implementing MapReduce and providing a high-level library-oriented parallel programming framework on multi-core and multi processor architectures [77, 54]. The significant speedup shown in these studies indicated that the MapReduce model is promising for scalable performance on shared-memory systems with simple parallel code.

There are also other compelling competitors to MapReduce as distributed/parallel programming environments, most notably the graphics processing unit (GPU) computing and Dryad.

GPU is not only a powerful graphics engine but also a highly parallel programmable processor featuring peak arithmetic and memory bandwidth that substantially outpaces its CPU counterpart [70], with a scalable parallel programming model [65]. Recently, there was some attempts to integrate MapReduce and GPU computing [23, 42]. Dryad [44] is another distributed programming framework that focuses the computations on a dynamic

dataflow graph whose vertices are physically mapped to processing nodes, whether on different computers or different CPU cores, and the developer has the control over the communication graph.

2.6 Chapter Summary

In this chapter, the research work on the general problem of entity resolution and the specific problem of identity resolution and modeling in email was reviewed. The chapter also cited some other research studies that looked at solutions of other problems in the email domain. It finally introduced MapReduce framework and other related distributed programming paradigms. The next chapter introduces the first step in the proposed approach to the specific dissertation problem, modeling identity in email.

Chapter 3

Identity Modeling

The first step in automatically resolving a personal name that is mentioned in the body of an email is to identify who the potential referents are, before choosing one of them. To repeat that process for any name-mention, the system needs to identify all possible people who are participating in the email collection. Identification here means building models of identity of those people that can computationally leveraged by the resolution algorithm. Therefore, two major questions are addressed:

1. How can identities in email collections be represented?

2. How can such model be computationally exploited in resolving observed mentions?

This chapter presents answers to both questions. First, some definitions are introduced in Section 3.1. The design of a representational model of identity is then presented in section 3.2, followed by an implementation in Section 3.3 [37]. The representational model is intrinsically evaluated in Section 3.4. Based on the representational model, Section 3.5 finally shows how a computational model, that is exploited in the resolution task, is built [38].

3.1 Definitions

In order to be precise about what an identity means in the context of email collections, three foundational concepts must be distinguished: (1) an entity, (2) an identity, and

(3) a model of identity.

An *entity* indicates a communicating party that is involved in any communication process in the email collection and that acts in some way that can be observed. There are generally three different types of entities in email collections: (1) a person, (2) a group of people, as in the case of a mailing list with a unified email address, and (3) a machine (e.g., an automatic email-generator that sends a periodic message reporting on stock prices.)

The notion used in this dissertation of a personal *identity*, the focus of the research, is somewhat more fine-grained:

- One *person* may adopt more than one *identity* (e.g., striving to completely separate their persona at work from a business that they run from their home trading merchandise on eBay.)

- An identity may have several (forms of) names that refer to that identity. A person whose name is "Robert E. Bruce" may also be known for example as "Robert," "Rob," "Bob," "Rob Bruce," or "Mr. Bruce."

- An identity might have multiple email addresses and can send email from any of them.

- Sometimes, several identities may send email from the same address. This happens with administrative assistants to senior executives, for example. It also happens frequently in customer service replies.

- An identity may also be referred to by role, for example as in "the *executive director* accepted the offer."

In the context of emails, identities rather than persons are modeled, because that is typically as far as the observable data will reveal. In fact, real identity is a hidden variable but the usage in emails is observable, so the hidden variable are estimated based on the observations. A representation of an identity is referred to as a *model of identity*. Using this model, an automatic system can computationally deal with the concept of a real identity. More details on how the models of identity are designed, constructed, and used in the context of email collections are presented in the following sections.

3.2 Representational Model of Identity

Identity is at best imperfectly observable in informal communication. People might sometimes intentionally hide their identities. So, what a model of identity can represent is just a merged set of "observable" features of the same identity. Evidence must therefore be combined from the available sources and reason based on that evidence if models of identity with the greatest possible degree of confidence are sought.

This section first identifies the observable elements of identity from which the models can be built. Next, sources of evidence, in the context of email collections, where those elements can be observed are presented. Finally, the design details of the proposed model are introduced.

3.2.1 Observable Features of Identity

Two basic elements that can be exploited in modeling the identity are identified:

1. **Attributes** that characterize observable distinguishing features of an identity. There

are two types of these attributes:

- **Referential attributes** that represent relatively stable explicit features of the identity such as name and email address.
- **Behavioral attributes** that represent basic communication features of the identity, such as patterns of communication or selection of discussion topics.

2. **Associations** that can serve as a basis for linking attributes together (e.g., email address to a name,) attributes to identities (e.g., reference to an identity,) and models of identities to other models of identities (e.g., model instance to another.) These associations provide greater support for inference when observed more often.

3.2.1.1 Referential Attributes

The most common personal attribute that is available in email collections is the email address. Email addresses are particularly useful because they control the routing of email and are thus strongly bound to identity. A second common personal attribute in email collections is the name. Names are often found in locations (e.g., headers or signature blocks) that make it relatively easy to associate them with email addresses. Names for the same identity can, of course, appear in different forms (e.g., full name, first name, or nickname). An observable instance of a referential attribute is called a *reference*. References can be observed either in the headers of the email (e.g., an email address) or in free text (e.g., "Bob" as in "I wrote to *Bob* yesterday".) In the latter case, a reference is called a *mention* as well.

3.2.1.2 Behavioral Attributes

A substantial number of behavioral attributes can be detected, including topic choice, lexical features (e.g., characteristic misspellings or use of emoticons,) stylistic features (e.g., whether new and quoted text are typically interleaved, or tendencies towards terseness or verbosity,) frequent correspondents (both individually and as groups,) conversational initiative, temporal rhythms (e.g., at what times on which days is email being sent,) and response times [37].

In the suggested model of identity, referential rather than behavioral attributes are focused on exclusively because (1) that is where the intuition was strongest, and (2) the referential attributes are the basis of the mention resolution technique as discussed in Chapter 4.

3.2.1.3 Associations of Attributes

Several types of evidence can be used to reason about associating two attributes of the same identity. Perhaps the most obvious is *attribute orthographic similarity*. For example, if "joe.engle@enron.com" as an email address and "Joe Engle" as a name were both observed, perhaps this similarity would support an inference that they refer to the same identity. This notion of similarity can be extended using side information (e.g., knowing that "Bill" is a common nickname for "William".) Attribute *co-occurrence* offers a complementary source of evidence. An example of attribute co-occurrence is the co-occurrence of an email address with any observed referential (or behavioral attribute,) within an email (e.g., in "Joe Engle <joe.engle@enron.com>".)

In general, the degree to which evidence of association is useful depends on the degree to which it is surprising (e.g., it might not be surprising to find "Joe Engle" is associated with "joe.engle@enron.com" but it is surprising to find it associated with "mrlilie@tvspot.com",) and the degree to which it is reinforced (since random variation will naturally produce some surprising but meaningless coincidences.) It is this tension between surprise and reinforcement that makes it necessary to work with large collections if interesting associations are to be discovered. Small collections simply lack the potential for multiple instances of rare (and thus surprising) events.

3.2.2 Sources of Evidence

Figure 3.1: An example email message (Enron collection).

The typical structure of email messages provides a rich resource for extracting elements of identity. An example email message from the CMU Enron collection is shown in Figure 3.1.

Email message in general has two main sections; header section and body section. The message header comprises the basic metadata part of the email and consists of a set of RFC-822 [1] header fields (e.g., from, to, cc, bcc, date, and subject) along with other fields (e.g., routing fields)–some of which are optional and some of which are specific to the email client application. The whole set of the header fields is called the "main header."

There are a number of special header fields that are specific to the CMU version of Enron collection, e.g., "X-To" field in Figure 3.1. Typical fields that represent the parties communicating in the email consist only of email addresses without names. The names then come in separate fields–sometimes they do not appear in the same order as email addresses in the corresponding RFC-822 header fields.

The rest of the message is considered the "message body." The text that the sender of the message originally wrote (including signature blocks) is referred to as the "main body" or sometimes the "new text." The message body may also include "quoted text," text that appeared in older email messages, if the message was a reply to or forward of another. The quoted text may start with a (system generated) "quoted header" section that includes information that was in the main header of the quoted message. The quoted header is usually structured header that resembles RFC-822 header (as shown in the figure). They may also start with an introductory line that is in a human-readable form such as "— Original Message —" or "On Wed, 12 Apr 2006, Martinez-Boyd, John R. wrote:". Such lines are referred to as "semi-structured" header lines, because they often resem-

ble free rather than structured text. The body of the quoted message, referred to as the "quoted body," may in turn include another quoted message, and so on.

The lines of any email body (either main or quoted) can be further classified into salutation, free text, and signature sections. The salutation (if any) may appear as a separate line or at the beginning of the first line of body. The free text is considered the actual message that the sender intends to express to the recipient(s). The signature section may consist of a manually-typed signature (called a "free signature") and/or a relatively static set of system-generated signature lines (called a "signature block".)

Fortunately, the above typical structure of email messages provides a substantial amount of evidence that opens up opportunities for building identity models. There are three basic sources of regularities that represent sources of evidence for observations:

- ***Email standards*** in terms of the standard format of the header section of the email (RFC-822), although it might not be perfectly adopted by the email client. This kind of metadata is naturally structured, and information extracted from it is often reliable.

- ***Email-client behavior*** represented by the rules followed by the email client when a user composes a new email or replies to another. An example of such behavior is the format of the quoted header that often embedded in the body of a reply-email. Sometimes, RFC-822-like quoted header is also included, as illustrated in Figure 3.1. The client also determines the quotation style of the replied-to message. Another behavior is the static signature block that the client automatically attaches to the end of each email. This source of evidence is not as reliable as the standards

for two reasons (1) it can be manually modified by the user since it is included in the body section of the email, and (2) it is client-dependent (i.e., not standard,) thus its format might be different in different clients.

- ***Regularities from user habits*** that the user often does when writing an email, such as greeting the recipients(s) by name at the beginning of the email, signing at the end, and interspersing quoted text with new text within an email in a way that is evocative of a conversation. Since they are more informal, these regularities may be harder to observe hence less accurate. However, they become very informative once observed and correctly extracted.

Section 3.3 describes how these sources were exploited to extract the basic referential attributes of the identity model.

3.2.3 Model Design

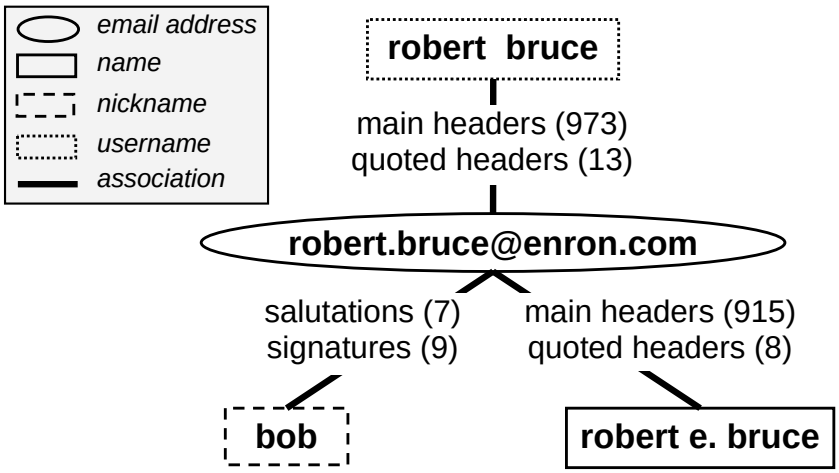


Figure 3.2: An example representational model of identity.

In a collection of emails, individuals often use different email addresses, multiple

forms of their names, and nicknames. The idea is to capture as many as possible of these different types of attributes that can be observed in one representation; in this dissertation, this is called a *representational model of identity*.

An identity is modeled as a set of referential attributes, much in the same way that WordNet[1] models meaning (i.e., a word sense) as a set of words that express that meaning. Four types of referential attributes are extracted: email addresses, names, nicknames, and usernames. Names, nicknames, and usernames are distinguished as follows:

- The string that is attached to the email address in the header is considered a "*name*". Many forms of names are observed in the header but the most frequent are "*First Last*", "*Last, First*", and "*First MI Last*". Individual tokens found in full names are further distinguished for the task of mention resolution, as explained in section 3.5.

- The string that is used to refer to an identity in the salutation and signature lines is considered a "*nickname*."

- "*Username*" is a tokenized version of the substring that precedes "@" in the email address; for example a username extracted from "susan_scott@enron.com" is "susan scott". The user name is sometimes useful in absence of any other type of names.

The model is built on pairwise co-occurrence of referential attributes (i.e., co-occurrence associations.) For example, an "address-nickname" association $< a, n >$ is inferred whenever a nickname n is observed in signature lines of emails sent from

[1] http://wordnet.princeton.edu

email address a. The observed name could actually be a first or last name, but it is always treated differently as a "nickname." Each association is weighted by the occurrence frequency that reflects the strength of its observation evidence.

Figure 3.2 depicts an example representational model of identity. In this simple example, three different associations were observed; each has "robert.bruce@enron.com" co-occurred with a different referential attribute. Sources and frequency of observation are indicated on each association. Notice that for each type of association, there are specific sources of evidence. A model can be viewed as an undirected graph in which the nodes represent attributes and the edges represent associations. Considering all the nodes in one graph, one model is one connected component.

Since an email address is bound to one personal identity (except in rare cases, when more than one person share the same email address,) email addresses serve as the basis on which the models are built. This is achieved by mandating that at least one email address must appear in any observed association. Therefore, only four types of associations exist; each associates an email address with one of the four different attributes listed above. Details on how each type is extracted are provided in Section 3.3. Notice that the selection of email addresses to be the basis in building the models facilitates subsequent merging of model instances of same identity by reducing it to simply linking different email addresses, as discussed in Section 3.3.3.

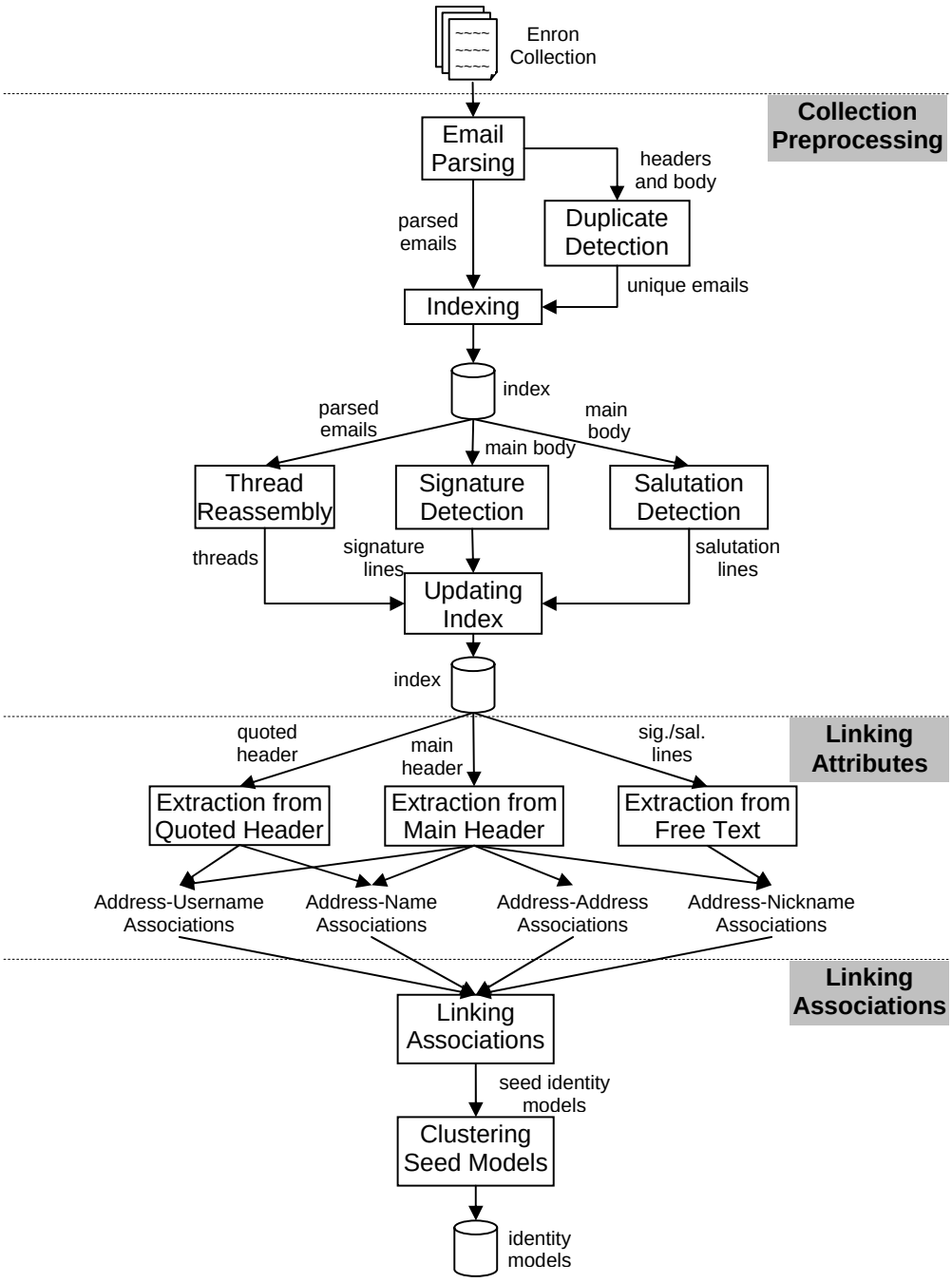


Figure 3.3: Data flow for identity modeling.

3.3 Implementation: *IdModeler*

In this dissertation, a system called *IdModeler* is implemented to populate the representational model of identity for the Enron collection. This section presents an overview of the implementation, followed by the details of each step in the subsections.

Figure 3.3 shows the feed-forward implementation for preprocessing and populating the identity models. The system can be outlined as follows:

1. **Collection Preprocessing**

   (a) Duplicates of emails in the collection are detected and information about these duplicates is retained as metadata attached to the unique copies.

   (b) All unique emails are parsed into main header, main body, quoted header(s), and quoted bodies. Email addresses and names (if available) of the sender and receivers(s) are also extracted from the headers.

   (c) The unique emails are indexed with each parsed part indexed in a separate field for future retrieval.

   (d) Threads are then reassembled using the quoted text technique proposed by Lewis in [51]. Each email is linked to its parent in the reply chain with a score that indicates how confident the system is in linking these two emails.

   (e) Signature blocks and salutation lines in the main body are detected and labeled for each email.

   (f) The threading links and labeled signature and salutation lines are added to the index.

2. **Linking Attributes**

   (a) For each email address that appears in the main header, nicknames (for the person who uses this address) are detected from manually-typed signature and salutation lines. This kind of association is called *address-nickname* association.

   (b) Three other types of associations are also extracted in this phase. *Address-name* associations are extracted from the main and quoted headers when both names and email addresses are provided together. Association of two email addresses for the same person (i.e., *address-address* associations) are also detected specifically in the Enron collection. Finally, an *address-username* association is considered between every email and its username part. The frequency of observing the email address is assigned as the frequency of such association.

3. **Linking Associations**

   (a) Having detected attributes and associations of them, all address-name, address-nickname, and address-username associations that have a common email address are first linked together in a graph whose nodes are the attributes and edges are the associations. This is valid as long as it is assumed that one email address is used by one identity. The graph now has a number of disconnected components equal to the number of detected email addresses. This initial set of models are called "seed models," since they are very simple models that each is built of only one email address.

(b) Address-address associations are then leveraged to merge such seed models into a more complex set of disconnected components (or clusters), each of which is a model of identity.

(c) A further simple merging step, that is based only on shared full names that have high degree of association strength, is implemented.

(d) Mailing lists are often attached to different names (probably the names of the sending members) in different header sections. Therefore, the models that have a long list of different full names are filtered out, and the focus will only be on the potential personal identities.

The following subsections describe each implementation step in greater details.

3.3.1 Collection Preprocessing

Email Parsing *IdModeler* has a parser for the main header of Enron emails. For the sender and recipients header fields, a set of regular expressions are used to detect names and email addresses. Different forms of names are supported for extraction.

In the CMU version of Enron collection, the email addresses and the associated full names are separated into two different sets of headers, as shown in Figure 3.1. Email addresses are included in "From, To, Cc and Bcc" headers, while corresponding names appear (when they are present at all) in the "X-From, X-To, X-Cc, and X-Bcc" headers (called X-header fields). Surprisingly, the number of entries in the address header does not always match the number in the corresponding name header. *IdModeler* relies on orthographic similarity to link a name (if possible) to the appropriate email address.

In order to identify the main (original) body of an email message, *IdModeler* first has to detect any quoted text included in that message. The quoted text generally appears in two forms. The first, used generally for forwarded messages (and sometimes for replied-to messages, by some email clients), typically consists of an entire message (i.e., the header section of the quoted message in RFC-822 format, followed by the main body of that quoted message) in its original format. In the other common style, used generally for messages that are being replied to, just the body of the quoted message appears (usually with each line marked on the left by some special character such as ”>” or ”|”). In either form, the quoted text is normally introduced by a single line that may include information extracted from the header of the message being quoted (e.g., the sender and the date). Quoted messages can themselves be quoted, and the two forms can be interleaved (as could happen if a forwarded message is later replied to, for example).

IdModeler detects the quotation style normally associated with forwarding by using a set of regular expressions that first detect the common forms of the system-generated head-line then detect the different header fields that come next. Because quoted body text is not generally marked in this format, it treats all of the lines following as the body of the quoted message. The process of quoted text detection is then repeated on that body. Detection of quoted body text in the form associated with replies is also performed.

Duplicate Detection In implementing *IdModeler* , two emails are considered duplicate if they have exactly the same: (1) email addresses of sender and receivers, (2) subject, and (3) body (after being normalized by Lucene's[2] standard tokenizer). This process resulted

[2] An open-source Java search engine available at http://lucene.appache.org/

in detection of 268,980 duplicate emails, about 52% of the whole CMU Enron collection. Subsequent processing was therefore restricted to the remaining 248,451 unique emails. This is just slightly fewer than the 250,484 unique emails found by Corrada-Emmanuel by using an MD5 hash function for duplicate detection [25]. The additional discovered duplicates could result from inconsequential differences such as date formats, layout differences, or optional header fields to which MD5 is sensitive.

Thread Reassembly Threads (i.e., reply chains) are imperfect approximations of focused discussions since people sometimes switch topics within a thread (and indeed sometimes within the same email). Threads are nonetheless expected to exhibit a useful degree of focus and therefore adopted in this dissertation as a computational representation of a discussion. Although thread relationships are sometimes encoded in an optional in-reply-to header field, reliable thread reconstruction generally involves more complex processing [51, 63, 93]. *IdModeler* adopted a technique similar to that introduced by Lewis [51] in which the quoted header fields and quoted text are used to form a query in an effort to identify the parent email in the same thread. If no parent is found in the collection, the child email is flagged as "having broken link to parent." That flag is used to determine whether the quoted parts of an email contain otherwise unavailable information that should therefore be processed.

Signature Line Detection *IdModeler* defines signature lines as lines that often appear similarly-formatted near the end of email messages sent from the same email address. Two types of signature lines are of interest: (1) machine-generated static signature lines,

from which signature blocks can be detected, and (2) lines containing manually typed free-form text from which nicknames can be detected. *IdModeler* identifies candidate regions for signature lines by identifying blank lines in the body text in a (possibly quoted) email, using those blank lines to separate blocks of contiguous text, and then focusing on the last two blocks of contiguous text. For each line in these two blocks, *IdModeler* tokenizes the text in that line using Lucene's standard tokenizer and then count the number of lines in which the same pattern of stemmed tokens are found. Use of a standard tokenizer suppresses some variations that are observed in hand-typed signatures (e.g., "-Dave", "dave", "–Dave", and "Dave"). Tokenized lines with a message-count that meet a pre-established threshold (in the conducted experiments described in Section 3.4.1, exactly 2 for the "weakest evidence" threshold and at least 3 for the "stronger evidence" threshold) are passed to further processing as signature lines. Signature blocks are reconstructed in order from the original (untokenized) lines, while nickname detection (described below) is performed using tokenized lines.

Salutation Line Detection The process of detecting salutations (brief greetings at the start of a message) is similar to that used to detect signature blocks. *IdModeler* 's implementation of salutation line detection is quite rudimentary, focusing solely on lines that contain nothing but a salutation. The process starts by identifying the first line of the main body of every email message in which the recipient's email address is alone in the "To" header (cc to other people is allowed) and in which the message body contains at least two lines. Lines that start with "fyi" or that end with "?", "!", "." and any line with a length exceeding two words are then filtered out. What remains is normalized using

Lucene's standard analyzer and then consider each normalized line that appears exactly three times (for the "weakest evidence" condition) or at least four times (for the "stronger evidence" condition) to be a salutation line. Limitations of this process are: (1) salutations embedded in the start of a longer line will not be detected, (2) complex salutation forms (e.g., "hey Bobbie Rae!!") are not detected, and (3) no use is made of evidence found in messages addressed to more than one primary recipient.

Indexing Emails All of the emails in the test collection are indexed using Lucene. Each identifiable part of the email (the header, main body, salutation, signature lines, quoted header and quoted body) are automatically identified using pattern matching and separately indexed. Any free text found in the subject header, main body or quoted body is stemmed using the Porter stemmer [76], and stop-words are removed using the Lucene stop-word list.

3.3.2 Linking Attributes

Extracting Attributes and Associations from Main Headers "Address-name" associations are constructed by mapping an email address in a header to a name in the corresponding X-header. Email addresses are sometimes found together with names in the X-headers; such cases are also extracted. If an address found in this way in the X-header differs from the address in the corresponding address header, an "address-address" association is also extracted. This process results in identification of 70,214 address-name associations and 10,708 address-address associations. Of course, some of these associations could be incorrect, since both attribute detection and orthographic matching employ

imperfect heuristics.

Extracting Attributes and Associations from Quoted Headers Once all quoted headers have been identified, address-name associations are then extracted from those quoted headers in which email addresses are associated with names (From, To, Cc and Bcc.) This process results in the extraction of 11,870 additional address-name associations that were not extracted before from main headers, increasing the relative recall by about 17%. An additional 9,289 address-name associations that were previously extracted from main headers were observed again in the quoted headers. Quoted messages normally (and in the CMU Enron collection) do not use separate header fields for names, so no address-address associations are extracted from quoted headers.

Nickname Extraction *IdModeler* uses a precision-oriented approach to detect nicknames from signature and salutation lines. For each known email address, *IdModeler* applies a set of filtering rules to each detected unique signature and salutation line. First, a set of hand-selected stop words (e.g., “Hi” or “Dear” for salutations or “Thanks” or “Regards” for signatures) are removed. Any line that includes one or more non-alphabetic characters (after Lucene’s normalization, this mainly removes digits), and any line found in a signature block after the fourth line in that block are then filtered out. Lines with more than one word are then compared with the first part of the email address (before the ”@”) using an edit distance measure, and those with little similarity are discarded. The entirety of any remaining line is then considered to be a nickname, with its frequency in salutation and signature lines serving as a measure of the strength of evidence for that

assignment. A total of 3,151 address-nickname associations were discovered in this way. There are, of course, many ways in which the exhaustivity (i.e., recall) of this process could be improved. For example, cue words (e.g., "Mr.") and name lists are often used by named entity recognition systems.

3.3.3 Linking Associations

Building Initial models The foregoing processes resulted in extraction of a total of 82,084 address-name associations, 19,708 address-address associations, and 3,151 address-nickname associations. These associations form links in an undirected graph on which *IdModeler* performs agglomerative clustering to identify connected components, each of which represents an identity model. Because it treats addresses as unique pivot points, address-name, address-nickname, and address-username links result in no reduction in the number of entities. Address-address associations can, however, connect two disconnected model instances (although the accuracy of those associations will naturally depend on the strength of the evidence.) The complete process resulted in 66,715 models that together cover 77,420 unique email addresses (58% of the 133,581 unique email addresses identified in the collection).

