Contextual Search and Name Disambiguation in Email Using Graphs

Einat Minkov
Language Technologies Inst.
Carnegie Mellon University
Pittburgh, PA 15213
einat@cs.cmu.edu

William W. Cohen
Machine Learning Dept.
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Andrew Y. Ng
Computer Science Dept.
Stanford University
Stanford, CA 94305
ang@cs.stanford.edu

ABSTRACT

Similarity measures for text have historically been an important tool for solving information retrieval problems. In many interesting settings, however, documents are often closely connected to other documents, as well as other non-textual objects: for instance, email messages are connected to other messages via header information. In this paper we consider extended similarity metrics for documents and other objects embedded in graphs, facilitated via a lazy graph walk. We provide a detailed instantiation of this framework for email data, where content, social networks and a timeline are integrated in a structural graph. The suggested framework is evaluated for two email-related problems: disambiguating names in email documents, and threading. We show that reranking schemes based on the graph-walk similarity measures often outperform baseline methods, and that further improvements can be obtained by use of appropriate learning methods.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models, Search process

General Terms

Algorithms, Experimentation

Keywords

graph-based retrieval, email, name disambiguation, threading

1. INTRODUCTION

Many tasks in information retrieval can be performed by clever application of textual similarity metrics: in addition to the canonical IR problem of *ad hoc* retrieval, which is often formulated as the task of finding documents "similar to" a query, textual similarity plays a prominent role in the literature for diverse tasks such as text categorization [32], data integration [6], summarization [28] and document segmentation [16].

In modern IR settings, however, documents are usually not isolated objects: instead, they are frequently connected to other objects, via hyperlinks or meta-data. (An email message, for instance, is connected via header information to other emails and also to the recipient's social network.) Thus it is important to understand how text-based document similarity measures can be extended to documents embedded in complex structural settings.

Our similarity metric is based on a lazy graph walk, and is closely related to the well-known PageRank algorithm [25]. PageRank and its variants (e.g., [14]) are based on a graph walk of infinite length with random resets. In a *lazy* graph walk, there is a fixed probability of halting the walk at each step. In previous work [30], lazy walks over graphs were used for estimating word dependency distributions: in this case, the graph was one constructed especially for this task, and the edges in the graph represented different flavors of word-to-word similarity. Other recent papers have also used walks over graphs for query expansion [31, 11]. In these tasks, the walk propagates similarity to a start node through edges in the graph—incidentally accumulating evidence of similarity over multiple connecting paths.

In contrast to this previous work, we consider schemes for propogating similarity across a graph that naturally models a structured dataset like an email corpus: entities correspond to objects including email addresses and dates, (as well as the usual types of documents and terms), and edges correspond to