Merging Multiple Model Instances Since a person can have multiple email addresses and detecting these association of addresses can enrich the models, *IdModeler* have adopted a simple technique that merges two identities whenever they share a common full name, with a relatively high frequency in both. Only one merging pass is performed over the detected models. Since each identity model must have at least one email address, the new

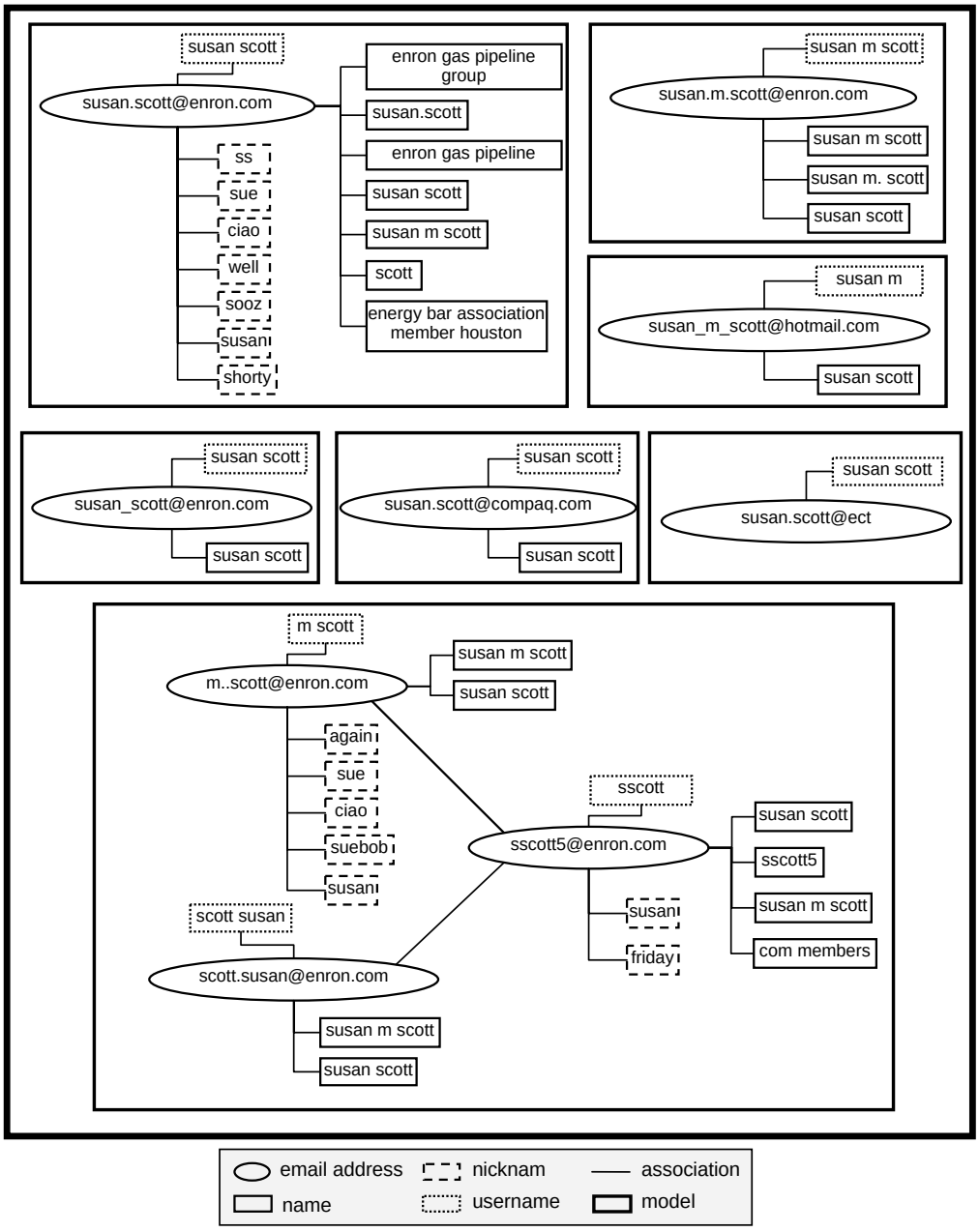


Figure 3.4: Multiple detected model instances for the same real identity.

(merged) identity inherits the union set of these addresses. This strategy may, of course, incorrectly merge different people with common names (e.g., John Smith), but within a single organization people will often adopt different name variants to prevent just that sort of confusion. Figure 3.4 illustrates a real example, extracted from Enron collection, that merges 7 instances of models for "Susan Scott." In the whole set of models, this process merged only 77 pairs of initial model instances.

Detecting Mailing lists The goal is to resolve personal identities, but the CMU Enron collection contain email addresses for mailing lists as well. As a simple (and naive) expedient to guard against inappropriate attempts to build a personal identity model for a mailing list, all identity models that have more than 10 different associated full names are removed.

3.4 Evaluation

The perceived accuracy and utility of the representational models of identity extracted from the CMU Enron collection are intrinsically evaluated. There are three levels of extraction to evaluate: attributes, associations, and identities. The evaluation focuses on the associations because (1) association accuracy can be extrapolated to estimate the accuracy of identity extraction, and (2) errors in attribute extraction can incidentally be detected when assessing the relationships that an attribute participates in.

The total number of extracted associations (95,943) is far larger than they could be assessed, so it was necessary to sample that set in some way. The 12-cell stratified sampling strategy shown in Table 3.1 was adopted to characterize the results based on

Table 3.1: Stratified samples and population sizes.

Association Type	Weakest Evidence	Stronger Evidence
Address-Name Associations		
Main header only	50 / 29677	50 / 31248
Quoted header only	50 / 8042	50 / 3828
Both headers	50 / 9289	
Address-Nickname Associations		
Salutations only	50 / 272	50 / 465
Signatures only	50 / 172	50 / 1754
Both types	50 / 490	
Address-Address Associations		
Main header	50 / 6514	50 / 4194

association type, source of evidence, and strength of evidence. Samples from all types of associations, except address-username, are judged. The "weakest evidence" was defined as the minimum absolute detected strength of evidence (indicated by 1 observation in headers, 3 in salutations, and 2 in signature lines), and the "stronger evidence" was defined as any other stronger condition. Address-address associations has only one source of evidence, so they were stratified based only on the strength of evidence in that case.

3.4.1 Evaluation Measures

Three cases of correct associations (for which real examples from the Enron collection can be found in Table 3.2) are distinguished:

1. "not informative": A correct association is "not informative" if a simple rule could have been used to extract one of the associated attributes from the other (e.g., a name from an email address), or if simple string matching would have indicated

that two attributes (e.g., email addresses) were likely for the same person.

2. "somewhat informative": A correct association is "somewhat informative" if its detection would have been possible, but only with some side knowledge (such as a list of common names.)

3. "very informative": A correct association is "very informative" if the information contained in the name and/or address(es) was not sufficient to reliably infer the association.

In order to measure both the accuracy and utility of the associations and based on the above categories, three evaluation measures were defined as follows:

$$Accuracy = \frac{|correct|}{|judged|} \tag{3.1}$$

$$Informativeness = \frac{|somewhat\ informative| + |very\ informative|}{|judged|} \tag{3.2}$$

$$Strong\ Informativeness = \frac{|very\ informative|}{|judged|} \tag{3.3}$$

Accuracy is the percentage of judged associations that is correct, regardless of usefulness of these associations. Two levels of utility are then supported: *Informativeness* is the percentage of judged associations that are informative (i.e., non-trivial and thus perhaps useful for subsequent processing by automated techniques) according to the definition above, and *Strong Informativeness* is the percentage of judged associations that would provide new information to a human user.

Notice that the fraction of the judged associations that are not informative (i.e., neither somewhat informative nor very informative) can serve as an estimated accuracy

Table 3.2: Examples of different types of correct associations.

Judgment	Association Examples
Not informative	june-deadrick@reliantenergy.com $\Longleftrightarrow$ "june deadrick" robbie.lewis@enron.com $\Longleftrightarrow$ "robbie lewis"
Somewhat informative	terriecovarrubias@hotmail.com $\Longleftrightarrow$ "terrie covarrubias" randal.maffett@enron.com $\Longleftrightarrow$ "randy"
Very informative	lemelpe@nu.com $\Longleftrightarrow$ "phyllis" piazzet@wharton.upenn.edu $\Longleftrightarrow$ "tom"

of an obvious baseline approach that links attributes by string matching techniques.

$$Baseline\ Accuracy = 1 - Informativeness \tag{3.4}$$

3.4.2 Judgment Process

Based on the above definitions and evaluation measures, one independent assessor, who had experience with email search system design, was recruited to judge the accuracy and potential utility of the sampled associations based on the following criteria: an address-name or address-nickname association is considered incorrect if either of the two attributes is incorrectly extracted or if both of them are correctly extracted but linking them is incorrect. Otherwise, the association is considered correct. For address-address associations, only the correctness of the linking was assessed. For correct associations, the assessor followed the definitions given in Section 3.4.1 to label each correct association. To simplify the judgment process, a java GUI tool, that is specifically designed to search the Enron collection using Lucene, was developed. A screen shot of the tool is shown in figure 3.5. The interface enables the assessor to search in headers (by ei-

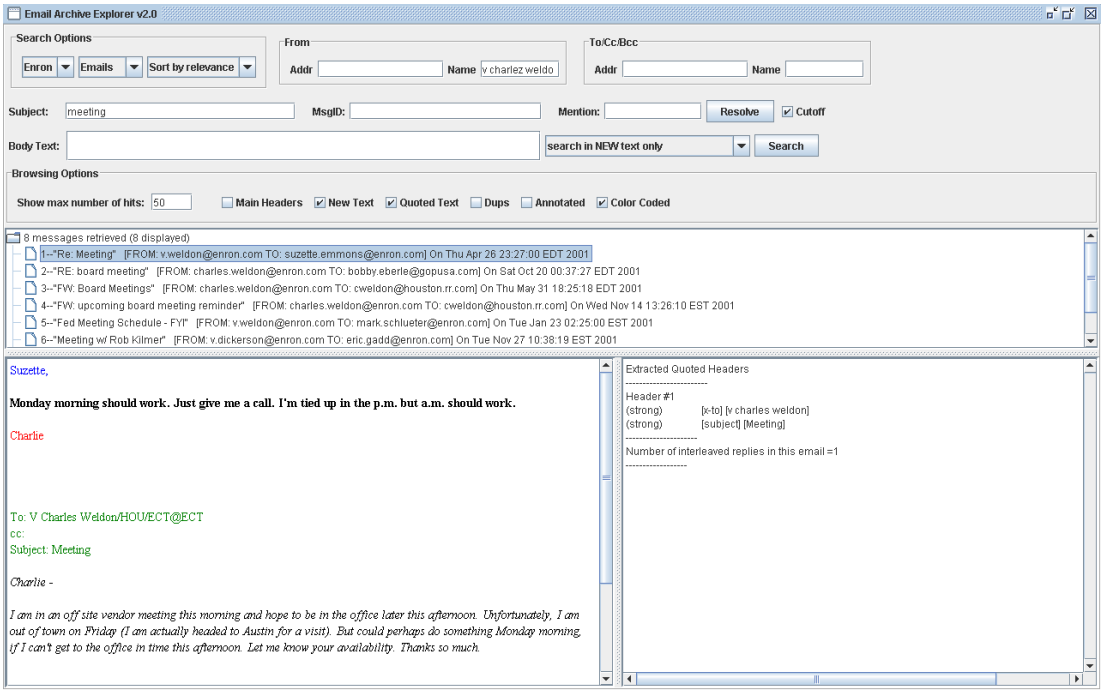


Figure 3.5: The GUI used in the judgment process.

ther email address and/or name), subject, or body text. The assessor can also restrict the search to the main body, quoted text, or both. The tool displays a ranked list of emails that match the query, and gives the user a chance to see each result in either raw text or HTML that can be customized to display any of the different email parts shown earlier in Figure 3.1. For each email listed in the search results, a list of the extracted attributes and associations is presented as well. The assessor then can use the assessed attributes to compose search queries that may help find evidences for the current judged association. If the assessor could not judge a specific association from the available evidence (including counter-evidence), then she could choose a fifth option: "Can't tell." This occurred in only 19 of 600 cases. As Figure 3.6 illustrates, stronger evidence never hurt and sometimes helped, and associations found in salutations were surprisingly reliable.

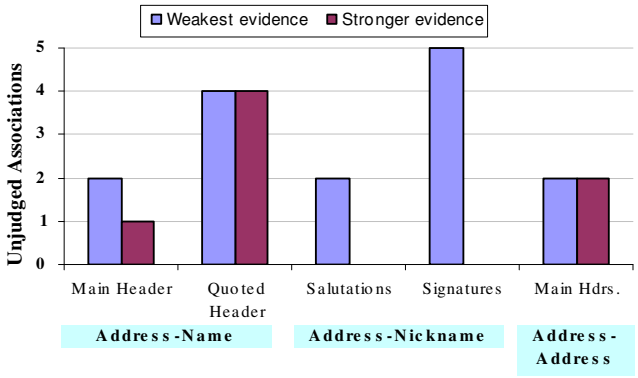


Figure 3.6: Unjudged Assocociations.

3.4.3 Results

Figures 3.7 and 3.8 show the evaluation results for address-name and address-nickname associations respectively. Each graph in those figures shows one type of association, one metric (on the vertical axis) and each source of evidence (along the horizontal axis). For ease of comparison with the "both" condition (where weakest and stronger evidence are combined) a weighted average is shown between the weakest and stronger evidence bars for the single sources of evidence. Figure 3.9 shows the results for address-address associations. The three performance measures are plotted side by side in that case since they are all based on just one source of evidence (main headers).

3.4.3.1 Accuracy

As Figures 3.7(a) and 3.8(a) illustrate, 100% accuracy was achieved whenever multiple sources of evidence supported an extracted association, regardless of the strength of each component of that evidence. Address-name association was nearly perfect in every case; while the minimum accuracy in any single source of evidence was 80% (in the case

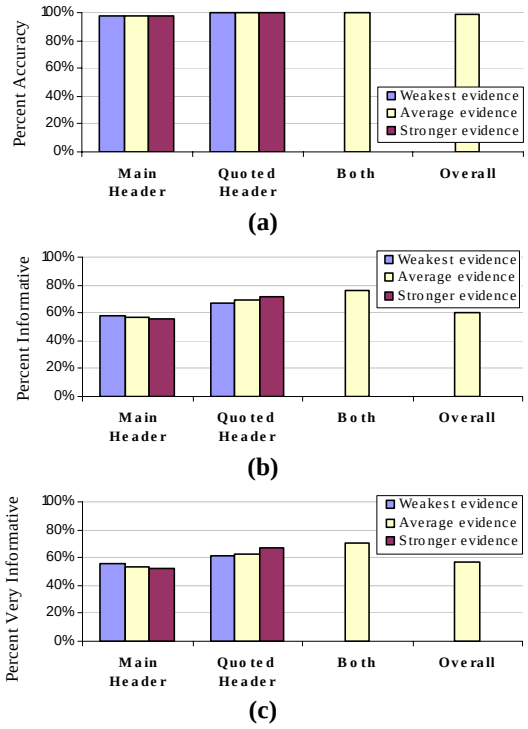


Figure 3.7: Evaluation of Address-Name associations.

of weakest evidence in signature-based address-nickname associations.) This is not surprising, since the proposed approach to address-name association exploits regularities in system behavior, while address-nickname association is based on more variable human behavior.

As Figure 3.8(a) illustrates, increasing the strength of evidence improved the accuracy of address-nickname association extraction from signatures, but a similar effect was not evident for salutations. This may result from deficiencies in the process of determining the start of signature blocks. As the weighted average indicates, however, there were relatively few cases with the weakest evidence.

As Figure 3.9 shows, address-address associations are almost always accurate. If the overall accuracy of address-name association extraction (squared) is factored in, sin-

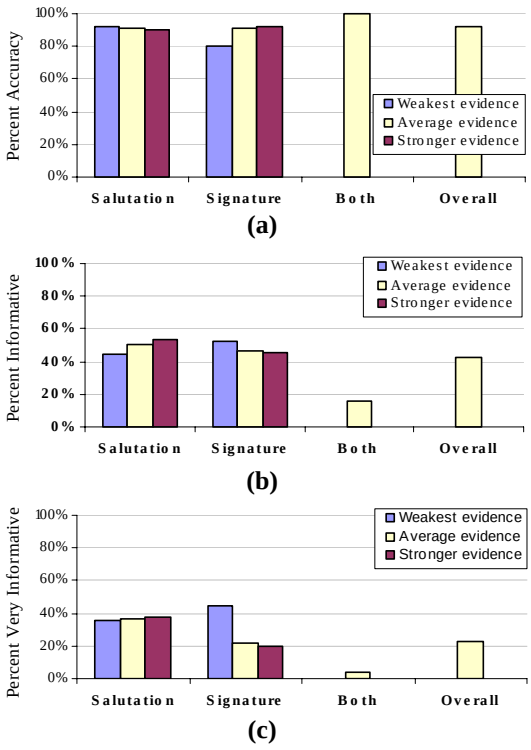


Figure 3.8: Evaluation of Address-Nickname associations.

gle address-address associations would be expected to be completely correct 97% of the time.[3] Inspection of the sample indicates that most address-address associations are for Enron employees. This might be an artifact of the cleanup preprocessing that was done before the release of that version of Enron collection.

The average identity model includes 1.23 address-name associations, 0.16 address-address associations, and 0.05 address-nickname associations, so the overall accuracy of an entity can be estimated as $(0.98)^{1.23} * (0.97)^{0.16} * (0.92)^{0.05} * 100 = 96.7\%$ (assuming independence between extraction of different associations).

Using the definition of the baseline accuracy in Section 3.4.1 and given the infor-

[3] A correct address-address association depends on the accuracy of two address-name associations (evaluated as 0.98^2) and its extraction accuracy (evaluated as 1.0).

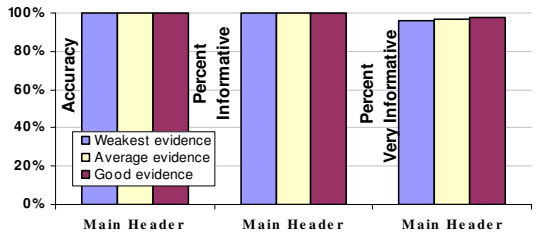


Figure 3.9: Evaluation of Address-Address associations.

mativeness results illustrated in figures 3.7(b), 3.8(b), and 3.9(b), the baseline accuracy can be estimated as $(0.39)^{1.23} * (0.58)^{0.16} * (0.03)^{0.05} * 100 = 24.2\%$

3.4.3.2 Utility

As figures 3.7(b)-(c) and 3.8(b)-(c) illustrate, address-nickname associations are generally less informative than address-name associations. This makes sense, since nicknames are usually just one word, while full names typically include two words. Interestingly, nicknames that appear in both salutations and signatures are almost uniformly uninformative. Most nicknames in that category belong to Enron employees. Enron email addresses were usually constructed from their first and last names separated by dot, thus leaving little opportunity for surprise. The opposite is true for address-name associations: when observed in both main and quoted headers, informativeness is higher than when the same type of association is inferred from a single source of evidence. In that case, it turns out that most of the email addresses were from non-Enron domains, for which it is less common to find the full name embedded within the email address.

Surprisingly, Figure 3.8(c) shows that in the case of evidence from signatures, the most informative associations resulted from the weakest evidence. Since this was exactly

the case in which the accuracy was lowest, this points up the importance of considering both accuracy and informativeness. A focused effort to improve the accuracy of address-nickname association extraction from signatures could therefore have a high payoff.

Overall, address-address associations were almost always very informative (97%), address-name associations were very informative in more than half the cases (57%), and address-nickname associations were very informative in about a quarter of the cases (23%). This is considerably higher than what was initially expected, suggesting that further work on increasing the accuracy of the extraction, and extending the range of evidence that can be productively exploited would be a good investment.

3.5 Computational Model of Identity

The model described in section 3.2 represents the identity of a communicating participant as a set of pairwise co-occurrence of referential attributes (i.e., co-occurrence "associations",) each weighted by the frequency of occurrence. That model captures what can be observed in the emails and can generally serve as a basis for more sophisticated models designed for specific computational tasks. This section describes how that model can be used in the context of name-mention resolution in emails. For that task, a "computational model of identity" is needed to help associate names to identities and vice versa. That is achieved in two steps: (1) labeling observed names, described in section 3.5.2, and (2) building the reasoning model, described in section 3.5.3. Section 3.5.1 first introduces some notations.

3.5.1 Notations

Chapter 4 proposes a probabilistic approach that ranks the candidates based on the estimated probability of having been mentioned. Formally, the goal is to estimate the probability $p(c|m)$ that a potential candidate c is referred to by the given mention m, observed in email e^m, over all candidates C.

A mention m is defined as a tuple $< l^m, e^m >$, where l^m is the "literal" string of characters that represents m and e^m is the email where m is observed.[4] In order to focus the research work on the resolution framework, the problem of detecting unresolvable mentions was not attempted. Consequently, the resolution process assumes that m can be resolved to a participant for whom at least one email address is present in the collection E.

3.5.2 Labeling Observed Names

For the purpose of resolving name-mentions, it is necessary to compute the probability $p(l|c)$ that a person c is referred to by a given "literal" mention l. Intuitively, that probability can be estimated based on the observed "name-type" of l and how often that association occurs in the representational model. T is defined as the set of 3 types of single-token name-types: first, last, and nickname. Just for simplicity, middle names and initials were not handled. Names that are extracted from salutation and signature lines are labeled as nicknames whereas full names extracted from headers are first normalized

[4]The exact position in e^m where l^m is observed can also be included in the definition, but it is ignored assuming that all exactly-matched literal mentions in one email refer to the same identity.

to “First Last” form and then each single token is labeled based on its relative position as being the first or last name. Usernames are treated similarly to full names if they have more than one token, otherwise they are ignored. Note that the same single-token name may appear as a first name and a nickname.

3.5.3 Reasoning

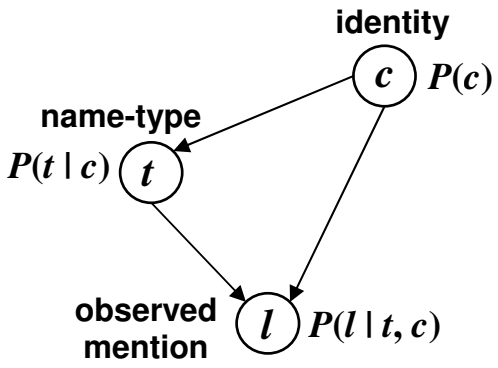


Figure 3.10: A computational model of identity.

Having tokenized and labeled all single-token names, the association of a single-token name l of type t to an identity c is modeled by a simple 3-node Bayesian network illustrated in Figure 3.10. In the network, the observed mention l is conditionally distributed on both the identity c and the name-type t. $p(c)$ is the prior probability of observing the identity c in the collection. $p(t|c)$ is the probability that a name-type t is used to refer to c. $p(l|t,c)$ is the probability of referring to c by l of type t. These probabilities

can be estimated from the representational model as follows:

$$
\begin{aligned}
p(c) &= \frac{|assoc(c)|}{\sum_{c' \in C} |assoc(c')|} \\
p(t|c) &= \frac{freq(t,c)}{\sum_{t' \in T} freq(t',c)} \\
p(l|t,c) &= \frac{freq(l,t,c)}{\sum_{l' \in assoc(c)} freq(l',t,c)}
\end{aligned}
\tag{3.5}
$$

where $assoc(c)$ is the set of observed associations of referential attributes in the represented model c.

The probability of observing a mention l given that it belongs to an identity c, with no assumption of a specific token type, can then be inferred as follows:

$$
p(l|c) = \sum_{t \in T} p(t|c)\ p(l|t,c) \tag{3.6}
$$

In the case of a multi-token names (e.g., John Smith), the first is assumed either a first name or nickname and the last is a last name. The probability is then computed accordingly as follows:

$$
p(l_1 l_2|c) = \{ \sum_{t \in \{f,n\}} p(t|c)\ p(l_1|t,c) \} \cdot p(l_2|last,c) \tag{3.7}
$$

where f and n above denote first name and nickname respectively.

Email addresses are also handled, but in a different way. Since each of them uniquely identifies the identity, all email addresses for one identity are mapped to just one of them, which then has half of the probability mass (because it appears in every extracted co-occurrence association).

The proposed computational model of identity can be thought of as a language model [75] over a set of personal references, so it is important to account for unobserved references. If a specific first name often has a common nickname (using a dictionary of

commonly used first to nickname mappings (e.g., Robert to Bob)), but this nickname was not observed in the corpus, then smoothing is needed [95]. That is achieved by assuming the nickname would have been observed n times where n is some fraction (0.75 in the conducted experiments) of the frequency of the observed name. That process is repeated for each unobserved nickname then they were treated as if they were actually observed.

3.5.3.1 Context-Free Resolution

This model can now be used to estimate how probable a name-mention l could refer to an identity c by computing $p(c|l)$ using Bayes' rule as follows:

$$p(c|l) = \frac{p(l|c)\ p(c)}{\sum_{c' \in C} p(l|c')\ p(c')} \tag{3.8}$$

All the terms above are estimated as discussed earlier. Notice that the model estimates that probability ignoring the context in which this mention is observed; that resolution model is called a "Context-Free" resolution.

3.6 Chapter Summary

Two models of identity are introduced in this chapter. The representational model associates names and nicknames to email addresses of a participant by observing co-occurrent attributes, while the computational model shows how to leverage those associations for the task of mention resolution using a simple 3-node Bayesian network. The next chapter introduces a richer model that takes the context of a mention into consideration in the resolution process, but it still builds upon that simpler computational model.

Chapter 4

Mention Resolution

The previous chapter presents an abstract model of identity that is built using the repetitive observations of personal attributes (e.g., email addresses and names) in the whole email collection, and shows how it could be used to resolve a name-mention without considering the context surrounding it. The problem addressed in this chapter is the resolution of a name-mention in a given email in the collection, i.e. given the (potentially observed) context in which the true referent was mentioned. The goal of this work is to develop a system that provides a list of potential candidates, ranked according to how strongly the system believes that a candidate is the true referent meant by the email author.

The discussion is this chapter starts with two motivating views in Sections 4.1 and 4.2 that both inspire the probabilistic approach for mention resolution described in detail in sections 4.3 and 4.4.

4.1 Finding Evidence: A Searcher's View

Imagine that a user is searching an email collection from an enterprise to figure out how a specific decision was made. During the search, the user finds that the decision was made during the conference call referred to in the email shown in Figure 4.1. The email clearly shows that the individuals "Sheila" and "Scott" are probably involved in the decision and so the user decides to look for some information about their identities.

Date: Wed Dec 20 08:57:00 EST 2000
From: Kay Mann<kay.mann@enron.com>
To: Suzanne Adams <suzanne.adams@enron.com>
Subject: Re: GE CONFERENCE CALL HAS BEEN RESCHEDULED

Did Sheila want Scott to participate? Looks like the call will be too late for him.

Figure 4.1: An email example from Enron collection.

Examining the header of the email, the mentions could not be resolved since neither "Sheila" nor "Scott" is the sender or recipient of the email. Moreover, the text of the email does not uniquely identify either person. The user extends the search to discover that this email was part of a longer discussion between a larger set of participants. Given the context of the name mentions, the user tries to find a clue in one of the emails in that discussion. If nothing is found there, the user may use the knowledge that for the email to have made sense to the recipient(s), the mention must have been unambiguous to both the sender and the receiver(s) of the email at the time it was sent. Consequently, the user may search the recent emails sent or received by those participants, trying to reconstruct the required context that may lead the user to the right referents. The user may also need to search for other emails or discussions that talk about the same topic in the near past or future. For example, the user may look at other emails about a GE conference call. Finally, once the user finds some evidence that provides a potential mapping of both mentions to identities, the user has the choice to stop and immediately draw a conclusion based on what has been found, or to continue to collect more evidence to gain more confidence in the resolution.

The intuitive evidence-based search strategy adopted in that example sheds light on

3 important questions:

- What kind of evidence to look for (e.g., which evidence can support the resolution of a given mention?).
- Where to look for evidence (e.g., which emails should be searched for evidence?).
- How to combine evidence to rank candidates (e.g., how can scores be assigned to candidates?).

These questions outline a framework for a family of mention resolution algorithms that are evidence-based. Each different set of answers to the above questions represents a specific strategy.

Motivated by this view, a heuristic approach for mention resolution, detailed in [36], was developed and evaluated. That approach was then modeled more rigourously by the generative view outlined in Section 4.2. The remainder of this chapter discusses the improved approach.

4.2 Generative Scenario: A Sender's View

The proposed probabilistic approach is motivated by a generative story of mentioning people in email. The story begins with the author of an email, intending to refer to a person in that email. To do that, the email author will:

1. Select a person to mention.
2. Select a context that is appropriate to mention that person.

3. Select a specific lexical mention to refer to that person given the context.

For example, suppose John is sending an email to Steve and wants to mention a common friend Edward. John knows that he and Steve know 2 people named Edward, one is a friend of both known by "Ed" and the other is his soccer trainer. If John would like to talk about the former, he would use "Ed," but he would likely use "Edward" plus some special terms (e.g., "soccer," "team," etc.) for the latter. John relies on the social context, or the topical context, for Steve to disambiguate the mention.

This story seems to answer the three questions raised by the searcher's view in Section 4.1. It proposes other "observed mentions" as the type of evidence to look for. It considers the "context" of a mention as the search space. Finally, it suggests that a generative probabilistic model can be used to rank candidates, based on the observed mentions in the context.

The steps of this scenario also impose a certain structure to the proposed solution:

1. A computational model of identity that supports reasoning about identities is needed. Chapter 3 presented such model.

2. Reconstruction of the context of the queried mention is needed. How this is achieved is described in Section 4.3.

3. A resolution technique that leverages both the identity models and the context to rank the potential candidates is needed. The technique is detailed in Section 4.4.

Within the above structure, this chapter presents a probabilistic approach that ranks the candidates based on the estimated probability of having been mentioned. Recall that

the goal is to estimate the probability $p(c|m)$ that a potential candidate c is referred to by the given mention m, observed in email e^m, over all candidates C. As introduced in Section 3.5.1, a mention m is defined as a tuple $< l^m, e^m >$, where l^m is the "literal" string of characters that represents m and e^m is the email where m is observed. m is called the "mention-query," and e^m is called the "mention-email." Details of the approach are introduced in the following sections.

4.3 Contextual Space

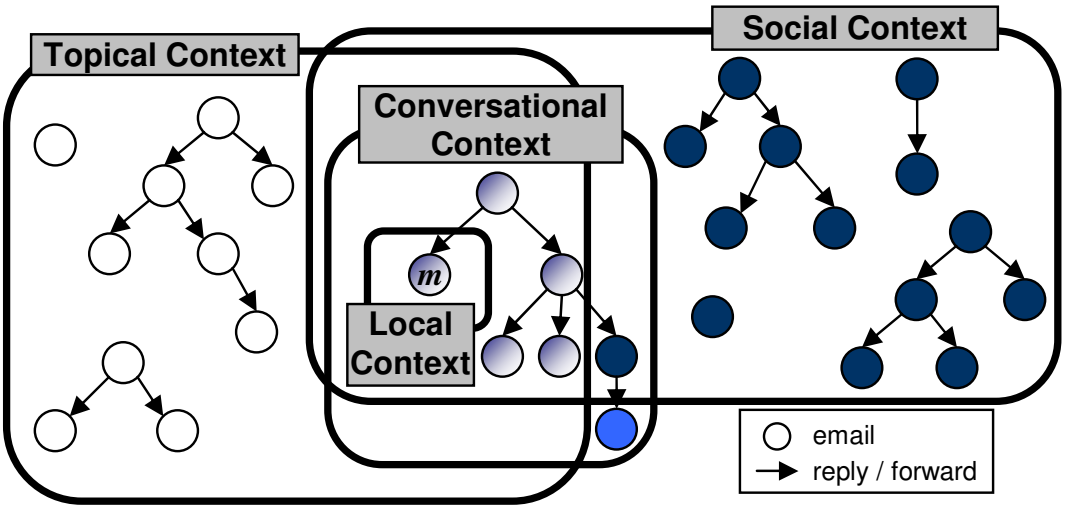


Figure 4.2: Contextual Space of an email m.

It is obvious that understanding the context of an ambiguous mention helps with resolving it. Fortunately, the nature of email as a conversational medium and the induced relationships between emails and people through communication, can reveal clues that can be exploited to partially reconstruct that context.

The contextual space $X(m)$ of a mention m is defined as a *mixture* of 4 types of contexts with λ_k as the mixing coefficient of context x_k. The four contexts (illustrated in

Figure 4.2) are:

1. ***Local Context***: the email e^m where the mention is observed.

2. ***Conversational Context***: emails in the broader discussion that includes e^m, typically the thread that contains e^m.

3. ***Social Context***: discussions that some or all of the participants (sender and receivers) of e^m joined or initiated at around the time of the mention-email. These might reveal some otherwise-undetected relationship to the mention-email.

4. ***Topical Context***: discussions that are topically similar to the mention-discussion that took place at around the time of e^m, regardless of whether the discussions share any common participants.

These generally represent a growing (although not strictly nested) contextual space around the mention in concern.

All mentions in an email are assumed to share the same contextual space. Therefore, the context of a mention is treated as the context of its email. However, each email in the collection has its own contextual space that could overlap with another email's contextual space.

K defines the set of the 4 types of contexts. A context x_k is represented by a probability distribution over all emails in the collection. An email e_j belongs to the k^{th} context of another email e_i with probability $p(e_j|x_k(e_i))$.

How each context is represented and how the distribution is estimated depend upon the type of the context. The following subsection explains that in detail.

4.3.1 Context Reconstruction

The "contextual space" of an email is defined as a mixture of four different contexts, each is a probability distribution over emails: (1) local, represented by the email message itself, (2) conversational, represented by the thread that contains the email, (3) social, represented by the other threads that involve some (or all) of the participants of the email, and (4) topical, represented by the threads that are topically relevant to the mention-query email. The social and topical contexts can also encode a notion of temporal similarity; this is modeled by assigning higher probabilities to messages that are temporally closer to the mention-email. Figure 4.3 illustrates the evidence used schematically. Reconstruction of

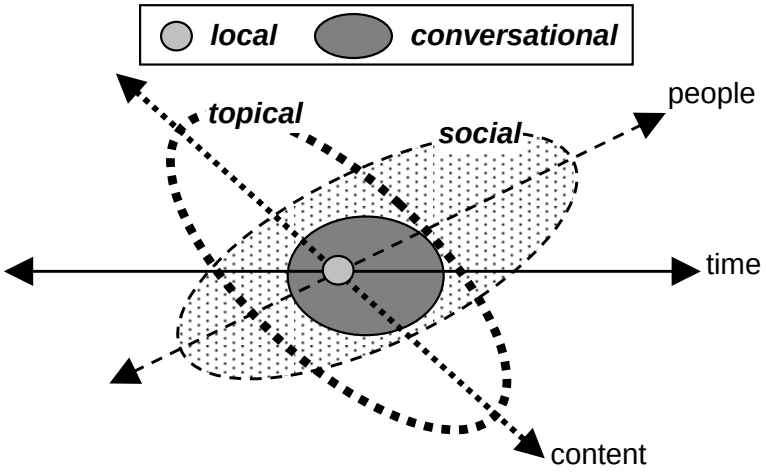


Figure 4.3: Contextual space of an email, relative to time, content, and people dimensions.

local context is pretty simple; e^m is assigned the whole probability mass.

In the case of conversational context, threads are adopted as a computational representation of a conversation in the implementation. Thread reconstruction results in a unique tree containing the mention-email. Although different paths or subtrees can be distinguished, a uniform distribution over all emails in the same thread was elected.

The reconstruction of the social and topical contexts are somewhat more sophisticated. The general approach adopted to estimate the probability distribution of the (social/topical) context of an email e_i is to first compute a (social/topical) context similarity score $s(e_i, e_j)$ of e_i and every other email e_j, and finally normalize the similarity scores (i.e., adjust them to sum to 1.0) to get the estimated probabilities $p(e_j|x_k(e_i))$.

Two main factors should be considered when computing the context similarity $s(e_i, e_j)$ of two emails e_i and e_j:

1. Their social/topical similarity $s_x(e_i, e_j)$

2. Their temporal similarity $s_t(e_i, e_j)$.

To account for both factors, the overall similarity is computed as the geometric mean of the two:

$$s(e_i, e_j) = \sqrt{s_t(e_i, e_j) * s_x(e_i, e_j)}$$

The following subsections discuss different ways of computing each of the required similarities. Notice that the approaches adopted to reconstruct the social and topical contexts were chosen for their relative simplicity, but there are clearly more sophisticated alternatives. For example, topic modeling techniques [60] could be leveraged in the reconstruction of the topical context.

4.3.2 Temporal Similarity

The temporal similarity of two emails should decay as the time difference increases. Earlier version of the resolution algorithm used the reciprocal rank of the absolute value of the time difference to achieve that effect [38]. This imposes a rather steep decay, which

Table 4.1: Temporal similarity functions

	Time Difference (d)	Rank (r)
Gaussian	$g_d(d) = ae^{-\frac{1}{2}(\frac{d}{T/4})^2}$	$g_r(r) = ae^{-\frac{1}{2}(\frac{r}{N_x(T)/2})^2}$
Linear	$l_d(d) = \frac{T}{2} - d$	$l_r(r) = N_x(T) - r$

might not be the best choice. Therefore a Gaussian of the form $g(z) = ae^{-\frac{1}{2}(\frac{z-\mu}{\sigma})^2}$ is chosen to model this effect, where z is the time difference between the two emails, a is a constant factor, and μ and σ are the mean and variance parameters respectively. Given a time window T centered at the time the mention-email was sent, the mean μ is 0. Since 95% of the area under the Gaussian distribution is within two standard deviations of the mean, the standard deviation σ is set at half of the maximum allowed time difference. One could instead emphasize the order by which messages sent with respect to the email of interest. In such a case, z denotes the rank of the absolute value of the time difference, and σ becomes half of the number of the emails selected by the contextual similarity within T, $N_x(T)$. This approach might be better with bursty email streams in which a time difference of an hour during normal working hours may be more meaningful than a time difference of a day over a weekend. Another, simpler, function that can model the required effect is the linear function of the form $l(z) = a - z$. As before, z can represent either the absolute time difference d, in such case $a = T/2$, or the rank r, in such case $a = N_x(T)$. Table 4.1 summarizes the different tried temporal functions.

4.3.3 Topical Similarity

The goal of expanding the topical context of an email message e is to find other emails that are topically-relevant to e. From the information retrieval perspective, this is a

traditional query-by-example problem that has been well researched in, for example, the TREC routing task [50] and the Topic Detection and Tracking evaluations [4].

Representing an email solely by its content in a bag-of-words similarity model might not yield satisfactory results, since many emails are quite terse. To partially mitigate this effect, the conversational structure of email threads can be exploited to obtain some (hopefully) topically related text. In earlier work in [38], *only* the body of the root of the thread was used as a content representation for every email in that thread. More generally, any subset of the messages between an email and the root of its thread could be used, as illustrated in Figure 4.4.

Three alternatives were tried: the email only (the black node in that figure,) the root of the thread (the highest grey node,) or the entire path from the root to the email (all the black and grey nodes, but none of the white ones.) In each case, any quoted text was automatically deleted. Not doing that would otherwise have the effect of limiting the differences between these alternatives. In each case, BM25S was used for similarity computations [69].

4.3.4 Social Similarity

The social context of an email e aims to include all other emails that involve any of its communicating participants $p(e)$. In prior work, only temporal similarity was used in constructing the social context [31, 38]. However, for the task of resolving mentions, emails that involve many common participants with the mention-email are expected to be more informative (i.e., more likely to be chosen in the generative story) than those

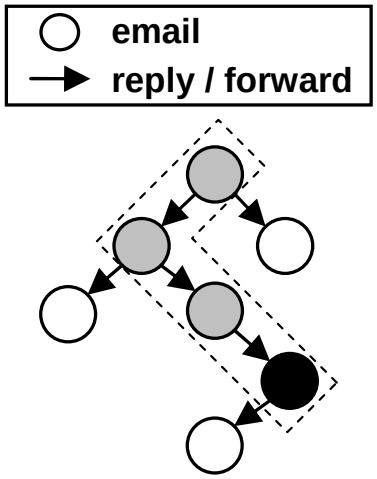


Figure 4.4: Email representation using thread.

Table 4.2: Social similarity functions

$Overlap(e_i, e_j) = \lvert v(e_i) \cap v(e_j) \rvert$
$Jaccard(e_i, e_2) = \frac{\lvert v(e_i) \cap v(e_j) \rvert}{\lvert v(e_i) \cup v(e_j) \rvert}$
$OvJacc(e_i, e_j) = \sqrt{Overlap(e_i, e_j) * Jaccard(e_i, e_j)}$

that involve fewer. One possible social similarity measure for two emails e_i and e_j is the size of overlap between the two participating sets $v(e_i)$ and $v(e_j)$ that each include the email addresses of the sender and the recipient(s). The Jaccard coefficient (union over intersection) is another way of accounting for the overlap, in this case with some normalization. Both measures may have strengths, so their geometric mean would also be taken. Table 4.2 summarizes the three proposed social similarity measures.

4.4 Ranking Candidates Using Context Expansion

Section 4.3 described how to reconstruct the context of an email in its 4 different types. It remains to present how to use the reconstructed context to rank candidate referents of a given mention. Given a specific mention m and the set of identity models C, the

goal in this step is to compute $p(c|m)$ for each candidate c and rank them accordingly.

As described in section 3.5.3.1, a simple solution ignores the context of m and relies solely on the identity models to compute a "context-free" resolution. In that case, $p(c|m)$ is computed by applying Bayes' rule as follows:

$$p(c|m) \approx p(c|l^m) = \frac{p(l^m|c)\ p(c)}{\sum_{c' \in C} p(l^m|c')\ p(c')} \tag{4.1}$$

Alternatively, the rich context of m can be used to help observe evidence for the potential candidates. The mention is expanded with its context to get

$$p(c|m) = p(c|l^m, X(e^m))$$

Bayes' rule is then applied to get

$$p(c|l^m, X(e^m)) = \frac{p(c, l^m, X(e^m))}{p(l^m, X(e^m))} \tag{4.2}$$

where $p(l^m, X(e^m))$ is the probability of observing l^m in the context. This probability can be ignored since it is constant across all candidates in the ranking. The focus now is restricted to the numerator $p(c, l^m, X(e^m))$; that is the probability that the sender chose to refer to c by l^m in the contextual space. As discussed in Section 4.3, X is defined as a mixture of contexts, therefore it can be further expanded as follows:

$$p(c, l^m, X(e^m)) = \sum_k \lambda_k\ p(c, l^m, x_k(e^m)) \tag{4.3}$$

Following the intuitive generative scenario introduced earlier, the context-specific probability can be decomposed as follows:

$$p(c, l^m, x_k(e^m)) = p(c)\ p(x_k(e^m)|c)\ p(l^m|x_k(e^m), c) \tag{4.4}$$

where $p(c)$ is the probability of selecting a candidate c, $p(x_k(e^m)|c)$ is the probability of selecting x_k as an appropriate context to mention c, and $p(l^m|x_k(e^m), c)$ is the probability of choosing to mention c by l^m given that x_k is an appropriate context to mention m.

- ***Choosing person to mention*:** $p(c)$ can be estimated as discussed in Section 3.5.

- ***Choosing appropriate context*:** Applying Bayes' rule to compute $p(x_k(e^m)|c)$ gets

$$p(x_k(e^m)|c) = \frac{p(c|x_k(e^m))\ p(x_k(e^m))}{p(c)} \tag{4.5}$$

  $p(x_k(e^m))$ is the probability of choosing x_k to generally mention people. In conducted experiments, a uniform distribution over all contexts was assumed. $p(c|x_k(e^m))$ is the probability of mentioning c in $x_k(e^m)$. Given that the context is defined as a distribution over emails, this can be expanded to

$$p(c|x_k(e^m)) = \sum_{e_i \in E} p(e_i|x_k(e^m)\ p(c|e_i)) \tag{4.6}$$

  where $p(c|e_i)$ is the probability that c is mentioned in the email e_i. This, in turn, can be estimated using the probability of referring to c by at least one unique reference observed in that email. By assuming that all lexical matches in the same email refer to the same person, and that all lexically-unique references are statistically independent, that probability can be computed as follows:

$$\begin{aligned} p(c|e_i) &= 1 - p(c \text{ is not mentioned in } e_i) \\ &= 1 - \prod_{m' \in M(e_i)} (1 - p(c|m')) \end{aligned} \tag{4.7}$$

  where $p(c|m^{'})$ is the probability that c is the true referent of $m^{'}$. This is the same general problem of resolving mentions, but now concerning a related mention $m^{'}$ found in the context of m.

This shows that resolving one mention depends on the resolution of other mentions in its context, which motivates a dependency-graph-like implementation that is discussed in Chapter 5. It also leads to the discussion of the iterative approach in the following section.

- ***Choosing a name-mention*:** To estimate $p(l^m|x_k(e^m), c)$, it is suggested that the email author would choose either to select a reference (or a modified version of a reference) that was previously mentioned in the context or just ignore the context. Hence, that probability is estimated as follows:

$$p(l^m|x_k(e^m), c) = \alpha\, p(l^m \in x_k(e^m)|c) + (1-\alpha)\, p(l^m|c) \tag{4.8}$$

where $\alpha \in [0, 1]$ is a mixing parameter (set at 0.9 in the consucted experiments), and $p(l^m|c)$ is estimated as in Section 3.5. $p(l^m \in x_k(e^m)|c)$ can be estimated as follows:

$$p(l^m \in x_k(e^m)|c) = \sum_{m^{'} \in x_k(e^m)} p(l^m|l^{m^{'}}) p(l^{m^{'}}|x_k(e^m))\; p(c|l^{m^{'}}) \tag{4.9}$$

where $p(l^m|l^{m^{'}})$ is the probability of modifying $l^{m^{'}}$ into l^m. All possible mentions of c are assumed to be equally similar to m and estimate $p(l^m|l^{m^{'}})$ by $\frac{1}{|\text{possible mentions of } c|}$. $p(l^{m^{'}}|x_k)$ is the probability of observing $l^{m^{'}}$ in x_k, which is estimated by its relative frequency in that context. Finally, $p(c|l^{m^{'}})$ is again a mention resolution problem concerned with the reference r_i which can be resolved as shown earlier.

4.5 Joint Resolution using Iterative Approach

As presented in section 4.4, the structure of the proposed solution for mention resolution requires the resolution of all mentions that appear in the context of a given mention m before resolving that mention. This can be formulated as follows:

$$p(c|m) = f(p(c|m')) \quad \forall m' \in X(e^m) \tag{4.10}$$

where $f(p(c|m'))$ indicates a function of the estimated probability that c is the true referent of a mention m' in the context of m.

There are two alternative solutions that can handle that case:

1. The cycle is broken by computing context-free resolution probabilities for those mentions in the context. Recall that context-free resolutions do not require any information from the context of the mention, thus can be computed without the need of resolving other mentions.

2. Jointly resolve *all* mentions, since every mention requires the resolution of all other mentions in its surrounding context.

A solution that combines both alternatives by resolving the mentions by an iterative approach is proposed as follows:

$$p_n(c|m) = f(p_{n-1}(c|m')) \quad \forall m' \in X(e^m) \tag{4.11}$$

where p_n indicates the computed estimated probability at iteration n.

In each iteration n, the resolution of a mention m is updated using the resolution of other mentions m', computed in the previous iteration $n-1$. The newly updated resolution

will be propagated to other mentions whose contexts contain m in the next iteration $n+1$. Initially, p_0 is set to the probability estimated by the context-free resolution.

This iterative approach requires a scalable implementation of the algorithm that is capable of jointly-resolving *all* mentions in the collection. Chapter 5 discusses the details of that implementation.

4.6 Chapter Summary

This chapter describes the details of the name-mention resolution algorithm. The algorithm is based on a generative model that follows one way of referring to a person in an email message. It initially expands the contextual space surrounding that mention in four types of contexts, then ranks the candidates based on related mentions that are observed in each type of context. The next chapter introduces an efficient implementation of that algorithm that scales well to resolve all mentions in the collection.

Chapter 5

Parallel Solution Using MapReduce

A sequential implementation of the mention resolution approach presented in Chapter 4 might be sufficient to resolve a few name-mentions in an email collection, but some applications require an efficient and scalable solution for this problem. This chapter presents a parallel MapReduce implementation that has two basic goals:

1. Scalable resolution of *all* mentions in the collection.

2. Joint resolution of all mentions by an iterative process that benefits from the structure of the resolution approach.

The tool used to achieve the scalable implementation, MapReduce, is first introduced, followed by an overview and detailed description of the implementation.

5.1 The MapReduce Framework

MapReduce [30] is a distributed programming framework that builds on the observation that many tasks have the same structure: a computation is first applied to a large number of records (e.g., documents) to generate partial results, which are then aggregated in some fashion. Naturally, the per-record computation and aggregation vary by task, but the basic structure remains fixed. Taking inspiration from higher-order functions in functional programming, MapReduce provides an abstraction that involves the programmer defining two functions, a "mapper" and a "reducer," with the following signatures:

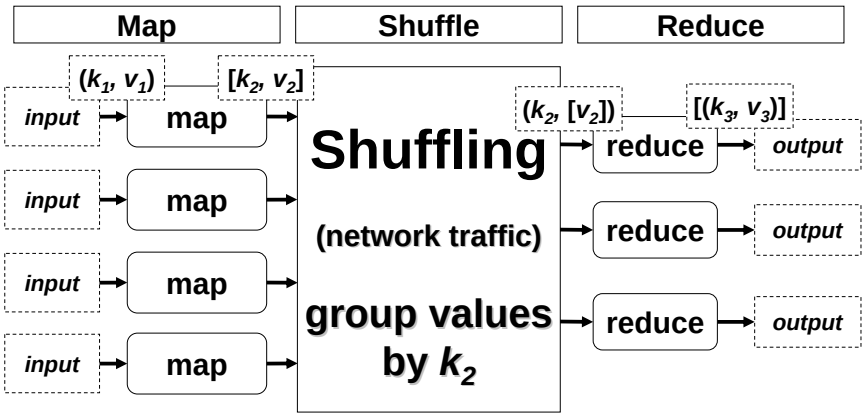


Figure 5.1: Illustration of the MapReduce framework: the "mapper" is applied to all input records, which generates results that are aggregated by the "reducer."

$$\text{map: } (k_1, v_1) \rightarrow [(k_2, v_2)]$$

$$\text{reduce: } (k_2, [v_2]) \rightarrow [(k_3, v_3)]$$

where k's indicate keys, v's indicate values, parentheses indicate single record, and square brackets indicate an array of records. Key/value pairs form the basic data structure in MapReduce. The "mapper" is applied to every input key/value pair to generate an arbitrary number of intermediate key/value pairs. The "reducer" is applied to all intermediate values associated with the same intermediate key to generate output key/value pairs (see Figure 5.1).

Since the mapper is applied to a single input record at a time, independently from any other record, a set of mappers can work simultaneously in parallel on separate processing nodes. Similarly, the reducer is applied to a set of values that are all mapped to a specific key, thus it can also work in parallel with other reducers on separate processing nodes.

On top of a distributed file system, the runtime transparently handles all other aspects of execution (e.g., scheduling and fault tolerance) on clusters ranging from a few to

a few thousand nodes. The runtime is responsible for scheduling map and reduce workers on commodity hardware. The runtime also manages data distribution issues, including splitting the input across multiple mappers and the potentially very large sorting problem between the map and reduce phases whereby intermediate key/value pairs must be grouped by key, a process called "shuffling."

All of these features make MapReduce an attractive distributed programming framework because it shields the programmer from many low-level issues (such as synchronization, data exchange, fault tolerance, and load balancing) which might otherwise hinder a clear focus on the original problem the programmer is trying to solve.

5.2 Implementation Overview of *IdResolver*

The mention resolution approach introduced in Chapter 4 can easily be decomposed into smaller components that fit the map-shuffle-reduce processing pipeline well, or at least can benefit from the transparent distributed processing by using that framework as a parallelization mechanism (using mappers only, without the "shuffle" and "reduce" stages.)

Figure 5.2 illustrates the main components of the resolution system, called *IdResolver* . Each block in the figure is implemented by at least one MapReduce job, with the ones that best fit the framework highlighted using a darker shade. Four major processing steps are required: packing, preprocessing, context expansion, and resolution; all are described in the following sections.

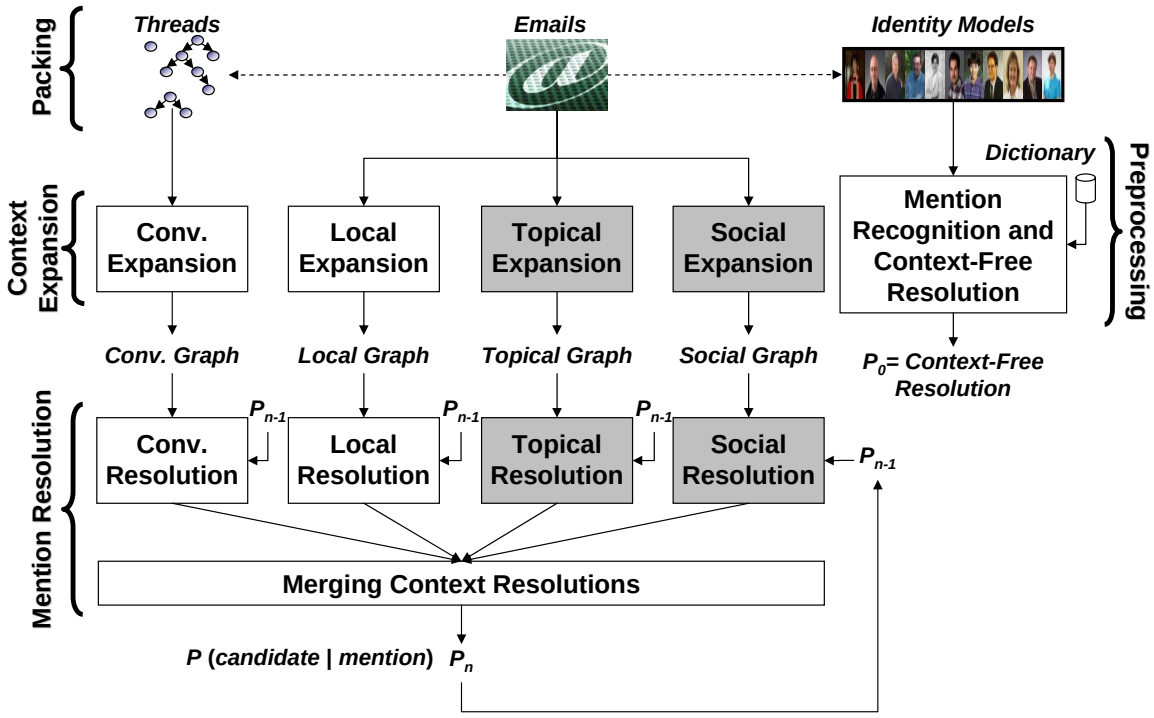


Figure 5.2: Overview of the resolution system design

5.3 Packing

The process starts with the index, threads, and identity models resulting from the identity modeling step described in Chapter 3. This data was packed into sequence files to fit the MapReduce processing pipeline. A sequence file is a sequence of key-value pairs. Emails are keyed by their message ids, identity models by their ids (produced as a sequence number during the process of constructing those models,) and threads by the message id of the root. The structure the email record includes separate fields for the participants, subject, body, the body of the thread root (if different), and the concatenated text of emails in the path to the root; this design facilitates investigation of alternative content representations with a single processing pipeline.

5.4 Preprocessing

In order to resolve all mentions, *IdResolver* must first automatically recognize their existence. For that goal, *IdResolver* adopt a simple name extraction approach that is corpus/dictionary-based. All identified names and nicknames in the identity models are first cleaned to filter out non-names using a merged set of online dictionaries, and then the surviving names are augmented with a list of frequent names provided by the U.S. Census Bureau [19], and with all email addresses in the identity models. The full list is then fed to the Aho-Corasick linear-time algorithm [2] to build a finite state machine that is used to scan text for exact matches.

After detecting a mention m (lexically l^m) in the body of an email, a likelihood probability $p(l^m|c)$ and a context-free posterior probability $p(c|l^m)$ is computed for each identity candidate c, using equations 3.6 and 3.8 respectively. An identity model is considered a potential referent for m if l^m appears as a name in the extracted list of names of that identity.

The two steps are implemented in one combined MapReduce job. Mappers load the Aho-Corasick structure to extract mentions from each email. Each reducer then loads the identity models and computes the context-free posterior probabilities for each candidate of each mention in one email.

5.5 Context Expansion

The goal of the context expansion process is to compute, for each email e_i, the probability $p(e_j|x_k(e_i))$ that an email e_j belongs to the context type x_k of e_i.

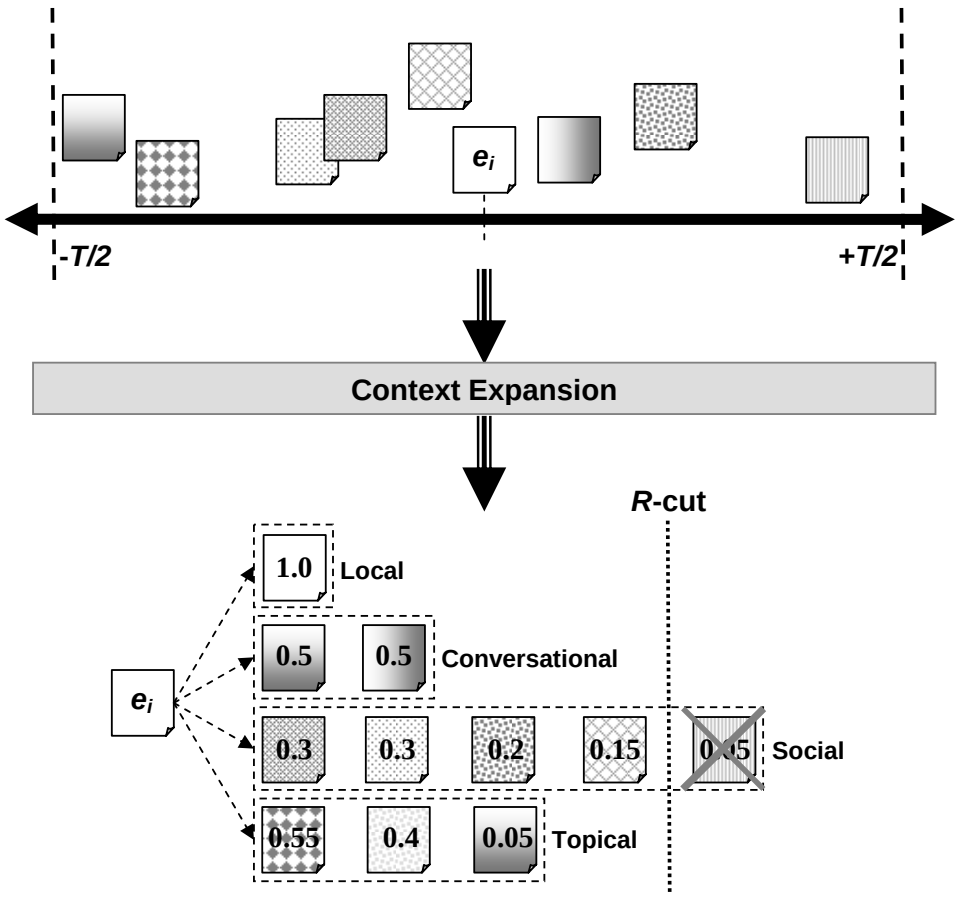


Figure 5.3: Overview of the context expansion process for one email e_i.

Theoretically, such a probability value must be computed for each email e_j in the collection, for each context type k. This is practically reasonable for both local and conversational contexts; in the former, there is only one email (which is the mention-email) in the context, while in the latter, only emails that are in the same thread belong to the context. Expanding either social or topical context, however, requires computing (social or topical) similarity between e_i and every other email e_j; repeating this process for every email in a large collection is computationally challenging.

To make the process tractable, *IdResolver* used two computational expedients:

1. A time period T defining a time window [time(e_i) - $\frac{T}{2}$, time(e_i) + $\frac{T}{2}$] that limits the

context to the emails that were sent within that period only; emails that were sent outside that period are assumed to have zero probability(i.e., do not belong to the context.)

2. A rank cut-off R-cut that indicates the maximum number of emails in the context whose probabilities will be retained for the subsequent processing. For example, if R-cut = 4, then only the emails with the highest 4 (or fewer) probabilities are retained.

Figure 5.3 illustrates the context expansion process for one email e_i using both T and R-*cut*. Notice that the final output of the expansion of one email is one ranked list of pairs for each context; each pair consists of an email id associated with the probability of being in the context. The process can be summarized as follows:

1. Emails are first filtered by the time window [time(e_i) - $\frac{T}{2}$, time(e_i) + $\frac{T}{2}$].

2. A score is computed for each surviving email to indicate its similarity with e_i.

3. The scores are normalized to sum to 1.0, so that they can be used as probabilities. Emails are then sorted according to the similarity score.

4. All emails ranked below R-cut are eliminated from the context representation.

The bottleneck in the whole process is the computation of pairwise similarity. Both social and topical expansion inherently involve computing "similarity" between two emails, but on different representations; for the topical context, the email is represented as a bag of "terms" (that appear in the body or concatenated bodies of some emails in the thread in case the email is topically-represented by its path to the root,) whereas

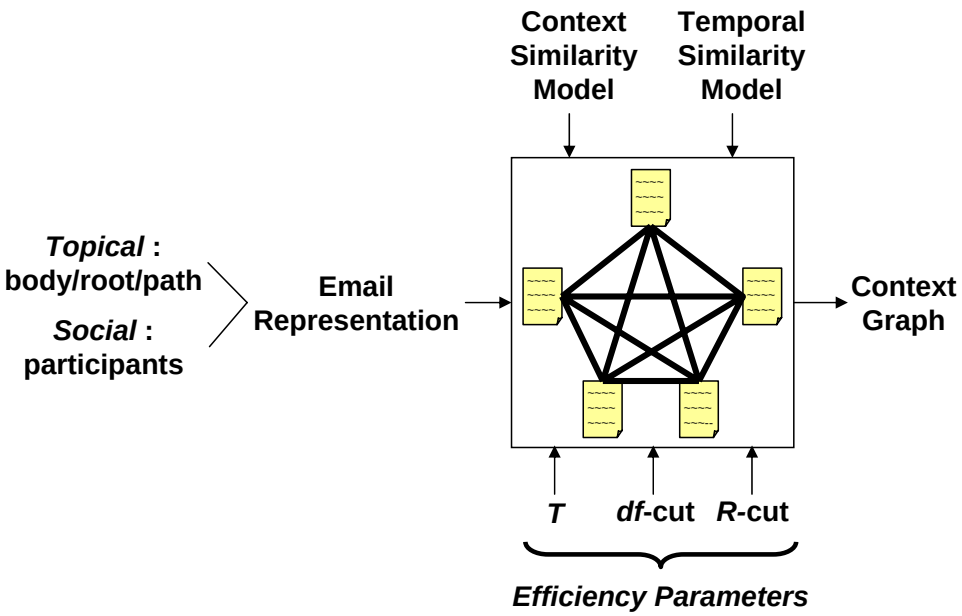


Figure 5.4: Pairwise similarity for context expansion.

each email in the social context is represented as a bag of "email addresses" (considered a special type of "terms" here) of its participants (i.e., sender and recipient(s).) Any bag-of-words similarity model can then be directly applied to compute social or topical expansion.

Figure 5.4 shows the pairwise similarity process as a black box controlled by the email representation, the similarity models and the efficiency parameters. Notice that the figure shows a df-cut parameter that is introduced in Section 5.5.1. By changing the similarity models, the social and topical expansion can be performed using the same mechanism.

Since the computation of pairwise similarity is the heart of the context expansion process, the attention is focused on how to efficiently solve that abstract problem in Section 5.5.1. How *IdResolver* leveraged that solution for context expansion is then discussed in Section 5.5.2.

5.5.1 Pairwise Document Similarity

This section addresses the abstract problem of computing pairwise document similarity in large collections [35]. A "document" here indicates text content (e.g., email message body.) The work described in this section focuses on a large class of document similarity metrics that can be expressed as an inner product of term weights. A document d is represented as a vector W_d of term weights $w_{t,d}$, which indicate the importance of each term t in the document, ignoring the relative ordering of terms (i.e., a "bag of words" model). Only symmetric similarity measures were considered. The measures are defined as follows:

$$sim(d_i, d_j) = \sum_{t \in V} w_{t,d_i} \cdot w_{t,d_j} \tag{5.1}$$

where $sim(d_i, d_j)$ is the similarity between documents d_i and d_j and V is the vocabulary set. In this type of similarity measure, a term will contribute to the similarity between two documents only if it has non-zero weights in both. Therefore, $t \in V$ can be replaced with $t \in d_i \cap d_j$ as follows:

$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} w_{t,d_i} \cdot w_{t,d_j} \tag{5.2}$$

Generalizing this to the problem of computing similarity between *all* pairs of documents, it is noted that a term contributes to *each* pair that contains it.[1] For example, if a term appears in documents x, y, and z, it contributes *only* to the similarity scores between (x, y), (x, z), and (y, z). The list of documents that contain a particular term is exactly what is contained in the postings of an inverted index. Thus, by processing all postings, the entire pairwise similarity matrix can be computed.

[1] Actually, since the focus is on symmetric similarity functions, only half the pairs is computed.

Algorithm 1 Compute Pairwise Similarity Matrix

1: $\forall i, j : sim[i, j] \Leftarrow 0$
2: **for all** $t \in V$ **do**
3: $\quad p_t \Leftarrow postings(t)$
4: $\quad$ **for all** $d_i, d_j \in p_t$ **do**
5: $\quad\quad sim[i, j] \Leftarrow sim[i, j] + w_{t,d_i} \cdot w_{t,d_j}$

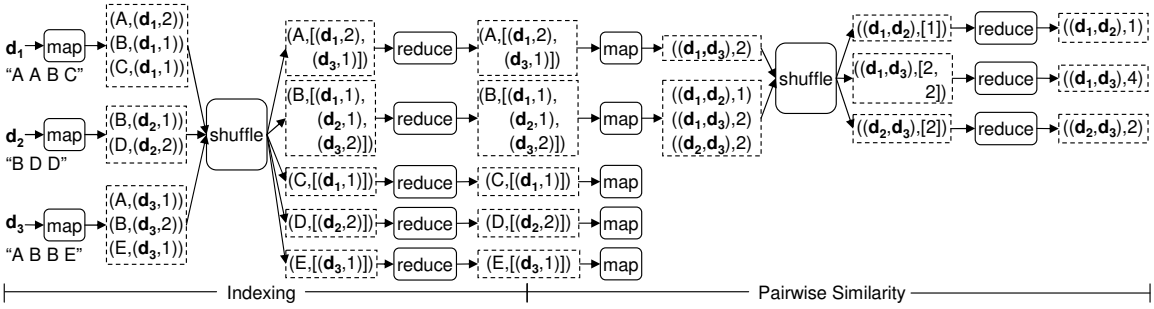


Figure 5.5: Computing pairwise similarity for a toy collection of 3 documents. A simple integer term weighting scheme ($w_{t,d} = tf_{t,d}$) is shown for illustration.

Algorithm 1 formalizes this idea. $postings(t)$ denotes the list of documents that contain term t. For simplicity, term weights are assumed to be also stored in the postings. For small collections, this algorithm can be run efficiently to compute the entire similarity matrix in memory. For larger collections, disk access optimization is needed—which is provided by the MapReduce runtime, without requiring explicit coordination.

Therefore, the proposed efficient solution to the pairwise document similarity problem, is expressed as two separate MapReduce jobs, illustrated in Figure 5.5:

1. **Indexing**: A standard inverted index is built [39], where each term is associated with a list of docid's for documents that contain it and the associated term weight. Mapping over all documents, the mapper, for each term in the document, emits the term as the key, and a tuple consisting of the docid and term weight as the value. The MapReduce runtime automatically handles the grouping of these tuples, which

the reducer then writes out to disk, thus generating the *postings*.

2. **Pairwise Similarity**: Mapping over each posting, the mapper generates key tuples corresponding to pairs of docids in the postings: in total, $\frac{1}{2}m(m-1)$ pairs are generated for a posting, where m is the posting length. These key tuples are associated with the product of the corresponding term weights—they represent the individual term contributions to the final inner product. The MapReduce runtime sorts the tuples and then the reducer sums all the individual score contributions for a pair to generate the final similarity score.

5.5.1.1 Experimental Evaluation

To evaluate the efficiency of the above implementation, a small experiment is conducted using Hadoop version 0.16.0,[2] an open-source Java implementation of MapReduce, running on a cluster with 20 machines (1 master, 19 slaves). Each machine had two single-core processors (running at either 2.4GHz or 2.8GHz), 4GB memory, and 100GB disk.

The symmetric variant of Okapi-BM25 [69] was implemented as the similarity function:

$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} \frac{tf_{t,d_i}}{0.5 + 1.5\frac{l_{d_i}}{l_{avg}} + tf_{t,d_i}} \cdot \frac{tf_{t,d_j}}{0.5 + 1.5\frac{l_{d_j}}{l_{avg}} + tf_{t,d_j}} \cdot \log \frac{N - df_t + 0.5}{df_t + 0.5} \tag{5.3}$$

where $tf_{t,d}$ denotes the frequency of term t in document d, df_t denotes the number of documents in which term t appeared, l_d denotes the length of document d, and l_{avg} denotes

[2] http://hadoop.apache.org/

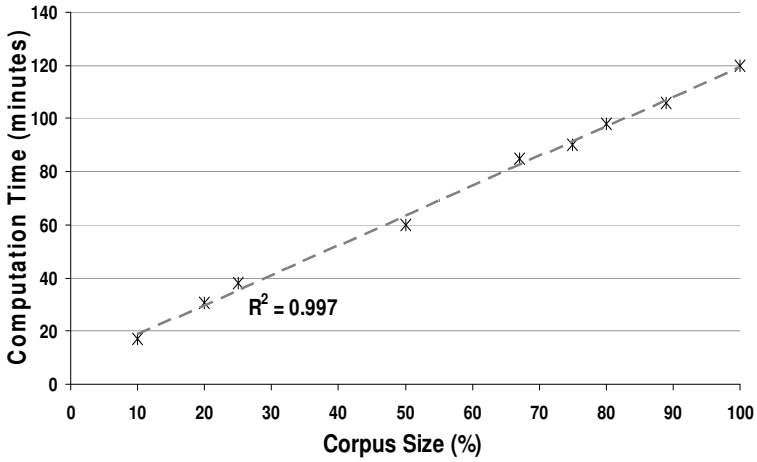


Figure 5.6: Running time of pairwise similarity comparisons, for subsets of AQUAINT-2.

the average length of documents in the collection.

The AQUAINT-2 collection of newswire text was used. It contains about 906,000 documents, totaling approximately 2.5 gigabytes. Terms were stemmed. To test the scalability of that technique, the collection was sampled into subsets of 10, 20, 25, 50, 67, 75, 80, 90, and 100 percent of the documents.

After stopword removal (using Lucene's stopword list), a *df*-cut was implemented, whereby a fraction of the terms with the highest document frequencies are eliminated. This has the effect of removing least-discriminative terms. For this experiment, a 99% cut was adopted, which means that the most frequent 1% of terms were discarded (9,093 terms out of a total vocabulary size of 909,326.) This technique greatly increases the efficiency of the proposed algorithm, since the number of tuples emitted by the mappers in the pairwise similarity phase is dominated by the length of the longest posting (in the worst case, if a term appeared in all documents, it would generate approximately 10^{12} intermediate pairs.)

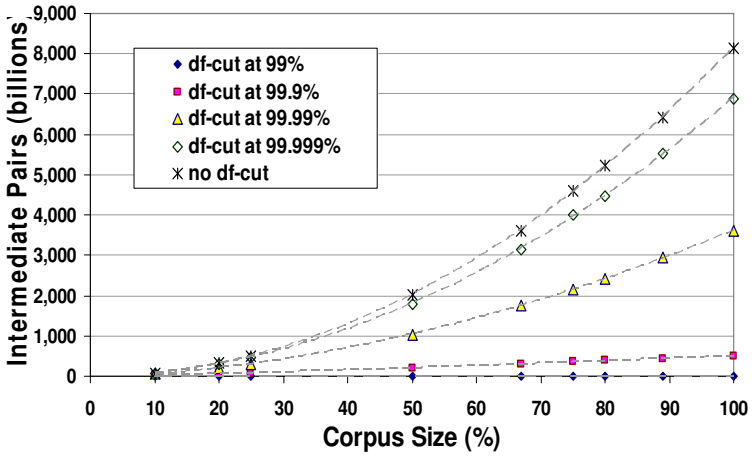


Figure 5.7: Effect of changing *df*-cut thresholds on the number of intermediate document-pairs emitted, for subsets of AQUAINT-2.

Figure 5.6 shows the running time of the pairwise similarity phase for different collection sizes.[3] The computation for the entire collection finishes in approximately two hours. It is empirically found that running time increases linearly with collection size over this region, which is a desirable property. To get a sense for the space complexity, the number of intermediate document pairs, that are emitted by the mappers, were computed. The space savings is large (3.7 billion rather than 8.1 trillion intermediate pairs for the entire collection), and space requirements grow linearly with collection size over this region ($R^2 = 0.9975$).

5.5.1.2 Complexity of the Algorithm

The complexity of the pairwise similarity algorithm is tied to the number of document pairs that are emitted by the mapper, which equals the total number of multiplications required in $O(N^2)$ inner products, where N is the collection size. This is equal

[3]The entire collection was indexed in about 3.5 minutes.

to:

$$\frac{1}{2}\sum_{t \in V} df_t(df_t - 1) \tag{5.4}$$

where df_t is the document frequency, or equivalently the length of the postings for term t. Given the necessity of computing $O(N^2)$ inner products, it may come as a surprise that empirically the algorithm scales linearly (for the explored collection sizes.) The key to this behavior is the *df*-cut technique, which eliminates the head of the *df* distribution. Eliminating the top 1% of terms reduces the number of document pairs by several orders of magnitude.

As Figure 5.7 illustrates, relaxing the *df*-cut to a 99.9% threshold still results in approximately linear growth in the requirement for intermediate storage (at least over this region). Recent experiments suggest that a *df*-cut of 99.9% results in almost no loss of effectiveness on a query-by-example task, compared to no *df*-cut [53]. In essence, optimizing the *df*-cut is an efficiency vs. effectiveness tradeoff that is best made in the context of a specific application; alternative approaches to similar problems based on locality-sensitive hashing [5, 78] face similar tradeoffs in tuning for a particular false positive rate; cf. [11].

5.5.2 Context Expansion using Pairwise Similarity

The 2-step pairwise similarity algorithm presented in Section 5.5.1 represents the core of the context expansion process. However, as illustrated in Figure 5.8, other steps are required:

1. A specific email representation (as a document) has to be chosen, based on the type

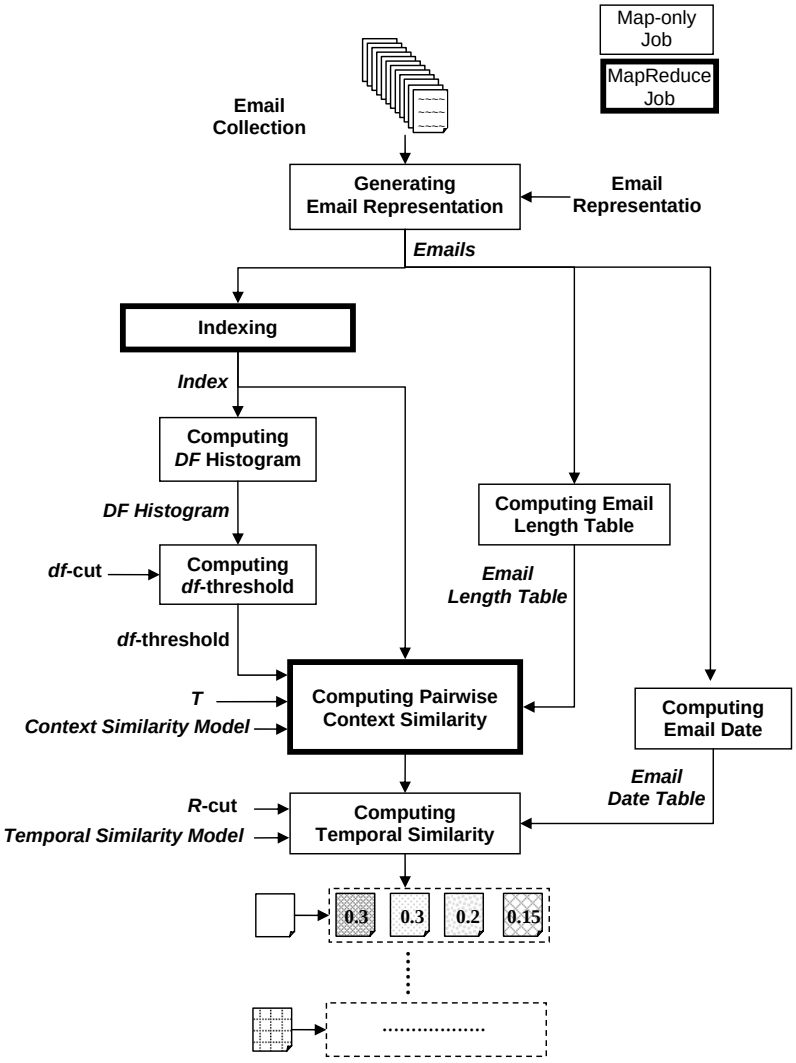


Figure 5.8: MapReduce components of context expansion process.

of the context and the similarity model. For topical context, four representations were tried: subject of the message, body of the message, body of the root of the thread, and concatenated bodies of all messages in the path back to that root. For social context, an email is represented as a textual list of email addresses of sender and recipient(s).

2. The length of each email is computed and stored in a table for lookup by the similarity computations when needed. A similar table is computed for dates of emails.

3. The use of df-cut as a percentage requires the computation of a df histogram that is used to convert the percentage into a df-threshold used in the pairwise similarity job to eliminate terms based on their df values. Notice that this process is only performed for the topical context; since the length of the social representation is already short (it only has the email addresses of the sender and recipients,) it was elected not to use df-cut for social expansion.

4. Generating the partial scores for a pair of emails is conditioned on the pair being sent within a time window T of each other. This limits the number of generated pairs, which in turn reduce the amount of data shuffled across the network. This has been included in pairwise similarity job.

5. A Map-only job (i.e., a MapReduce job that uses no reducers) is used to compute the temporal similarities and thus the final context similarity values. It takes 4 inputs: the output of the pairwise similarity job, a temporal similarity measure, R-cut, and the email date table. This process could equivalently be added at the end of the

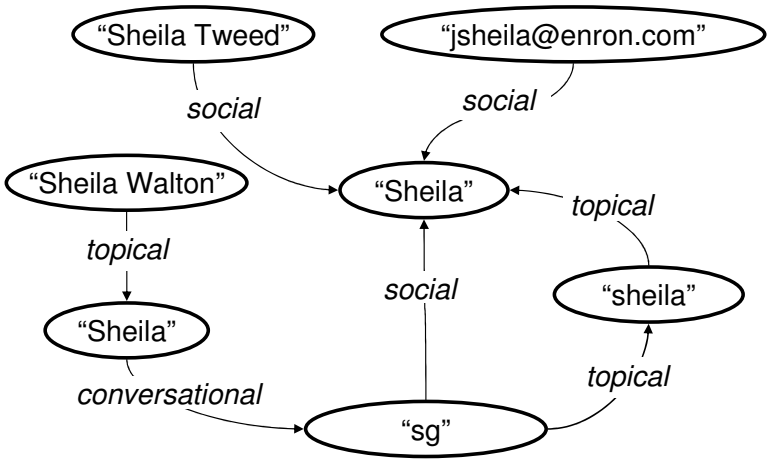


Figure 5.9: An example of a mention graph.

reducers in the pairwise similarity job, but this approach allows the easy use of the same context similarity output with different temporal measures. The output of this job associates each email id with a list of tuples consisting of the email ids in the specified context and normalized weights.

This implementation allows the reuse of the same MapReduce design for both topical and social expansion by changing the document representation, the context similarity measure, and the temporal similarity measure.

5.6 Mention Resolution

The output of the expansion step for one context for all emails can be viewed as a graph, in which nodes are emails and edges connect emails in that context. That output, as shown in Figure 5.3, is actually in adjacency list representation, where each node, representing an email, has a list of edges *from* other nodes, representing emails that appear in its context. In fact, there are 4 different graphs, one per context type, with the local

graph as a special one that has self-edges only.

Since all mentions in one email share the same context, this email-graph can virtually be mapped into a mention-graph, as illustrated in Figure 5.9. Ideally, the resolution of one mention would benefit from first resolving all other mentions in the context (i.e., all other mentions that have edges to it in the mention-graph.)[4]

This can be approximated iteratively. Initially (at iteration 0), each mention is resolved using context-free resolution, therefore each node in the mention-graph (which represents an email with its recognized list of mentions), has a resolution vector (that tells how probable it is that each candidate could be the true referent) for each mention. At the start of iteration 1, each node shares its resolution vectors with its neighboring nodes, so that each can update its current resolution vectors. Those updated resolution vectors then start the next iteration, and so on.

This can easily be implemented in MapReduce. Figure 5.10 shows the three MapReduce jobs needed for the task; the first two are performed only once, initially, whereas the third can be repeated iteratively:

1. Since what is encoded in the context graph is the list of in-edges (i.e, from emails providing context information), not the out-edges (i.e, to emails that need context information for resolving its own mentions), a MapReduce job is required to convert in-edges into out-edges. Each mapper in that job processes one node in the graph (i.e., one email) and emits key/value pairs that associate each email in the ranked list (as the key of the output pair) to the email id of the node along with

[4]Such setup is very similar to the problem of ranking Web pages, of which PageRank [71] is the famous solution.

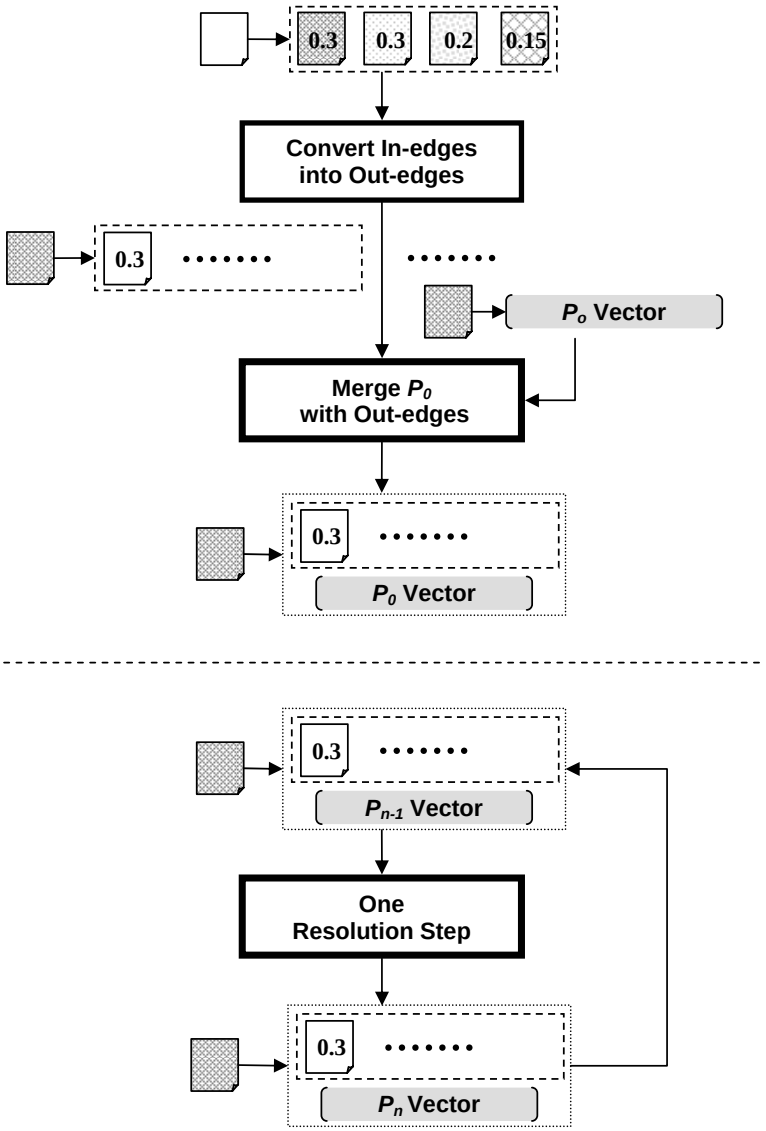


Figure 5.10: Mention resolution process. P_n denotes the resolution vector computed by the end of iteration n.

the context weight of the key (as the value of the output pair;) this indicates an out-edge. Reducers then combine all out edges of the same node.

2. The resolution vectors, initially computed using context-free resolution, are merged with the out-edges list using one MapReduce job. The job has "identity mappers" that take both the out-edges list and the resolution vectors, both keyed by email ids, and passes them all to the shuffler. The Out-edges list and the resolution vectors that are associated with the same email id are combined by the reducer in one structure that is eventually emitted to represent an initialized node in the graph.

3. The MapReduce job that does the actual resolution runs next to conclude the process. At iteration n, each mapper processes one node (email) and emits its current resolution vectors through its out-edges, keyed by the id of the destination node. All information coming to one node is grouped by the shuffling mechanism. Each reducer, processing one node updates its own resolution values (using the resolution approach described in Chapter 4) and writes the updated resolution vectors for the node to disk. The updated resolution vectors can then be fed back as an input to the same MapReduce job for a subsequent iteration.

That process is repeated for each type of context separately. Eventually, the scores are merged through one additional MapReduce job that is similar to the merging step described above.

One big graph could equivalently have all different types of context, and then one resolution step is needed for all contexts combined, but the present design was favored because it is simpler, it reduces the maximum memory and disk space needed in any one

MapReduce job, and it allows easier experimental evaluation of individual contexts. For a system with enough processing resources, it might be more efficient to merge graphs and do one consolidated resolution iteration.

5.7 Chapter Summary

The MapReduce implementation described in this chapter can be used to resolve all recognized mentions in the email collection. The implementation makes use of the inherent characteristics of the MapReduce framework to represent the mention graph and distribute the resolutions across different types of contexts. In doing that, an efficient algorithm for computing pairwise document similarity is developed to help reconstruct the contextual space of all emails simultaneously. Chapter 6 discusses how the results of that implementation can be evaluated.

Chapter 6

New Test Collection for Mention Resolution

Prior work in [31, 63, 38] has been evaluated on test collections that focused on resolution of only some types of references (in general, references for which substantial evidence was available;) for example, all of the queries in the collections described in [38] are selected from emails sent from Enron email domain to at least one recipient in the same domain, and all resolved to addresses in the same domain as well. This chapter introduces a new test collection in which random sampling is used to characterize accuracy across the full range of references to individuals that can be found in the Enron collection. The study described in this chapter is conducted not only to produce, to the best of the author's knowledge, the largest and least-biased test collection for the task, but also to provide some insights on the feasibility of the task in terms of how hard and how time-consuming it is when humans perform it, and reliability of their performance.

As a task model, it is assumed that the user wishes to resolve all mentions of single-token names (which are the ones that are expected to be most ambiguous) in some specific email. A three-step process is designed for building the new collection:

1. Selecting the mention-queries from the whole collection.

2. Manually resolving the selected queries.

3. Measuring inter-annotator agreement on the primary resolution of a subset of the queries.

The following sections describe each of these steps in detail.

6.1 Query Selection

As a preprocessing step, duplicate emails (which amount to about half the collection, as described in section 3.3.1) are automatically removed. Quoted text is also automatically removed if that same text appeared elsewhere in the collection (e.g., as the original quoted message).

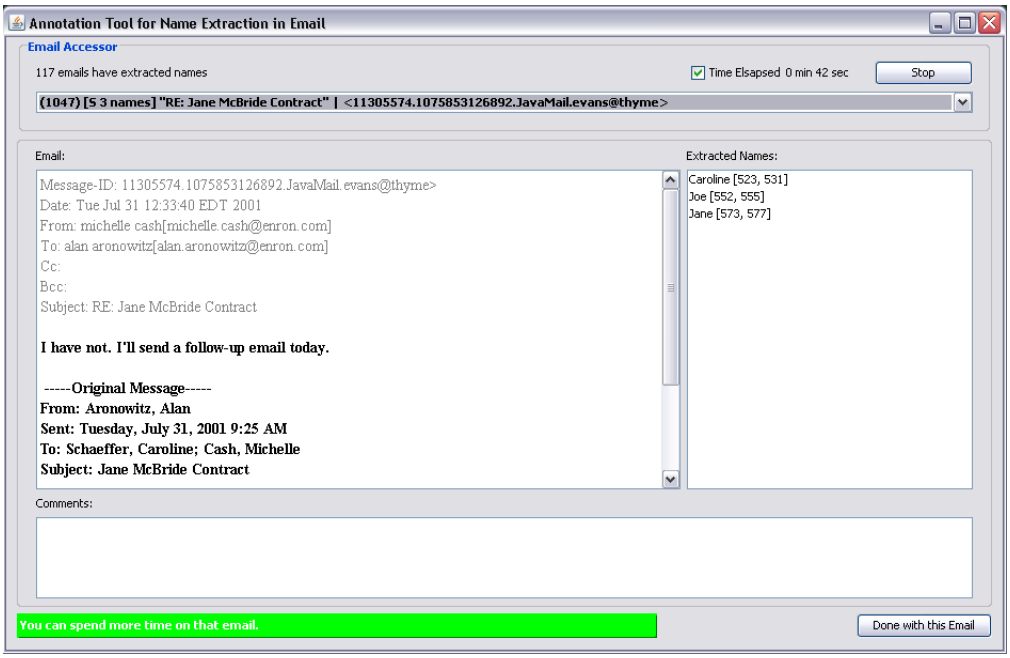


Figure 6.1: Name extraction tool used for mention-query selection.

1,500 emails were then randomly selected and an annotator was recruited to annotate all name mentions in those emails. Figure 6.1 shows a screen-shot of the simple Java tool developed to help extract names from this list of emails; the user can select an email, highlight names to be extracted and optionally write comments about the extracted list. Using that tool, the annotator removed all messages that in their opinion contained no

single-token references to people and all emails that could not reasonably be characterized as communication between people (e.g., automatically generated replies, spam, news articles, weather reports, and forwarded jokes.) In the remaining 346 emails, the annotator identified all single-token person names in the body text of the email (but disregarding salutations and signatures.) This resulted in 781 single-token names extracted from 346 emails. After removing duplicate names within the same email, 584 single-token names remained, about 1.7 names per retained email. Those names indicate the mention-queries.

6.2 Manual Resolution

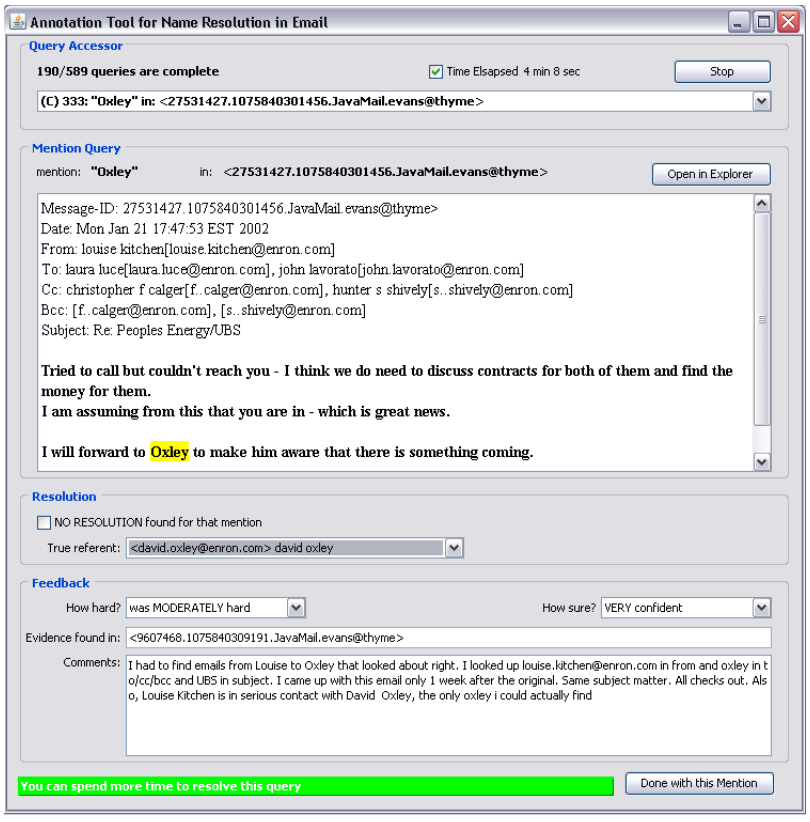


Figure 6.2: Resolution annotation tool.

Three new annotators were recruited to create ground truth resolutions for these 584

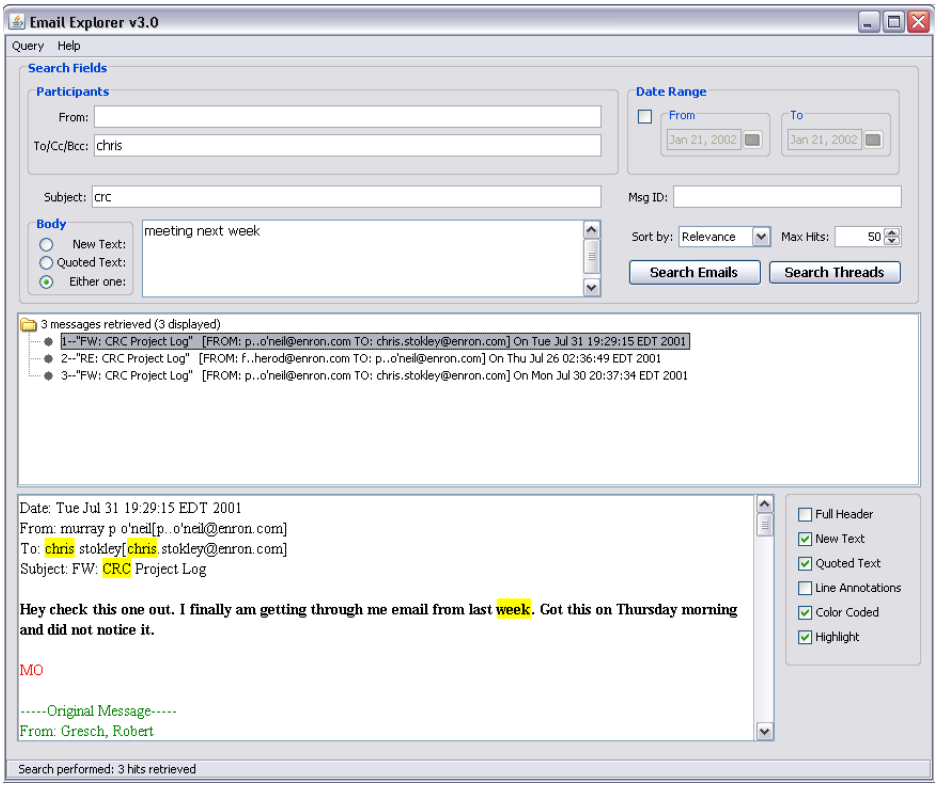


Figure 6.3: Enron search interface.

mention-queries; each was assigned about one-third of the mention-queries. An annotation interface (illustrated in Figure 6.2) that lists the queries and records the resolution decisions taken by the annotator was developed. Alerts were displayed at 6 minutes and at 10 minutes to encourage the annotator not to spend too much time on any one mention-query. The annotator could flag any mention-query as "unresolvable" if they could not resolve it (e.g., if there was not enough evidence available, or if no email address was found in the collection for the true referent.) In such cases, the annotator was asked to guess whether the true referent was an Enron employee. Annotators were also asked to record a confidence value for each resolution, and to indicate how difficult the resolution process had been; each was recorded on a 3-point scale. For a resolution decision to be

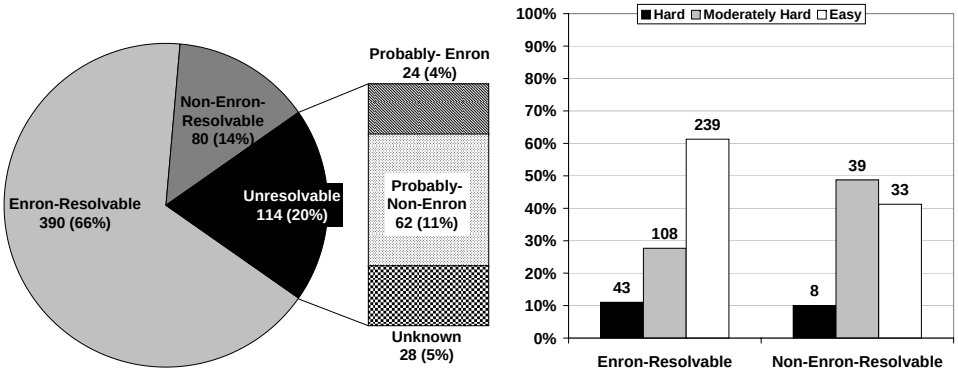
valid, the annotator was also required to identify specific email messages that support their decision, and to write a brief comment explaining their reasoning.

To facilitate the manual resolution task, the annotation tool was linked to a search interface, shown in Figure 6.3, that allows the user to pose queries to search the whole collection using different email fields (e.g., from/to, subject, date, main body, etc.). The search results are presented as a ranked list of messages, and the user can select any individual message for display. The system memorizes the search history, so the user can go back to any previously issued query from the same session.

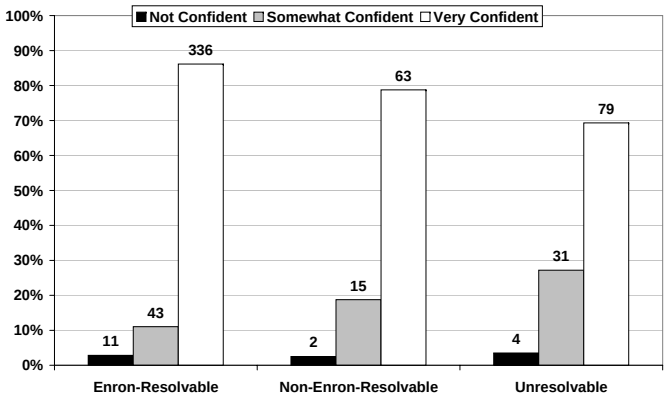
6.3 Results of Primary Annotations

Coverage: As Figure 6.4(a) shows, the annotators were able to resolve about 80% of the mention-queries. The proposed automated resolution techniques always attempt a resolution; these results suggest that is reasonable in about 80% of the cases (for this collection). Most of the resolvable mention-queries were resolved by the annotators to people from Enron, and nearly half of the unresolvable mentions were estimated by the annotators to also refer to people at Enron. This suggests that the existing collections used earlier in [31] and [38] may be somewhat more representative than first suspected.

Confidence and Difficulty: Figure 6.4(c) shows how the confidence of the annotators differed by mention-query type; annotators had somewhat more confidence when resolving queries to people inside Enron than outside Enron. Overall, annotators recorded that they were "very confident" in their decisions in more than 80% of the cases. Reported task difficulty exhibited a similar pattern; as Figure 6.4(b) illustrates, a considerably larger



(a) Coverage of true referents (b) Difficulty of resolution



(c) Confidence of resolution

Figure 6.4: Characteristics of the new test collection

fraction of non-Enron resolutions were reported to be at least moderately difficult.

Effort: Table 6.1 shows that on average annotators spent more time on unresolvable mentions than on those that they could resolve. The automated "nag alerts" at 6 and 10 minutes do not seem likely to have affected many cases, since on average even the unresolvable mentions were completed in about five minutes. This is also supported by the histograms shown in Figure 6.5. The figure shows that the alerts generally had a slight effect on the finishing time.

Table 6.1: Annotation statistics; new test collection.

Number of mention queries	584
Average time spent/query	4 min 10 sec
Average time spent/Enron-resolvable query	3 min 52 sec
Average time spent/non-Enron-resolvable query	4 min 19 sec
Average time spent/resolvable query	3 min 56 sec
Average time spent/unresolvable query	5 min 6 sec
Max time spent for a query	12 min 9 sec

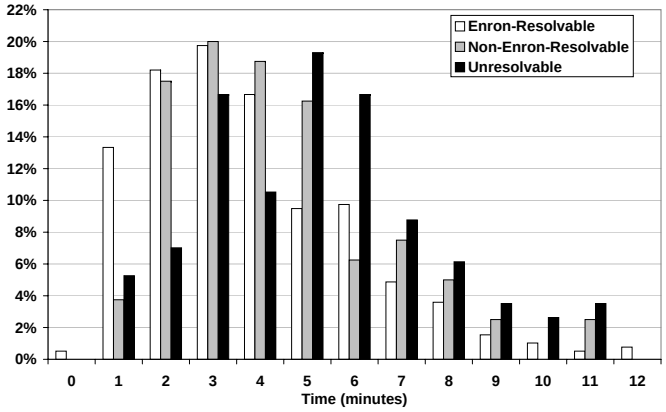


Figure 6.5: Time spent on the queries.

6.4 Collection Availability

The collection was made available for research community by posting it at: http://www.cs.umd.edu/ telsayed/collections/elsayed-mention-resolution-enron.zip.

Both mention-queries and their answers given by the annotators were provided in TREC-style queries and qrels format [68]. Only resolvable mentions were included in the available collection.

6.5 Measuring Inter-Annotator Agreement

At that point, each mention-query is handled by one annotator, but not all of the mention-queries were handled by the same annotator. In general, the resolution decisions

Table 6.2: Pairwise inter-annotator agreement.

Primary	Queries		Secondary	
	Type	Number	a_2	a_3
a_1	Enron	132	12/16	16/16
	Non Enron	27	1/5	2/4
	Unresolvable	36	2/2	4/7
	Total	195	15/23	22/27
a_2	Enron	135	-	35/38
	Non Enron	18	-	2/2
	Unresolvable	37	-	6/10
	Total	190	-	43/50
a_3	Enron	123	24/27	-
	Non Enron	35	5/12	-
	Unresolvable	41	11/11	-
	Total	199	40/50	-

taken by different annotators on the same query might vary, based on how accurate the annotator was and the search technique the annotator used to find clues that support the decision. Measuring how reliable the first-pass resolution process was is therefore necessary before using the test collection in evaluating automatic systems. This can be achieved by conducting an inter-annotator agreement study to estimate the accuracy of the annotations. In this kind of study, a mention-query that was already annotated by a primary annotator is double-annotated by a different secondary annotator. The agreement of two annotators on a the resolution of a mention-query is declared in one of two cases: both annotators resolved it to the same email address, or both have flagged it as unresolvable. Any other combination of annotations is considered disagreement.

Among the three primary annotators, a_1, a_2, and a_3, who participated in the first resolution pass, only two, a_2, and a_3, participated in the inter-annotator study. 150 queries,

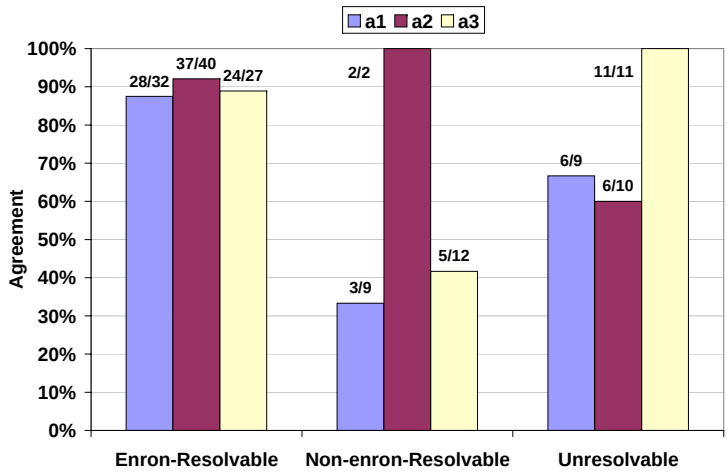


Figure 6.6: Inter-annotator agreement with annotators.

about 25% of whole set, were selected for double-assessment; 50 queries were randomly sampled from the query set of each annotator, and then each sampled query was randomly assigned to a secondary annotator who was different from the primary one, so that each is guaranteed to be double-assessed by two different annotators.

6.6 Results of Secondary Annotations

Agreement of Individual Annotators: Table 6.2 shows how the queries were assigned to each annotator. The ratio x/y in row a_i and column a_j indicates that y queries annotated primarily by a_i were attempted by a secondary annotator a_j who only agreed on x queries of them. For example, 23 queries out of the 195 queries that were primarily annotated by a_1, were double-annotated by a_2; both annotators agreed only on 15 of them. Numbers in the third column indicate the total number of queries annotated by the primary annotator. Figure 6.2 also shows the inter-annotator agreement on the three different categories of queries based on the resolution of the primary annotator: Enron-resolvable,

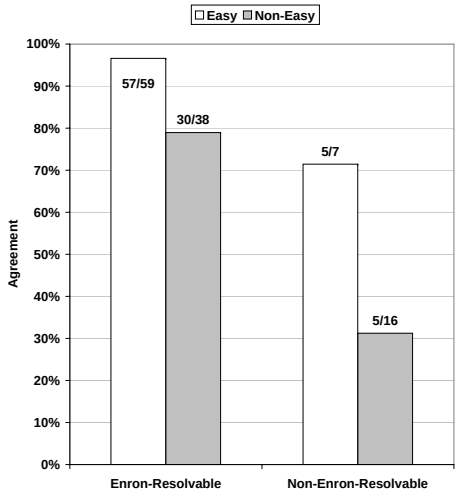


Figure 6.7: Inter-annotator agreement based on self-reported difficulty.

non-Enron-resolvable, and unresolvable. It is evident that agreement on Enron-resolvable mention-queries were higher than that of non-Enron-resolvable on all arrows but one that only has 2 non-Enron-resolvable queries. The agreement on unresolvable queries was generally also higher than that of non-Enron queries, while the agreement on non-Enron-resolvable queries didn't exceed 50% in three of the four arrows, which indicates how difficult it was to resolve such queries. This is also shown by Figure 6.6 which illustrates the overall agreement for each primary annotator, regardless of who double-annotated the primary annotator mention-queries.

Difficulty and Confidence: Inter-annotator agreement based on the difficulty and confidence indicated by the primary annotator are illustrated in Figures 6.7 and 6.8. Mention-queries that the primary annotators annotated as easy are distinguished from any other query that the annotator thought it is not (i.e., not easy, including both moderately-hard and hard ones); similar distinction was applied to confidence levels between very-confident and less-confident levels. As expected, Figure 6.7 shows that agreement on

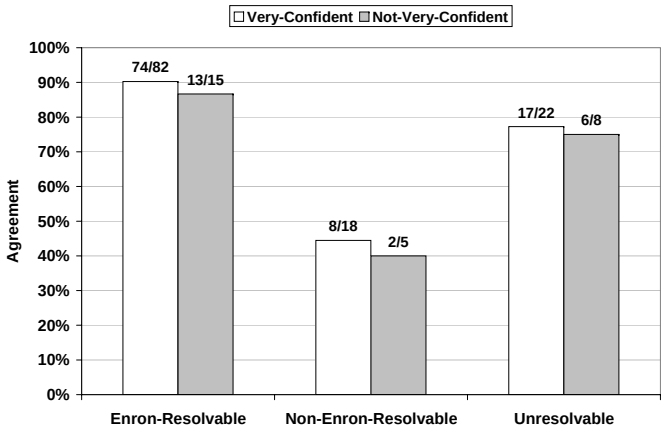


Figure 6.8: Inter-annotator agreement based on confidence.

easy queries is higher than that of not-easy ones. Moreover, the decrease in the non-Enron-resolvable queries is noticeably greater. Figure 6.8 shows similar pattern but with smaller decrease of agreement, probably due to inaccurate reported confidence levels.

Overall Agreement: Figure 6.9 shows the overall agreement on the three query categories. The inter-annotator study resulted in 90% agreement on Enron-resolvable queries when 97 queries were sampled, which is quite high considering that the average ambiguity level of the sampled queries is 194 candidates (in a range from 0 to 1512). A lower level of agreement (77%) was achieved on unresolvable ones, which indicates that secondary annotators were able to resolve about 23% of the sampled queries that were originally unresolved by the primary annotators. Although, the agreement on these two categories indicates that the resolution decision on those queries were quite reliable, the figure shows substantial drop in agreement in case of non-enron-resolvable; the annotators only agreed on about 45% of the type of queries, which means that they disagreed in more than half of the sample (13 out of 23 queries in particular). This low level of agreement suggested that extending the sample by double-annotating more queries of this category

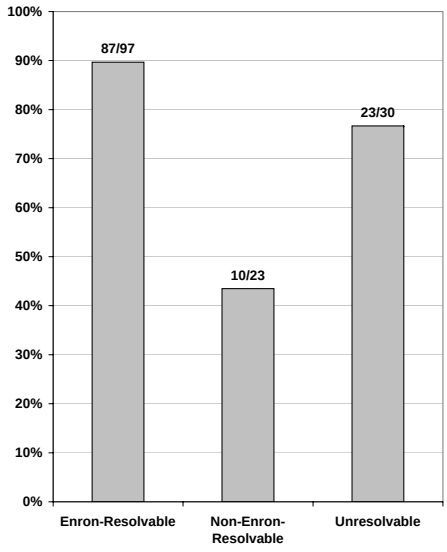


Figure 6.9: Overall inter-annotator agreement.

was needed to obtain a more accurate estimate of agreement. Therefore, the double-annotation of non-enron-resolvable queries was extended to 71 queries out of 80 available from the first pass. It was originally planned to double-annotate all of them, but that was not feasible due to lack of annotators at different points in time. The result illustrated in Figure 6.10 shows a 62% agreement in the larger sample.

6.7 Chapter Summary

This chapter describes the development of a new test collection for the task of mention resolution that is more balanced and larger than previously-used test collections. The basic characteristics of the collection are presented and an inter-annotator study was conducted to measure human agreement on the resolution decisions. The mention resolution approach can be tested using this collection as described in chapter 7.

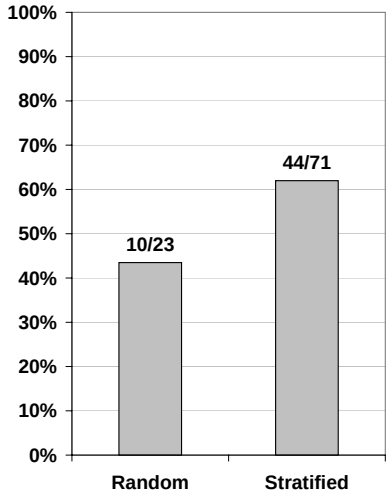


Figure 6.10: Agreement on stratified sample of non-enron-resolvable queries.

Chapter 7

Experimental Evaluation of Mention Resolution

Chapter 1 raised the question of how to evaluate the mention resolution system in a repeatable yet affordable way. The new test collection, developed and presented in Chapter 6, was the first half of the answer that was concerned with the test collection, whereas the second half of the answer, which is the main goal of this chapter, is to explain how this test collection can be used to evaluate the system experimentally.

The chapter starts with a description of the experimental setup in Section 7.1, followed by the system training process in Section 7.2. Finally, the results produced by *IdResolver* are presented and discussed in Section 7.3.

7.1 Experimental Setup

For evaluating *IdResolver* , the following should be specified:

- The mention-queries and email collection used to test *IdResolver*
- The measures used to evaluate the results
- The training/testing strategy adopted for the evaluation
- The hardware configuration in which *IdResolver* implemented and tested

The following sections discuss each of these.

7.1.1 Test Collections

Eight test collections were used to evaluate the mention resolution approach; all are based on the CMU version of the Enron collection; each was created by selecting a subset of that collection, selecting a set of query-mentions within email messages from that subset, and creating an answer key in which each query-mention is associated with a single email address.

Table 7.1 lists all of these test collections with a brief description of each. The first four collections were previously used in experiments by other researchers in [63] and [31]. Preliminary results were also reported on these four collections using two earlier techniques with non-scalable sequential-processing implementations in [36] and [38]. Notice that the name of each collection starts with the last-initial of the researcher who created them.

The first two test collections were created by Minkov et al [63]. These test collections correspond to two email accounts in CMU-Enron email collection: "sager-e" (the "M-Sager" collection) and "shapiro-r" (the "M-Shapiro" collection). Their mention-queries and answer keys were generated automatically by identifying name mentions that probably correspond uniquely to individuals referenced in the cc header, and eliminating that cc entry from the header.

Namata et al. [38] created the third test collection, "N-Subset," which is a larger version of the test collection originally created by Namata's collaborators, Diehl at al. [31]. Emails from all top-level folders were included in the collection, but only those that were both sent by and received by at least one email address of the form <name1>.<name2>@enron.com

were retained. A set of 78 mention-queries were manually selected and manually associated with the email address of the true referent by the third author using an interactive search system developed specifically to support that task. The set of queries was limited to those that resolve to an email address of the form <name1>.<name2>@enron.com. Names found in salutation or signature lines or that exactly match <name1> or <name2> for any of the email participants (indicated in "from", "to", "cc", or "bcc" header fields) were not selected as query-mentions. Those 78 queries include the 54 used by Diehl et al.

For the fourth test collection, "N-Extended", the same 78 mention-queries and the answer key from the N-Subset collection were used, but now all emails from the full CMU version of the Enron collection (with duplicates removed) were included.

The fifth collection, "E-All," is the one described in Chapter 6. The remaining three are subsets of E-All, with all the subsequent ones being subsets of it based on the annotations of the primary annotators.

Some descriptive statistics for each test collection are shown in Table 7.2. The M-Sager and M-Shapiro collections are typical of personal collections, while the others represent organizational collections. These two types of collections differ markedly in the number of known identities and the candidate list sizes as shown in the table (the candidate list size is presented as an average over that collection's mention-queries, as a median, and as the range of values.)[1]

[1]The maximum cardinality 1512 appears for disjoint mention-query sets because more than one 'John" appears as a mention-query (in different messages.) The zeros correspond to names that were not recognized by *IdResolver* because they were not included in any identity model, and thus generate no candidates.

Table 7.1: Test collections used in the experiments.

Test Collection (*Created by*)	Email Messages	Queries
M-Sager (*Minkov et al.* [63])	One directory of CMU-Enron	Automatically created by removing CCed referent
M-Shapiro (*Minkov et al.* [63])	One directory of CMU-Enron	Automatically created by removing CCed referent
N-Subset (*Namata et al.* [38])	Only emails between @enron addresses in CMU-Enron	Manually resolved mentions, all to Enron domain
N-Extended (*Namata et al.* [38])	Whole CMU-Enron	Manually resolved mentions, all to Enron domain
E-All (*Elsayed et al.*)	Whole CMU-Enron	Randomly-selected manually-resolved single-token mentions
E-Enron (*Elsayed et al.*)	Whole CMU-Enron	Subset of E-All resolved to @enron addresses
E-NonEnron (*Elsayed et al.*)	Whole CMU-Enron	Subset of E-all *not* resolved to @enron addresses
E-Hard (*Elsayed et al.*)	Whole CMU-Enron	Subset of E-all labeled as "hard" by the primary annotator.

Table 7.2: Descriptive statistics for the rest collections used in the experiments.

Test Collection	Emails	Addresses	Queries	Candidates			
				Average	Median	Min	Max
M-Sager	1,628	627	51	3	2	1	10
M-Shapiro	974	855	49	5	4	1	16
N-Subset	54,018	27,340	78	131	91	1	441
N-Extended	248,451	123,783	78	454	338	3	1512
E-All	248,451	123,783	470	241	116	0	1512
E-Enron	248,451	123,783	390	246	121	0	1512
E-NonEnron	248,451	123,783	90	213	66	1	1512
E-Hard	248,451	123,783	51	267	150	0	1512

7.1.2 Evaluation Measures

There are two commonly used single-valued evaluation measures for "single-answer"-retrieval tasks. The "*Success @ One*" (S@1) measure characterizes the accuracy of one-best selection, computed as the mean across queries of the precision at the top rank for each query. For a single-valued figure of merit that considers every list position, "*Mean Reciprocal Rank*" (MRR), computed as the mean across queries of the inverse of the rank at which the true referent is found, is commonly reported.

7.1.3 Training/Testing Plan

The adopted evaluation strategy is standard: train the system (i.e, tune its parameters) using one collection, and test the trained system on another collection. E-All and all of its subsets are not used in the training process, so that *IdResolver* can be evaluated on the best available collection. The E-Extended collection was chosen as the training collection, since it is the closest collection to E-All in terms of the number of email messages and the query set. The trained system was then evaluated on the other test collections. Two caveats on that choice are worth mentioning here:

1. There is a big difference in scale between the training collection and the two small collections specially, M-Sager and M-Shapiro.

2. All of the mention queries in the training collection are focused on Enron employees, for which there is likely to be a substantial amount of evidence in Enron email. This is surely *not* the only type of queries in E-All.

7.1.4 Hardware Setup

Experiments were run on a large cluster of approximately 480 machines provided by Google and managed by IBM, shared among a few universities [55]. Each machine has two single-core processors (2.8 GHz,) 4 GB memory, and two 400 GB hard drives. The entire software stack (down to the operating system) is virtualized; each physical machine runs one virtual machine hosting Linux. Experiments used Java 1.5 and Hadoop version 0.17.2, an open-source Java implementation of MapReduce. Experiments reported in this chapter were limited to 200 processing nodes (cores.)

7.2 Tuning Parameters

Table 7.3 lists the parameters that were tuned for the social and topical contexts. Local and conversational context do not have specific parameters. In this section, the tuning process for both the social and topical context is described.

Table 7.3: Parameters for the social and topical contexts.

Parameter	Social Context	Topical Context
Context Similarity	Jaccard/Overalp/OvJacc	Subject/Root/Body/Path
Temporal Similarity	g_d, g_r, l_d, l_r	g_d, g_r, l_d, l_r
Context Time Period (T)	> 0	> 0
Rank Cut-off (R-cut)	> 0	> 0
DF Cut-off (df-cut)	-	$\geq 99\%$

7.2.0.1 Social Context

R-cut was first fixed at 250 while changing T to one of 3 values: 10, 100, and 200 days; these values were adopted in previous research in [31] and [38]. Both the social and

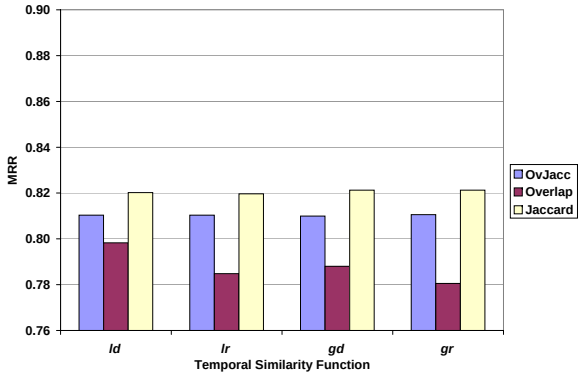
temporal similarity models were then jointly changed over all possible values. Figure 7.1 shows MRR results for these experiments. Figure 7.1(a) indicates that MRR is not very sensitive to the choice of the temporal similarity at T=10. Figures 7.1(b) and 7.1(c) show that MRR improves for some temporal similarity measures with wider time windows, and that the combination <OvJacc, l_d > is a good choice over both periods.

To further refine values for T, possible values were stepped through in a range from 50 to 400 days with a step size of 50 days, as shown in Figure 7.2. A sharp peak discovered at 100 days motivated a more fine grained search that eventually found the peak MRR value at 138 days.

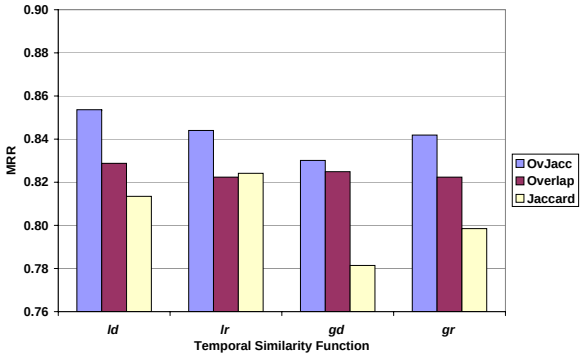
With the similarity models and time period T tuned, the only remaining parameter of the social context to be tuned is R-cut, the size (in messages) of the context of a single message. The values of R-cut were tried in a range from 50 to 500 emails, as illustrated in Figure 7.3, to find a peak value between 250 and 300. A value of 250 messages was then chosen for efficiency reasons.

7.2.0.2 Topical Context

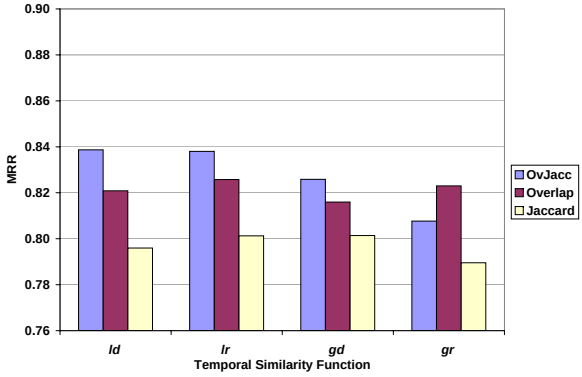
A similar tuning process is repeated for the parameters of the topical context. The *df*-cut value was initially fixed at 99.9% and BM25S [69] was used as the term weighting function for computing topical similarity of pairs of emails. Unlike the social context, MRR continues to improve slightly as T increases to 200, with the *Path* representation clearly superior at the highest value of T. In this case, g_d seems to be a good choice for measuring temporal similarity, as illustrated in Figure 7.2.0.2.



(a) T=10



(b) T=100



(c) T=200

Figure 7.1: Effect of changing *social* and temporal similarity with different time periods (N-Extended collection, R-cut=250).

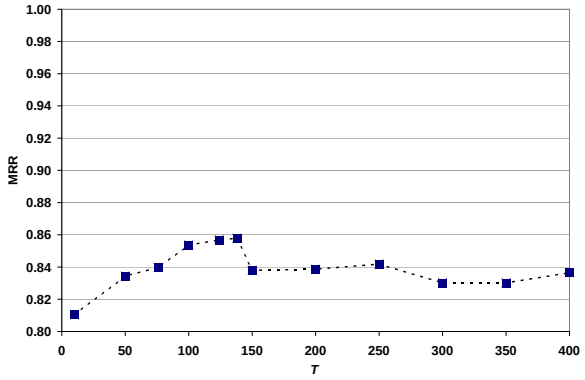


Figure 7.2: Effect of changing *social* time period T (N-Extended collection, Sim = [Ov-Jacc, l_d], R-cut=250).

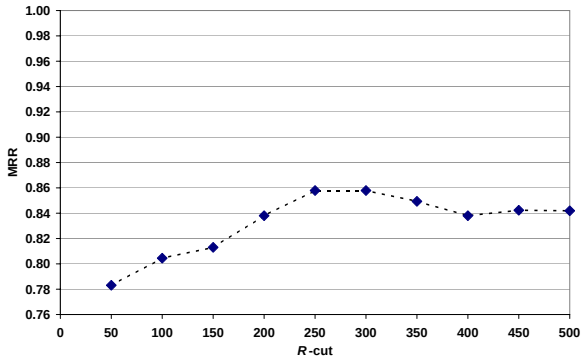


Figure 7.3: Effect of changing *social* R-cut (N-Extended collection, Sim = [OvJacc, l_d], T=138).

Table 7.4: Tuned parameter values of social and topical contexts.

Parameter	Social Context	Topical Context
Context Similarity	OvJacc	Path
Temporal Similarity	l_d	g_d
Context Time Period T	138	250
Rank Cut-off R-cut	250	200
df-cut	-	99.9%

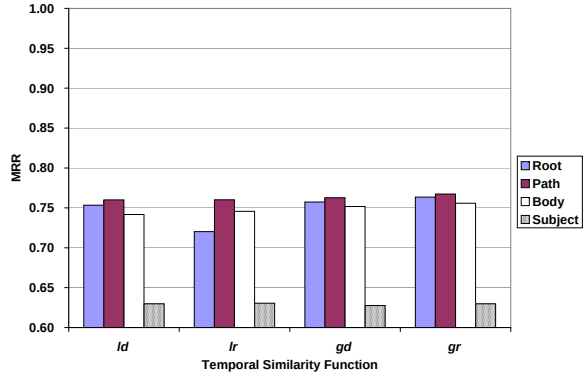
It is noticed that the "Path" representation, which represents the content of an email by a concatenated text from all emails that belong to its path to the root of the thread, was substantially better than the other alternatives; it captures more content about the original email, which is important since many emails are often short.

Peak values of T and R-cut were 250 days and 200 messages, as shown in Figures 7.5 and Figure 7.6 respectively. Finally, the tuned system was tested on different values of df-cut at 99%, 99.5%, 99.95%, and 99.99%. Figure 7.7 shows precision improvement with higher values up to 99.9%, followed by an unexpected drop at 99.99%. That drop (which happened by removing smaller number of highest frequent terms) indicates that tuning the term weighting function might be a good idea for future work.

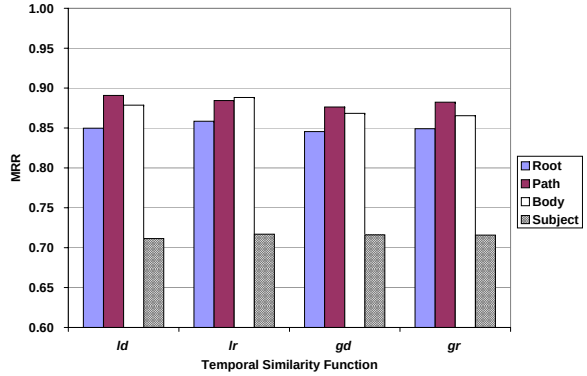
The final tuned parameter values for both social and topical contexts are listed in Table 7.4. The MRR values achieved of the tuned system were 0.858 and 0.927 for the social and for the topical contexts respectively.

7.3 Evaluation

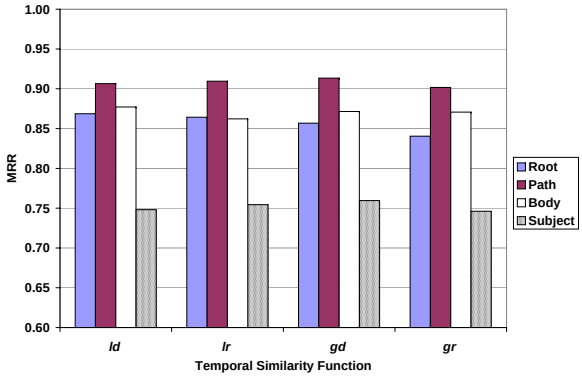
The trained system obtained in section 7.2 were then tested on all of the collections listed in Table 7.1. All of the possible combinations of contexts were tried, with the mix-



(a) T=10



(b) T=100



(c) T=200

Figure 7.4: Effect of changing *topical* and temporal similarity with different time periods(N-Extended collection, R-cut=250, df-cut=99.9%).

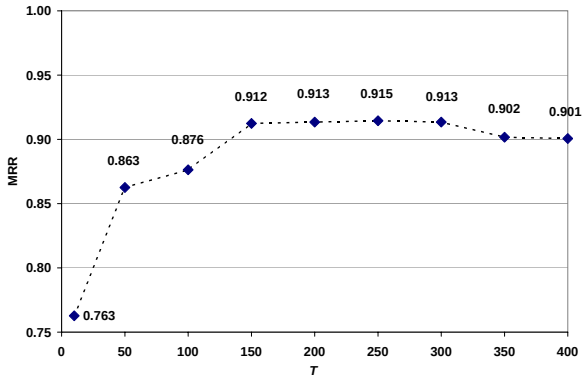


Figure 7.5: Effect of changing *topical* time period T (N-Extended collection, Sim = [Path, g_d], R-cut=250, *df*-cut=99.9).

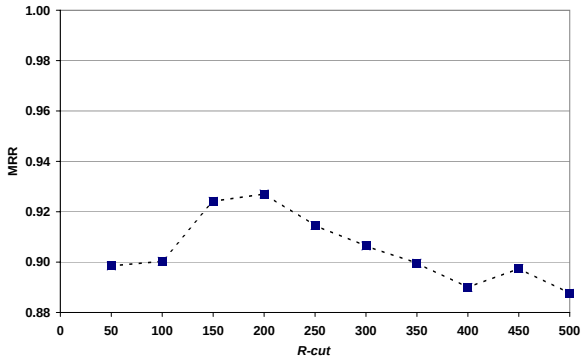


Figure 7.6: Effect of changing *topical* R-cut (N-Extended collection, Sim = [Path, g_d], T=250, *df*-cut=99.9).

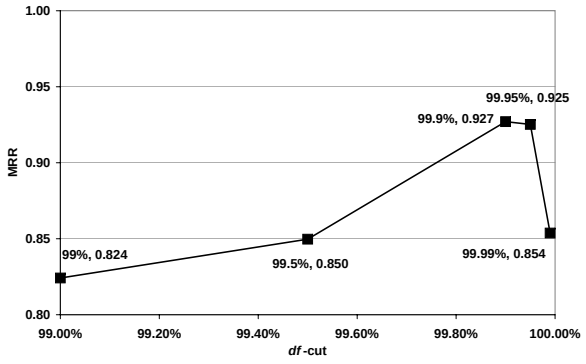


Figure 7.7: Effect of changing *topical* *df*-cut (N-Extended collection, Sim = [Path, g_d], T=250, R-cut=200).

ing coefficients λ_k and the context priors $p(x_k)$ in each case set to a uniform distribution over all included contexts. The figures and tables of this section use initials of the names of the four contexts to indicate which of them comprise a specific combination (L: Local, C: Conversational, S: Social, and T: Topical.) The "context-free" resolution (which gives no attention to the context of a given mention-query, as described in Chapter 3) was included as a baseline system and labeled "None."

There are four basic questions addressed in the experimental evaluation:

1. How does the resolution approach perform compared to other approaches?

2. How is it affected by the size of the collection?

3. Which context makes the most important contribution to the resolution task?

4. Does the mixture help?

The trained system is first tested on the training collection as described in Section 7.3.1, followed by testing on small collections as described in Section 7.3.2, and the new collection and its subsets as described in Section 7.3.3.

For statistical significance test, the Wilcoxon Matched-Pairs Signed-Ranks Test, a non-parametric test used to determine differences between sets of paired samples [90], was used. It is preferred here over the more popular Student t-test, since it doesn't assume that the difference (in measurements of the paired samples) is normally distributed.

7.3.1 Testing on Training Collection: N-Extended

Figure 7.8 shows the MRR results of the trained system on the training collection, N-Extended. The combinations are grouped based on the number of contributing contexts.

Since the individual contexts were well tuned for that collection and the mixing parameters are equally weighted, there is no statistical significance improvement from combining contexts over the best individual context, topical.

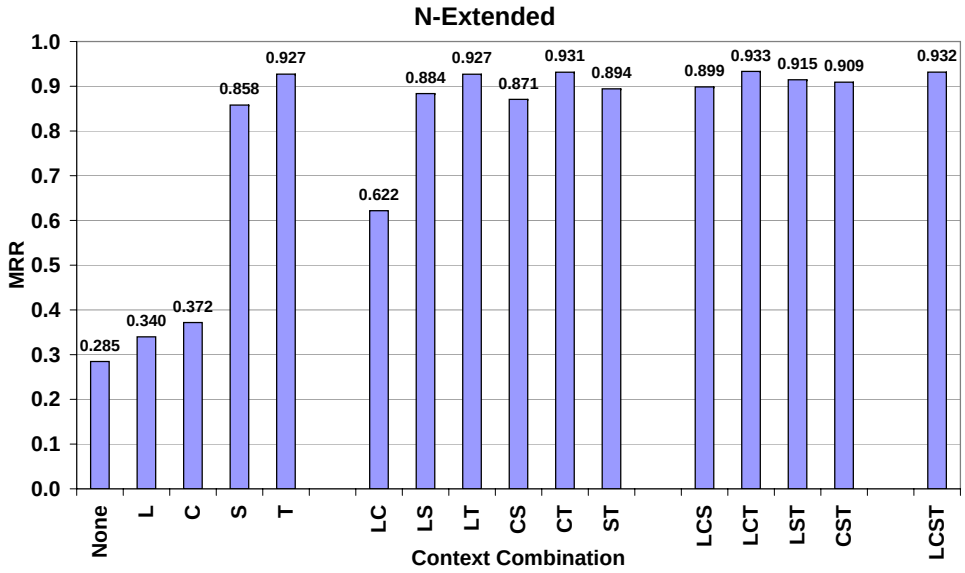


Figure 7.8: Results on N-Extended collection.

7.3.2 Testing on Small Collections

A summary of the results on the small collections are shown in Tables 7.5 and 7.6 with the best results for each test collection highlighted in bold. The table includes results of context-free resolution, the best individual context, the best combination of contexts, and the combination of *all* four contexts. It also includes the results reported in Minkov

Table 7.5: MRR results on small collections.

Test Coll.	Context-Free	Best Context		Best Comb.		LCST	Literature	
M-Sager	0.708	T	0.784	LCT	**0.905**	0.856	Minkov	0.899
M-Shapiro	0.628	S	0.844	LCS	**0.894**	0.888	Minkov	0.879
N-Subset	0.288	S	0.893	LCST	**0.934**	**0.934**	-	-

et al [63] and Diehl et al. [31] for comparison purposes.[2] Given these scores, the results compare favorably with the previously reported results for all of those collections.

Table 7.6: S@1 results on small collections.

Test Coll.	Context-Free	Best Context		Best Comb.		LCST	Literature	
M-Sager	0.627	T	0.686	LCT	**0.863**	0.765	Minkov	0.804
M-Shapiro	0.571	S, T	0.755	LCS	**0.837**	0.816	Minkov	0.779
N-Subset	0.128	S	0.833	LCST	**0.897**	**0.897**	Diehl	(0.82)

7.3.3 Testing on the New Test Collection and Subsets

Table 7.7: Summary of MRR and S@1 results on E-All and subsets.

Test Coll.	MRR	S@1
E-All	0.785	0.738
E-Enron	0.820	0.774
E-NonEnron	0.611	0.563
E-Hard	0.582	0.510

Figures 7.9, 7.10, 7.11, and 7.12 illustrate the MRR results of all context combinations on E-All and its subsets. A large drop in MRR values is generally noticed from the results reported earlier on N-Extended and the smaller subsets. For example, the highest

[2]For N-Subset collection, it is not known which 54 mention-queries Diehl et al used in [31], so it is not possible to directly compare the results reported here to theirs. Notice that they reported S@1 but not MRR values.

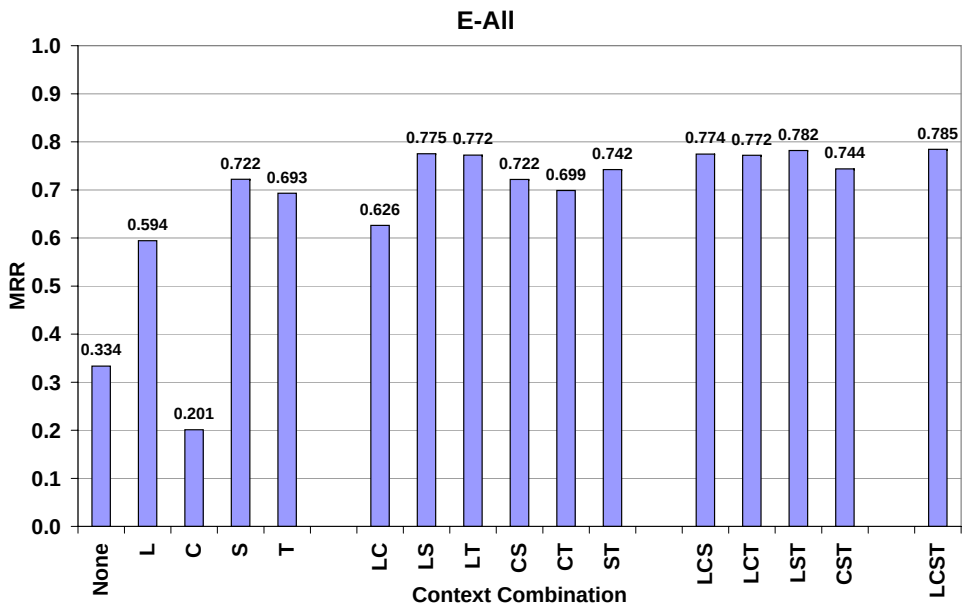


Figure 7.9: Results on E-All collection.

MRR value achieved on E-All is 0.785 (as indicated in Table 7.7) compared to values around 0.9 in the case of the other collections. This is somewhat expected for two reasons: (1) the queries in the other collections are biased; they are all resolvable to Enron employees, while the queries of E-All are fairly sampled, and (2) 25 mention-queries in E-All, about 5.5% of the query set, were not automatically recognized by *IdResolver* ; an adjusted MRR score (after removing those queries) for E-All would be 0.829. An adjusted MRR value for E-Enron is 0.867, which is closer to the results on the similar type of queries of Minkov's and Namata's collections. Notice also that MRR results on E-Enron is generally higher than E-NonEnron, which matches the difficulty measures reported in Chapter 6. Moreover, E-Hard has been experimentally proved to be hard, as it has the lowest reported MRR and S@1 values over all Elsayed's collections.

To characterize the contribution of each individual context to the results, Figure 7.13 shows the MRR achieved with each context individually and the context-free resolution

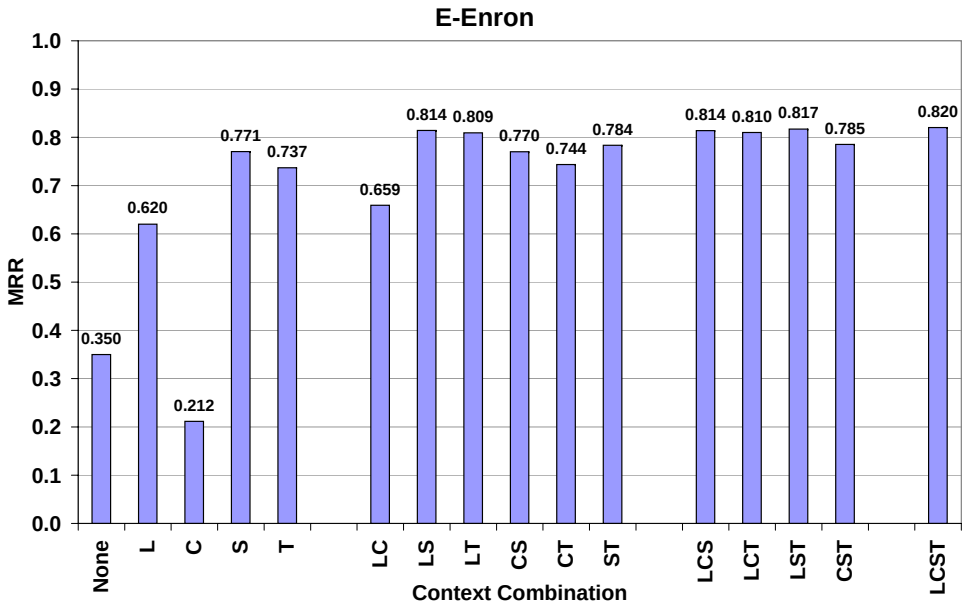


Figure 7.10: Results on E-Enron collection.

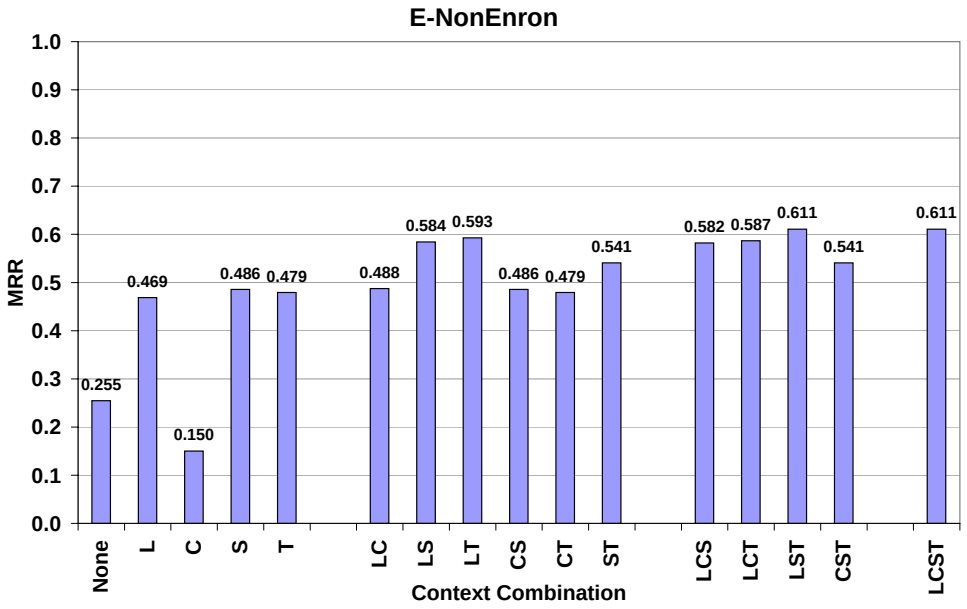


Figure 7.11: Results on E-NonEnron collection.

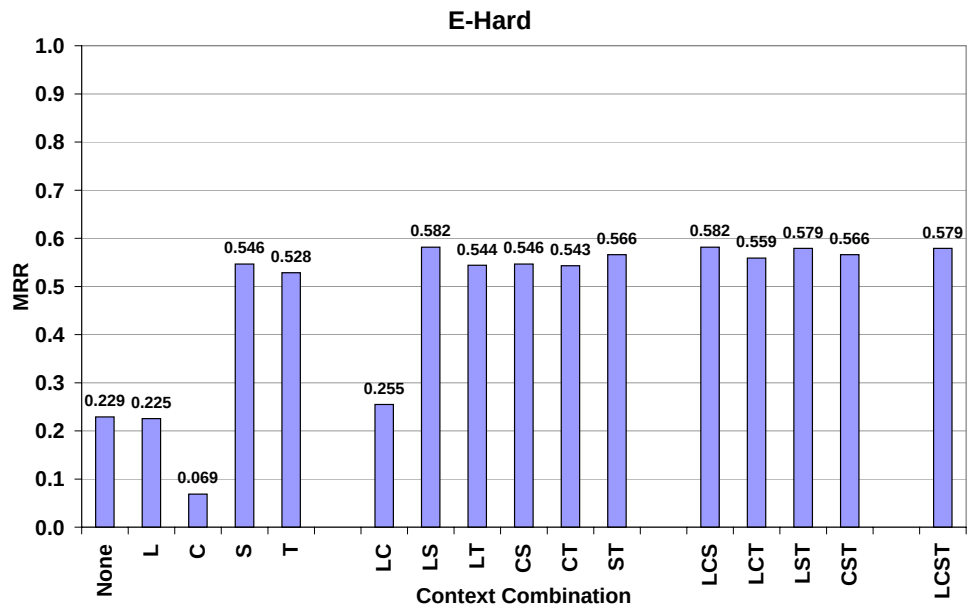


Figure 7.12: Results on E-Hard collection.

over all of the four collections. The figure indicates that the social context is clearly quite useful, more so than any other single context, for every collection. This tends to support the expectation that social networks can be as informative as content networks in email collections. The topical context also seems to be useful on its own. The local context is moderately useful on its own in the larger collections. The conversational context alone is not very informative for the larger collections.

Table 7.8: Statistical significance of differences between context contributions.

Test Coll.	LCST vs. S	S vs. T
E-All	significant ($p < 0.001$)	not significant
E-Enron	significant, ($p < 0.001$)	significant, ($p < 0.05$)
E-NonEnron	significant ($p < 0.001$)	not significant
E-Hard	not significant	not significant

The principal motivation for combining different types of contexts is that different sources may provide complementary evidence. To characterize that effect, Figure 7.13

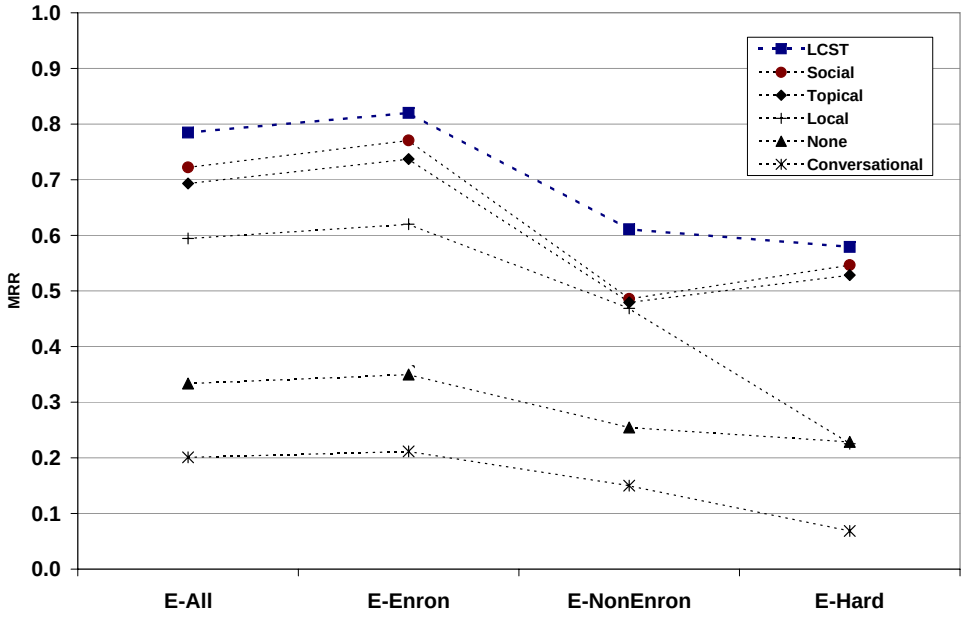


Figure 7.13: Context combination vs. individual contexts.

also illustrates the MRR curve of LCST (all contexts combined.) The figure shows a difference between LCST and social contributions, illustrated in Figure 7.14 as a percentage MRR improvement. Table 7.8 shows the results of statistical significance test on those differences. It shows that the differences are significant in all collections but E-Hard (possibly because of the few queries.) The table also shows that the MRR differences between social and topical contexts were insignificant in all collections but E-Enron.

7.3.4 Iterative Experiments

An experiment was conducted in which *IdResolver* ran for four iterations. Figures 7.15(a), 7.15(b), and 7.15(c) show the results on E-All, E-Enron, and E-NonEnron respectively, using two individual contexts (social and topical) and three different combinations (LCT, LCS, and LCST.) The MRR results show that resolution using topical context decays consistently over subsequent iterations, while the resolution using social context is

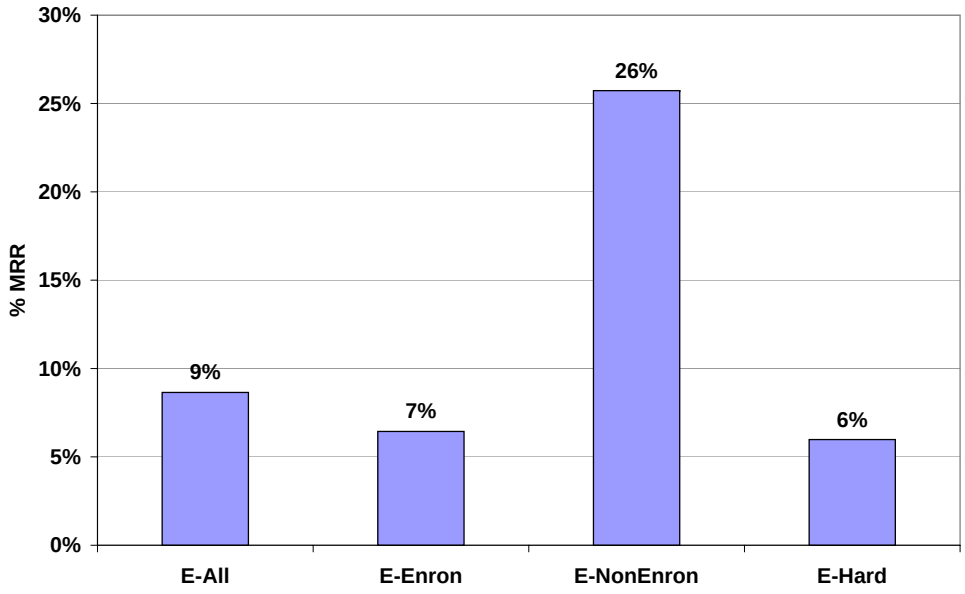


Figure 7.14: Percentage improvement over best context.

stable or improving at the second iteration, as shown in Figures 7.15(a) and 7.15(c). The combinations of contexts naturally follow similar patterns.

Figures 7.16(a) and 7.16(b) show the difference in performance between the first two iterations of the topical context resolution over E-Enron and E-NonEnron collections respectively, sorted by the difference in Reciprocal Rank (RR). They also show the difference in the score (i.e., probability) assigned by the resolution algorithm to the actual true referent of each query over the two iterations. Figures 7.17(a) and 7.17(b) illustrate the same but for the social context resolution.

The figures indicate that:

- About 88% of the queries had a stable RR score over the two iterations, and thus only 12% of them were affected.

- Generally, there are more queries where *IdResolver* got worse in the second iteration (measured by RR) than queries where it got better.

- As expected, there is a positive (but not too strong) correlation between the RR score and the probability assigned by *IdResolver* to the true referent.

- The performance loss is higher in topical resolution than social resolution and over E-Enron queries than E-NonEnron queries.

The last observation is also supported by Figure 7.18 which compares the average difference in performance per query over the four different cases. It is interesting that the MRR score of the social context resolution is getting a bit worse, although the average difference of the probability assigned to the true referent is increased. This is due to the score quantization levels of the MRR measure. The figure clearly shows that social context resolution generally is benefitted (or at least less harmed) by the iterative approach.

Upon inspection of most of the adversely-affected queries in the four cases, it was obvious that there are two main reasons of the performance decay:

1. There are many mistakenly recognized non-names (such as "may") for which the negative effect is growing as the number of iterations increases by creating noisy resolution vectors – this means that a better name recognition system than the non-context-sensitive one used in the experiments might be needed to get benefit of the iterative resolution approach.

2. Those non-names often refer to (and thus vote for) identity models that are actually non-personal, such as mailing lists or faked addresses. One of these (which is "40enron@enron.com") dominates the resolution of many queries on the second iteration simply because many common names are associated with it and their resolution vectors get more biased over time. This kind of model should be filtered out

of the candidate list early on in the process.

As a side experiment, the context mixing parameters λ_l's were tuned over the training collection and the tuned system was used with the same iterative experiment design. The results were similar to the ones shown here for equally weighted contexts when those tuned weights were used on E-All collection.

7.3.5 Efficiency

Table 7.9: Extracted mentions from CMU-Enron collection.

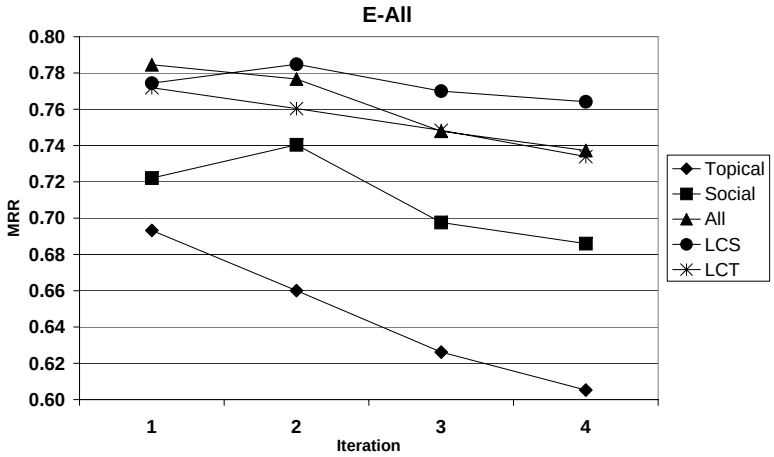
from Main body	999,291
from Subject	51,386
from Main Header	1,642,923
from Quoted Body	442,099
from Quoted Header	522,716
Email-addresses	1,746,636
Single-token Names	1,331,375
Multi-token Names	580,407

Table 7.9 shows number of extracted mentions from different message parts for the whole CMU-Enron email collection. A total of about 3.65 million mentions (including email addresses and multi-token names) were extracted; only about 1.3 million of them are single-token names that required non-trivial resolution.

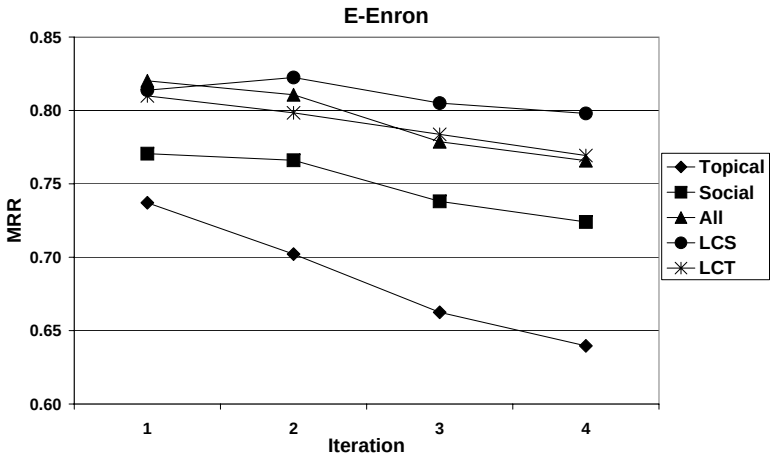
Table 7.10: Time used for major MapReduce processing stages (in minutes).

Packing	48	Social: Indexing	1.5	Topical: Indexing	1.5
Preprocessing	5	Social: Pairwise Sim.	5	Topical: Pairwise Sim.	5-13
Local: Total	9	Social: Resolution	13	Topical: Resolution	17-35
Conv.: Total	10	Social: Total	35	Topical: Total	45-75
Merging Scores	10				

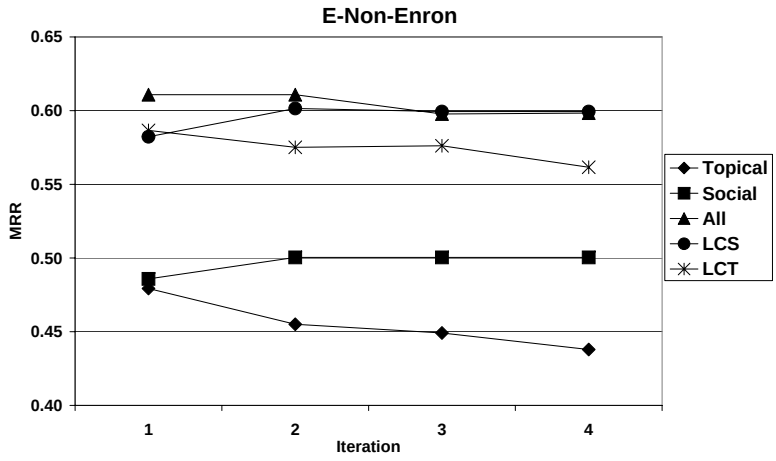
Table 7.10 reports average (over several different runs with different parameter set-



(a) E-All collection

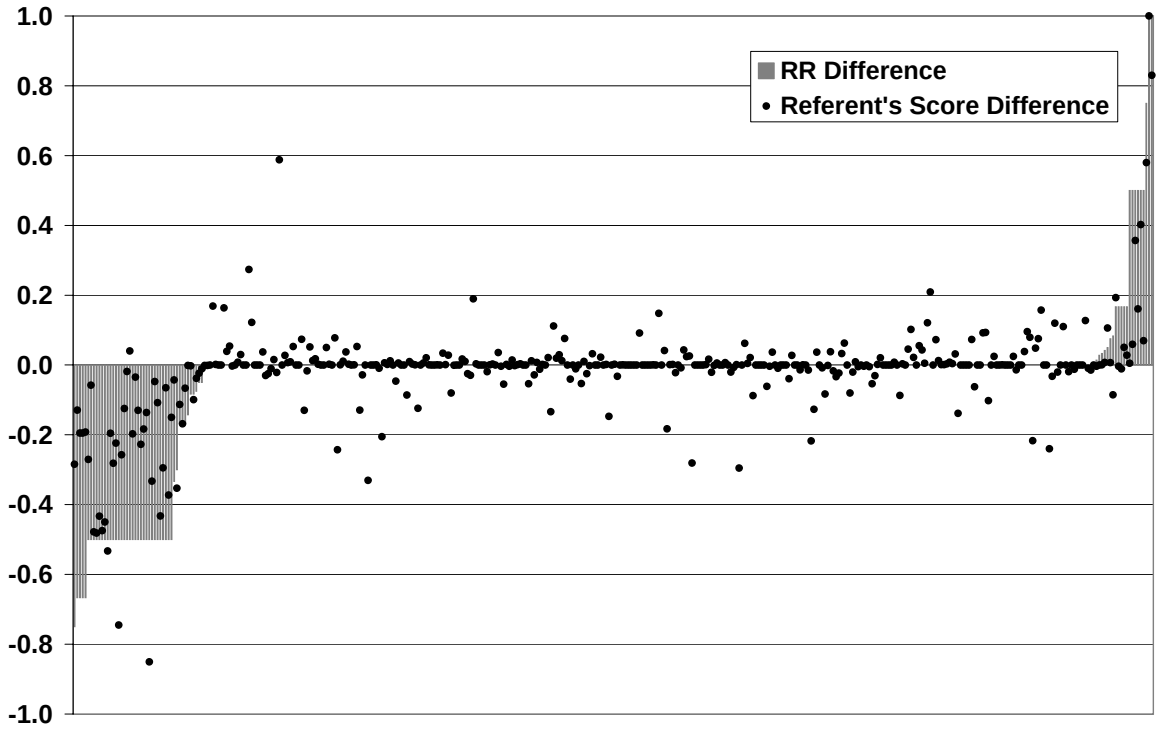


(b) E-Enron collection

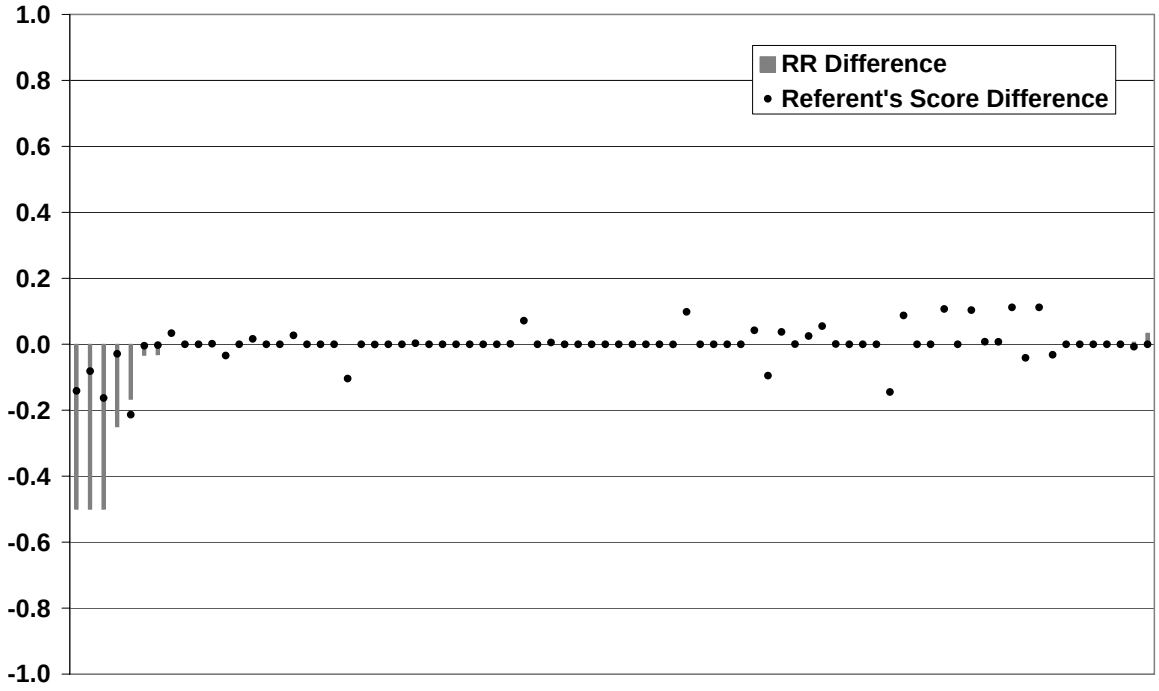


(c) E-NonEnron collection

Figure 7.15: Results of the iterative experiments.

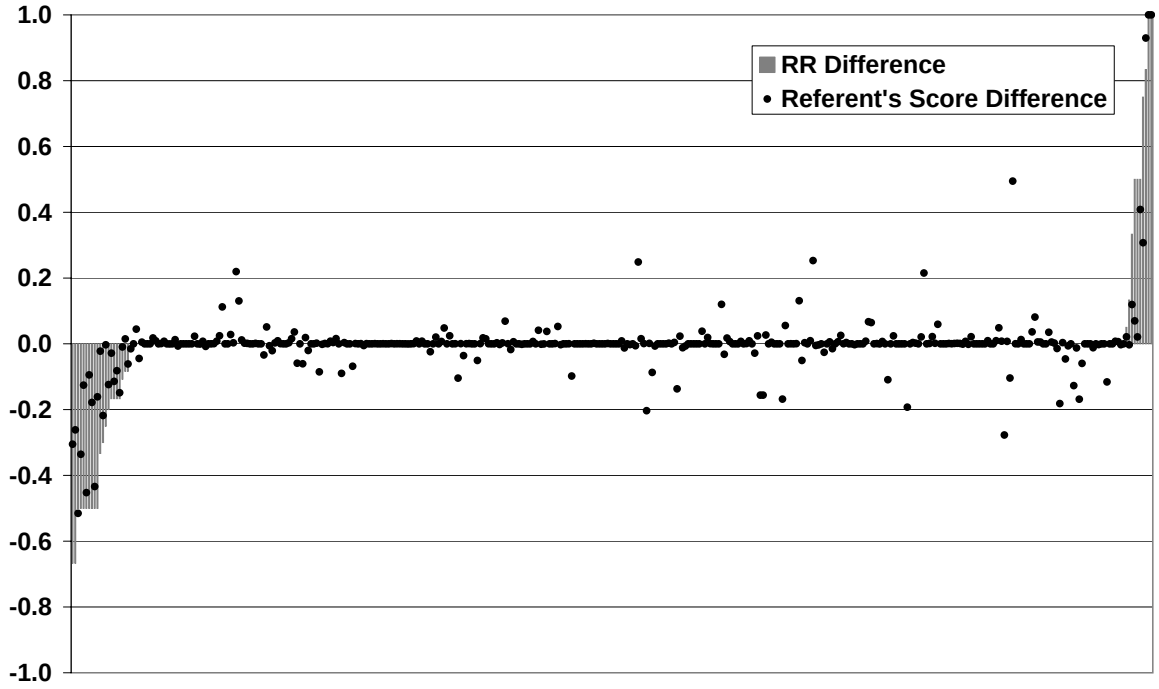


(a) E-Enron collection (390 queries)

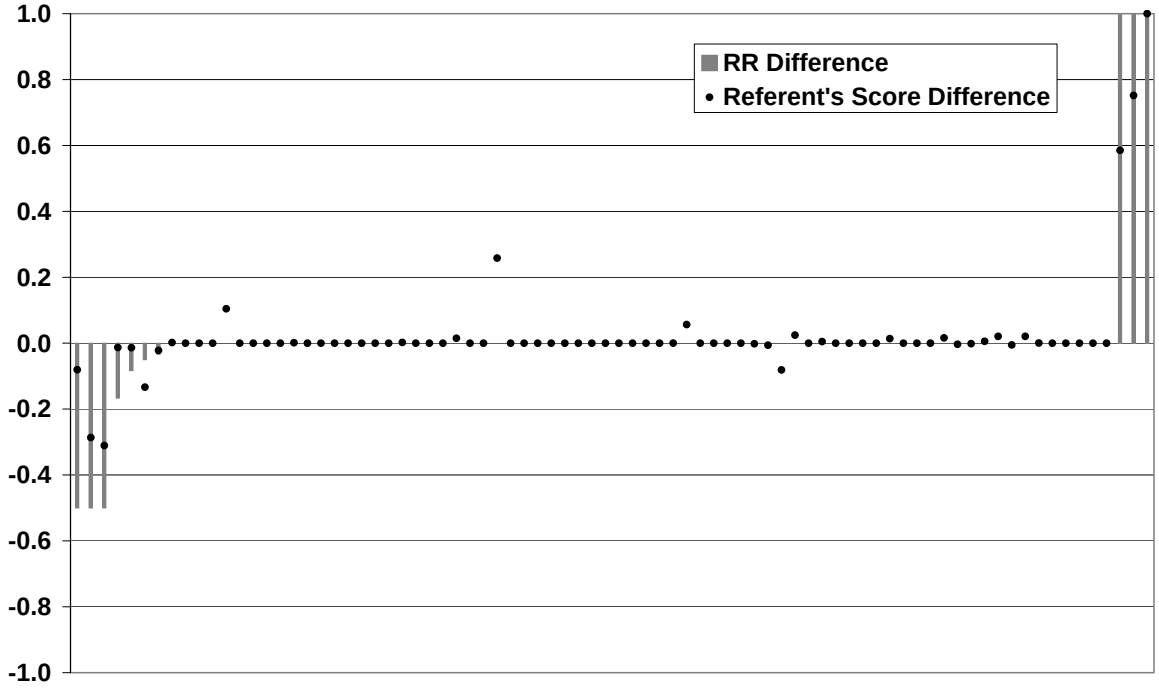


(b) E-NonEnron collection (80 queries)

Figure 7.16: Difference between the first two iterations of *topical* context resolution over mention-queries sorted by RR difference.



(a) E-Enron collection (390 queries)



(b) E-NonEnron collection (80 queries)

Figure 7.17: Difference between the first two iterations of *social* context resolution over mention-queries sorted by RR difference.

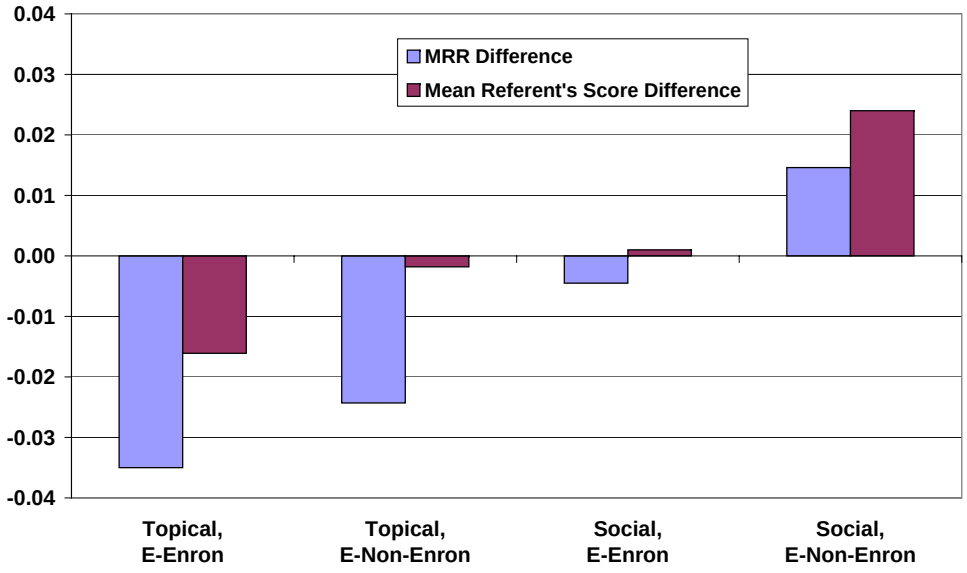


Figure 7.18: Average difference between the first two iterations per query.

tings) time durations in minutes spent in major processing steps. Notice that the Hadoop cluster was shared among different users, and that the processing time might change based on the available resources.

As the table indicates, the end-to-end runs, resolving about 1.3 million extracted mentions, take roughly 2 to 3 hours on 200 processing nodes, including indexing, extraction of all names in the collection, the pairwise similarity for both social and topical contexts, name resolution on both context graphs, and the context combination process.

7.4 Chapter Summary

In this chapter, the resolution results on all of the available test collections (including the largest collection, developed and described in Chapter 6, and its subsets) are reported. The next chapter concludes the dissertation, discussing some limitations and possible directions for future work.

Chapter 8

Conclusion and Future Work

This dissertation has tackled the problem of resolving personal identity in the context of large email collections. To "resolve identity," solution of two problems were needed: (1) modeling the *identity* of the participants, and (2) *resolving* mentions to these identities. A simple computational model of identity, built on extracting unambiguous references (e.g., full names from headers, or nicknames from free-text signatures) to people from the whole collection, was presented. This model was then leveraged to resolve mentions using a generative probabilistic approach that expands the context surrounding a mention in four directions: the message where the mention was observed, the thread that includes that message, topically-related messages, and messages sent or received by the original communicating parties. The algorithm relies on less ambiguous references (e.g., email addresses or full names) that are observed in some context of a given mention to rank potential referents of that mention.

In order to jointly resolve all mentions in the collection, a parallel implementation has been presented using the MapReduce distributed-programming framework. For scalable context expansion of all mentions, a parallel algorithm was proposed for efficiently computing pairwise document similarity in large collections, a general solution that can be used in other applications as well.

The resolution approach compares favorably with previously-reported techniques

on the small test collections previously-reported in the literature. However, the mention-queries in those collections, besides being relatively few in number, are limited in that all refer to people for whom a substantial amount of evidence would be expected to be available; thus omitting the "long tail" of the identity distribution for which less evidence is available. This motivated the development of a new test collection that now is, to the best of the author's knowledge, the largest and best-balanced test collection available for the task. To build this collection, a user study was conducted, which provided some insight into the difficulty of the task, how time-consuming it is when humans perform it, and the reliability of their task performance. The study revealed that at least 80% of the 470 annotated mentions were resolvable to people who had sent or received email within the same collection.

The new test collection was used to experimentally evaluate the resolution system. The results highlight the importance of the social context when resolving mentions in email. Moreover, the results show that combining evidence from multiple types of contexts yields better resolution than what can be achieved using any individual context. *IdResolver* one-best selection is correct 74% of the time when tested on the full set of the mention-queries, and 51% of the time when tested on the mention-queries labeled as "hard" by the annotators. Experiments run with iterative reformulation of the resolution algorithm resulted in modest gains only for the second iteration in the social context expansion. Finally, *IdResolver* 's MapReduce implementation resolves about 1.3 million mentions on a 200-node Hadoop cluster in 2 to 3 hours.

This chapter discusses some limitations of the work done in this dissertation in Section 8.1 and suggests possible directions for future work in Section 8.2 before it concludes

by highlighting some implications of the work in Section 8.3.

8.1 Limitations

All experimental investigations are subject to the limitations of the experimental setting. Among the most significant limitations of the setting are the following:

- The CMU version of the Enron collection, and subsets of it, was used in all of the experiments because it was the biggest collection available at the time for research. Therefore, the conclusions drawn are limited to that collection and can not be generalized before testing the proposed technique on different collections.

- In order to focus the dissertation work on the resolution framework, the detection of unresolvable mentions was not considered. That required the resolution algorithm to make an assumption that all mentions are resolvable and to actually attempt to resolve every mention that could be automatically recognized. The evaluation results focused only on resolvable mentions, but incorrect resolution of unresolvable mentions may adversely affect the iterative solution.

- *IdResolver* uses dictionaries of names, nicknames, and non-names for mapping nicknames to first names and for automatically recognizing personal names in the whole collection. However, the system does not make use of additional resources that are more closely related to the collection itself, such as the list of employees and their job positions. Moreover, the use of those dictionaries and the nickname detection approach that benefits from specific regularities of the user behavior might

also make it harder to generalize the proposed technique to other languages and cultures.

- In building the new test collection, the recruited human annotators were all university students who had no prior experience with the people involved nor with the topics discussed in the collection. This might have limited the accuracy of the annotations to some extent. The timing guidelines provided to them might also have been a limiting factor for decisions regarding unresolvable mentions.

8.2 Future Work

The results of this dissertation open several important directions for future work. Opportunities can be envisioned to improve the resolution approach, to use other email collections, to apply similar techniques in other applications, and to leverage the ideas presented to solve related problems. This section discusses those opportunities.

8.2.1 Improving The Resolution Algorithm

- An intuitive generative probabilistic solution, that was experimentally shown to be effective, was proposed to solve the problem. Alternative approaches could be designed (e.g., a discriminative model) that can possibly leverage a larger number of features from the context of a mention (e.g., other mentions and people sending or receiving emails in that context.) Moreover, the parameters of the Bayesian network of the identity model were estimated using maximum likelihood estimates, but alternatively, the context-free estimates could be used as priors to update the

estimated values of these parameters using Bayesian updating model.

- In the iterative resolution approach, the resolution of *all* mentions are attempted simultaneously. A different approach could take advantage of the fact that not all mentions are equally ambiguous, so it might be better to focus on the least ambiguous mentions before the more ambiguous ones. The level of ambiguity may not be easy to determine, but the number of candidates could be a possible clue. Another possibility is to divide the mention graph into sub-graphs, separately resolve each sub-graph, and then propagate resolutions among them. Sub-graphs might, for example, be partitioned by time periods.

- The study described in Chapter 6 indicates that about 20% of mentions that were manually extracted and then randomly selected were unresolvable. Ultimately, there is a need to automatically recognize when not to attempt a resolution for a name-mention. There are many features one can think of for a binary classification problem (e.g., the strength of evidence that supports the most probable referent.)

8.2.2 Using Other Email Collections

- A new version of the Enron collection is expected to be released soon by NIST as a test collection for the Legal Track of the Text REtrieval Conference (TREC) [66]. That collection will be interesting from this work's perspective in two ways:

  - It is expected to be a superset of the CMU collection (i.e., larger than the version used in this dissertation) so it can be used to test the effectiveness and

efficiency of the proposed techniques using additional evidence from previously unseen messages.

  - It will include attachments (that were stripped out before releasing the CMU version of Enron collection.) Two versions of the attachments will be available: native format (e.g., a Microsoft Excel sheet) and a version that includes any text that could be automatically extracted. The availability of easily-processed attachments raises the question of how to use them in the context expansion framework to better resolve mentions (e.g., by treating co-attachments as a fifth type of context.)

- Another email collection that could be used is Shneiderman's collection. That collection includes about 45,000 emails that were intentionally retained between 1984 and 1998 by Professor Ben Shneiderman at University of Maryland [74]. Some advantages of using that collection could be:

  - It is a personal collection, where one communicating entity is common in all emails (i.e., a snapshot of one mailbox.)

  - It spans a longer period of time.

  - It is expect that it would be relatively easy to obtain accurate name annotations with the help of Professor Shneiderman.

  One potential disadvantage of using that collection is that the collection may not be as widely available for other research teams. Another is that the size of the collection is considerably smaller than even the CMU-Enron collection.

8.2.3 Content Resolution

- A user might also need to resolve unfamiliar terms other than names. Figure 8.1 shows an example email from the Enron email collection.

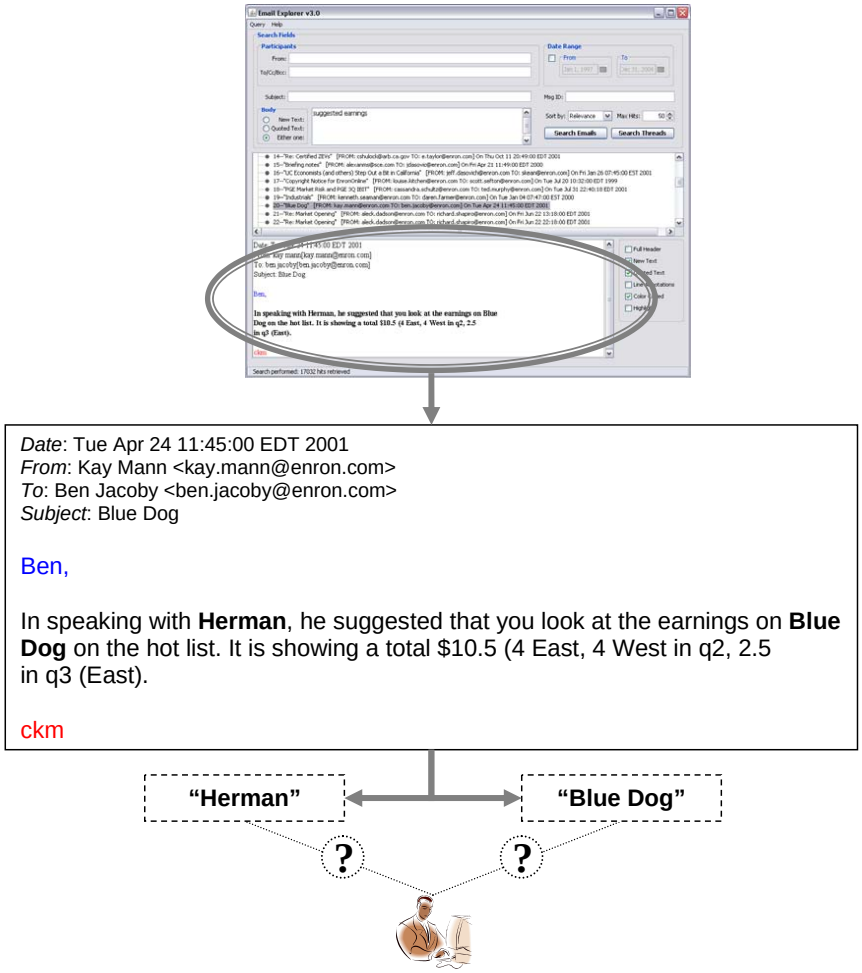


Figure 8.1: Two key aspects in email conversations: content and people.

In that example, the user may be interested not only in determining who "Herman" is, but also in determining the meaning of "Blue Dog". These two problems, identity and content resolution, share some common characteristics, and the techniques used to resolve the identities might also be useful for resolving term meaning.

Document expansion techniques presented in [85] and [17] might be worth exploring as a solution for the content resolution problem, guided by the context expansion approach followed in identity resolution in this work.

- Identity resolution can affect content resolution and vice versa. There is some potential that resolution of term meaning might be helpful for resolving the identities (e.g., a clue about"Herman" may shed some light on an explanation for "Blue Dog" and vice versa.) This suggests either a joint solution, or that two separate solutions may be complementary to each other.

8.2.4 Applications

- The general interest of this work is in resolving identity in conversational text, although to date the proposed techniques were applied only to email. Applying the solution framework to other genres (e.g., chat room conversations) is an obvious next step. Moreover, it might be worthwhile to try to apply the proposed techniques on multiple genres simultaneously. For example, some transcribed phone calls are available among some participants in the Enron collection.

- The results of the resolution system could be extrinsically evaluated by using the system as a preprocessing step in some more complex task (e.g., social network analysis, email retrieval, or an email exploration tool) for which an insightful evaluation measure could be crafted.

- The proposed algorithm for computing pairwise document similarity could be leveraged for other tasks such as a distributed document clustering using MapReduce,

where the similarity matrix could be used as a basis for merging or splitting clusters.

8.3 Implications

A number of implications of this work can be envisioned. The work provides some insights into some of the major problems in handling conversational text in general, and email in particular. It has given a structure and a solution to the problem that both can be adopted in other similar genres. *IdResolver* can help users better understand (or make sense of) large collections of emails when they are non-participants. It also illustrates the need to design scalable solutions to problems that can practically arise due to the growing scale of available data. Finally, it suggests several directions for future work.

Appendix A

System Specification

The system built in the work of this dissertation has two main components:

1. *IdModeler* : the component responsible of building the models of identity using the "unambiguous" observable attributes from the the whole set of emails in the collection.

2. *IdResolver* : the component responsible of resolving mentions in emails using the context expansion approach and the probabilistic generative model.

A.1 *IdModeler*

IdModeler is written completely in Java 1.5 and uses Lucene 2.3 to index the emails using both header fields (e.g., "FromName" and "ToAddress") and body fields (e.g., "mainBody" and "quotedBody".) The code is object-oriented with separate classes for collections, emails (with header and body components), and identity models (including all statistics of observed attributes). Utility classes include indexer, searcher, email readers (one per collection that parses the email format), attribute and association detector, thread links' detector, identity extractor (which is the core service that calls the other components to produce the final set of identity models.)

IdModeler runs sequentially, performing a few passes over the whole set of emails. The implementation details are discussed in Chapter 3, Section 3.3. The major blocks and

data flow are specifically illustrated in Figure 3.3.

A.2 *IdResolver*

IdResolver is written completely in Java 1.5 and uses Hadoop 0.17.2 for a scalable implementation of the resolution algorithm. All the components of *IdResolver* have been designed to fit the MapReduce model, as detailed in Chapter 5 Figure 5.2. Each block in the figure indicates one (or more) MapReduce job(s). The final output is a ranked list of candidates of every mention in every email. The code ideally runs on a Hadoop cluster (i.e., a set of commodity machines running Hadoop Distributed File System (HDFS)) but can also run in a stand-alone mode.

A.3 System and Data Usage

Some parts of the system and the data produced have been already used by other researchers. Multiple versions of the search system, used as an assistant tool in annotating name mentions (as shown in Chapter 6), were used by Prof. Ben Shneiderman at Computer Science Department and several researchers at Joint Institute for Knowledge Discovery (JIKD) at the University of Maryland. The code that implements the efficient computation of pairwise document similarity was used in the coreference resolution task in the 2008 Automatic Content Extraction evaluation (ACE) [67] by a joint team of the University of Maryland and Johns Hopkins University. Multiple versions of the topical similarity matrices (with different parameters) produced by the topical expansion have been used by Prof. Carey Prebe and his colleagues at Department of Applied Mathemat-

ics at Johns Hopkins University.

It is also planned to release that implements the efficient computation of pairwise document similarity as a part of a new open source information retrieval engine called "Ivory" in a joint work of the University of Maryland and Yahoo. The source code of the two major system components *IdModeler* and *IdResolver* are also planned to be released along with the final resolution data for the CMU Enron collection.

Bibliography

[1] RFC 822 - Standard for ARPA Internet text messages. http://www.ietf.org/rfc/rfc0822.txt.

[2] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[3] Elif Aktolga, Marc-Allen Cartright, and James Allan. Cross-document cross-lingual coreference retrieval. In *CIKM '08: Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 1359–1360, Napa Valley, California, United States, 2008.

[4] James Allan, editor. *Topic Detection and Tracking: Event-based Information Organization.* Kluwer Academic Publishers, Boston, Massachusetts, United States, 2002.

[5] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[6] Shlomo Argamon, Marin Saric, and Sterling S. Stein. Style mining of electronic messages for multiple authorship discrimination: first results. In *KDD '03: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 475–480, Washington, D.C., United States, 2003.

[7] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 79–85, Montreal, Quebec, Canada, 1998.

[8] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR '06: Proceedings of the 29th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50, Seattle, Washington, United States, 2006.

[9] Krisztian Balog and Maarten de Rijke. Finding experts and their details in e-mail corpora. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 1035–1036, Edinburgh, Scotland, 2006.

[10] Krisztian Balog and Maarten de Rijke. Non-local evidence for expert finding. In *CIKM '08: Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 489–498, Napa Valley, California, United States, 2008.

[11] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 131–140, Banff, Alberta, Canada, 2007.

[12] R. Bekkerman, A. McCallum, and G. Huang. Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. Technical Report IR-418, Center of Intelligent Information Retrieval, UMass Amherst, 2004.

[13] Omar Benjelloun, Hector Garcia-Molina, Hideki Kawai, Tait Eliott Larson, David Menestrina, Qi Su, Sutthipong Thavisomboon, and Jennifer Widom. Generic entity resolution in the SERF project. *IEEE Data Engineering Bulletin*, 29(2):13–20, 2006.

[14] Indrajit Bhattacharya and Lise Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM '06: Proceedings of The SIAM International Conference on Data Mining*, Bethesda, Maryland, United States, 2006.

[15] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[16] Indrajit Bhattacharya, Lise Getoor, and Louis Licamele. Query-time entity resolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 529–534, Philadelphia, Pennsylvania, United States, 2006.

[17] Bodo Billerbeck and Justin Zobel. Document expansion versus query expansion for ad-hoc retrieval. In A. Turpin and R. Wilkinson, editors, *Proceedings of the tenth Australasian Document Computing Symposium*, pages 34–41, Sydney, Australia, December 2005.

[18] Matthias Blume. Automatic entity disambiguation: Benefits to NER, relation extraction, link analysis, and inference. In *Proceedings of International Conference on Intelligence Analysis*, May 2005.

[19] U.S. Census Bureau. Frequently occurring first names and surnames from the 1990 census. http://www.census.gov/genealogy/names/.

[20] Christopher S. Campbell, Paul P. Maglio, Alex Cozzi, and Byron Dom. Expertise identification using email communications. In *CIKM '03: Proceedings of the twelfth International Conference on Information and Knowledge Management*, pages 528–531, New Orleans, Louisiana, United States, 2003.

[21] Vitor R. Carvalho and William W. Cohen. Learning to extract signature and reply lines from email. In *CEAS '04: Proceedings of the first Conference on Email and Anti-Spam*, Mountain View, California, United States, August 2004.

[22] Vitor R. Carvalho and William W. Cohen. On the collective classification of email "speech acts". In *SIGIR '05: Proceedings of the 28th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–352, Salvador, Brazil, 2005.

[23] B. Catanzaro, N. Sundaram, and K. Keutzer. A MapReduce framework for programming graphics processors. In *Proceedings of The Third Workshop on Software*

Tools for MultiCore Systems (STMCS '08), Boston, Massachusetts, United States, April 2008.

[24] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary R. Bradski, Andrew Y. Ng, and Kunle Olukotun. Map-Reduce for machine learning on multicore. In *NIPS '07: Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, pages 281–288, Vancouver, B.C., Canada, December 2007.

[25] Andres Corrada-Emmanuel. Enron email dataset research. http://ciir.cs.umass.edu/ corrada/enron/index.html.

[26] Nick Craswell, Arjen P. de Vries, and Ian Soboroff. Overview of the TREC-2005 enterprise track. In *TREC '05: Processdings of the 14th Text REtrieval Conference*, Gaithersburg, Maryland, United States, November 2005.

[27] Aron Culotta, Ron Bekkerman, and Andrew McCallum. Extracting social networks and contact information from email and the web. In *CEAS '04: Proceedings of the first Conference on Email and Anti-Spam*, Mountain View, California, United States, 2004.

[28] Olivier de Vel. Mining e-mail authorship. In *Workshop on Text Mining, ACM International Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts, United States, August 2000.

[29] Olivier de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Record*, 30(4):55–64, 2001.

[30] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI '04: Proceedings of the 6th Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, California, United States, 2004.

[31] Chris Diehl, Lise Getoor, and Galileo Namata. Name reference resolution in organizational email archives. In *SDM '06: Proceddings of SIAM International Conference on Data Mining*, Bethesda, Maryland , United States, April 2006.

[32] Byron Dom, Iris Eiron, Alex Cozzi, and Yi Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In *DMKD '03: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 42–48, San Diego, California, United States, 2003.

[33] Mark Dredze, Hanna M. Wallach, Danny Puller, and Fernando Pereira. Generating summary keywords for emails using topics. In *IUI '08: Proceedings of the 13th International Conference on Intelligent user interfaces*, pages 199–206, Gran Canaria, Spain, 2008.

[34] Chris Dyer, Aaron Cordova, Alex Mont, , and Jimmy Lin. Fast, easy, and cheap: Construction of statistical machine translation models with mapreduce. In *the Third Workshop on Statistical Machine Translation at ACL 2008*, June 2008.

[35] Tamer Elsayed, Jimmy Lin, and Douglas Oard. Pairwise document similarity in large collections with mapreduce. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio, United States, June 2008.

[36] Tamer Elsayed, Galileo Namata, Lise Getoor, and Douglas W. Oard. Personal name resolution in email: A heuristic approach. Technical Report UMIACS LAMP-TR-150, University of Maryland, March 2008.

[37] Tamer Elsayed and Douglas W. Oard. Modeling identity in archival collections of email: A preliminary study. In *CEAS '06: Proceedings of the third Conference on Email and Anti-Spam*, Mountain View, California, United States, July 2006.

[38] Tamer Elsayed, Douglas W. Oard, and Galileo Namata. Resolving personal names in email using context expansion. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, United States, June 2008.

[39] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., Upper Saddle River, New Jersy, United States, 1992.

[40] Carmen Galvez and Félix Moya-Anegón. Approximate personal name-matching through finite-state graphs. *Journal of the American Society for Information Science and Technology*, 58(13):1960–1976, 2007.

[41] Chung Heong Gooi and James Allan. Cross-document coreference on a large scale corpus. In *HLT-NAACL 2004: Proceedings of Human Language Technology Conference / North American chapter of the Association for Computational Linguistics annual meeting*.

[42] Bingsheng He, Wenbin Fang, Qiong Luo, Naga K. Govindaraju, and Tuyong Wang. Mars: a MapReduce framework on graphics processors. In *PACT '08: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, pages 260–269, Toronto, Ontario, Canada, 2008.

[43] Ralf Holzer, Bradley Malin, and Latanya Sweeney. Email alias detection using social network analysis. In *LinkKDD '05: Proceedings of the 3rd International Workshop on Link discovery*, pages 52–57, Chicago, Illinois, United States, 2005.

[44] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 59–72, Lisbon, Portugal, 2007.

[45] Hyunmo Kang, Catherine Plaisant, Bongshin Lee, and Benjamin B. Bederson. Exploring content-actor paired network data using iterative query refinement with

NetLens. In *JCDL '06: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital libraries*, pages 372–372, Chapel Hill, North Carolina, United States, 2006.

[46] P. S. Keila and D. B. Skillicorn. Detecting unusual email communication. In *CASCON '05: Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative research*, pages 117–125, Toranto, Ontario, Canada, 2005.

[47] Bryan Klimt and Yiming Yang. The Enron corpus: A new dataset for email classification research. *Machine Learning: ECML 2004*, 3201:217–226, November 2004.

[48] Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *CEAS '04: Proceedings of the first Conference on Email and Anti-Spam*, Mountain View, California, United States, July 2004.

[49] Michal Laclavik, Martin Seleng, and Ladislav Hluchy. Towards large scale semantic annotation built on mapreduce architecture. *Computational Science*, 5103:331–338, 2008.

[50] David D. Lewis. The TREC-4 filtering track. In *TREC '97: Processdings of the 6th Text REtrieval Conference*, Gaithersburg, Maryland, United States, November 1997.

[51] David D. Lewis and Kimberly A. Knowles. Threading electronic mail: a preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[52] Xin Li, Paul Morie, and Dan Roth. Semantic integration in text: from ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*, 26(1):45–58, 2005.

[53] Jimmy Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. In *SIGIR '09: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 155–162, Boston, Massachusetts, United States, 2009.

[54] Michael D. Linderman, Jamison D. Collins, Hong Wang, and Teresa H. Meng. Merge: a programming model for heterogeneous multi-core systems. In *ASPLOS XIII: Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 287–296, Seattle, Washington, United States, 2008.

[55] Steve Lohr. Google and IBM join in "cloud computing" research. *New York Times*, October 2007.

[56] Bradley Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counter-terrorism, and Security, in conjunction with the SIAM International Conference on Data Mining*, Newport Beach, California, United States, April 2005.

[57] Gideon S. Mann and David Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh Conference on Natural language learning at HLT-NAACL 2003*, pages 33–40, 2003.

[58] Steve Martin, Anil Sewani, Blaine Nelson, Karl Chen, and Anthony D. Joseph. Analyzing behaviorial features for email classification. In *CEAS '05: Proceedings of the second Conference on Email and Anti-Spam*, Stanford, California, United States, August 2005.

[59] Robert McArthur and Peter Bruza. Discovery of implicit and explicit connections between people using email utterance. In *Proceedings of the Eighth European Conference of Computer-supported Cooperative Work*, pages 21–40, Helsinki, Finland, September 2003.

[60] Andrew McCallum, Andres Corrada-Emmanuel, and XueruiWang Wang. Topic and role discovery in social networks. In *IJCAI '05: Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 786–791, Edinburgh, Scotland, UK, August 2005.

[61] Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *International Joint Conference on Artificial Intelligence Workshop on Information Integration on the Web*, pages 79–84, Acapulco, Mexico, August 2003.

[62] Andrew W. McNabb, Christopher K. Monson, and Kevin D. Seppi. Mrpso: Mapreduce particle swarm optimization. In *GECCO '07: Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation*, pages 177–177, London, United Kingdom, 2007.

[63] Einat Minkov, William W. Cohen, and Andrew Y. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR '06: Proceedings of the 29th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 27–34, Seattle, Washington, United States, 2006.

[64] Einat Minkov, Richard C. Wang, and William W. Cohen. Extracting personal names from email: applying named entity recognition to informal text. In *HLT '05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 443–450, Morristown, NJ, United States, 2005. Association for Computational Linguistics.

[65] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *Queue*, 6(2):40–53, 2008.

[66] Douglas W. Oard, Bruce Hedin, Stephen Tomlinson, and Jason R. Baron. Overview of the trec 2008 legal track. In *TREC '08: Processdings of the 17th Text REtrieval Conference*, Gaithersburg, Maryland, United States, November 2008.

[67] National Institute of Standard Technology (NIST). Automatic Content Extraction 2008 evaluation plan (ACE' 08).

[68] National Institute of Standard Technology (NIST). Text REtrieval Conference (TREC). http://trec.nist.gov/.

[69] J. Scott Olsson and Douglas W. Oard. Improving text classification for oral history archives with temporal domain knowledge. In *SIGIR '07: Proceedings of the 30th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 623–630, Amsterdam, The Netherlands, 2007.

[70] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, and J.C. Phillips. GPU computing. 96(5):879–899, May 2008.

[71] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University.

[72] Spiros Papadimitriou and Jimeng Sun. Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 512–521, 2008.

[73] Ted Pedersen, Amruta Purandare, and Anagha Kulkarni. Name discrimination by clustering similar contexts. *Computational Linguistics and Intelligent Text Processing*, 3406:226–237, 2005.

[74] Adam Perer, Ben Shneiderman, and Douglas W. Oard. Using rhythms of relationships to understand email archives. *Journal of the American Society for Information Science and Technology*, 57(14):1936–1948, 2006.

[75] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, 1998.

[76] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[77] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, and Christos Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *HPCA '07: Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pages 13–24, 2007.

[78] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 622–629, Ann Arbor, Michigan, 2005.

[79] Patric Reuther. Personal name matching: New test collections and a social network based approach. Technical Report Mathematics/Computer Science TR-06-01, University of Trier, 2006.

[80] Mark Sanderson. Word sense disambiguation and information retrieval. In *SIGIR '94: Proceedings of the 17th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 142–151, Dublin, Ireland, 1994.

[81] Asad Sayeed, Tamer Elsayed, Nikesh Garera, David Alexander, Tan Xu, Douglas W. Oard, David Yarowsky, and Christine Piatko. Arabic cross-document coreference detection. In *ACL-IJNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language*, Suntec, Singapore, August 2009.

[82] Hinrich Schutze. Automatic word sense discrimination. *Computational Linguistics. Special issue on word sense disambiguation*, 24(1):97–123, 1998.

[83] Michael F. Schwartz and David C. Wood. Discovering shared interests among people using graph analysis of global electronic mail traffic. *Communications of the ACM*, 36:78–89, 1992.

[84] Lokesh Shrestha and Kathleen McKeown. Detection of question-answer pairs in email conversations. In *COLING '04: Proceedings of the 20th International Conference on Computational Linguistics*, page 889, Geneva, Switzerland, August 2004.

[85] Amit Singhal and Fernando Pereira. Document expansion for speech retrieval. In *SIGIR '99: Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 34–41, Berkeley, California, United States, 1999.

[86] Ellen M. Voorhees. The philosophy of information retrieval evaluation. In *CLEF '01: Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, pages 355–370, London, United Kingdom, 2002. Springer-Verlag.

[87] W3C. The W3C test collection. http://research.microsoft.com/users/nickcr/w3c-summary.html.

[88] Wouter Weerkamp, Krisztian Balog, and Maarten Rijke. Using contextual information to improve search in email archives. In *ECIR '09: Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, pages 400–411, Berlin, Heidelberg, 2009. Springer-Verlag.

[89] Michael Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. An entity based model for coreference resolution. In *SDM '09: Proceedings of SIAM International Conference on Data Mining*, pages 365–376, Reno, Nevada, United States, April 2009.

[90] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[91] Yejun Wu, Douglas Oard, and Ian Soboroff. An exploratory study of the W3C mailing list test collection for retrieval of emails with pro/con arguments. In *CEAS'06: Proceedings of the third Conference on Email and Anti-Spam*, Mountain View, California, United States, July 2006.

[92] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. Map-Reduce-Merge: simplified relational data processing on large clusters. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1029–1040, Beijing, China, 2007.

[93] Jen-Yuan Yeh and Aaron Harnly. Email thread reassembly using similarity matching. In *CEAS '06: Proceedings of the thrird Conference on Email and Anti-Spam*, Mountain View, California, United States, July 2006.

[94] David M. Zajic, Bonnie J. Dorr, and Jimmy Lin. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing and Management*, 44(4):1600 – 1610, 2008.

[95] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New Orleans, Louisiana, United States, 2001.

[96] Jun Zhang and Mark S. Ackerman. Searching for expertise in social networks: a simulation of potential strategies. In *GROUP '05: Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*, pages 71–80, Sanibel Island, Florida, United States, November 2005.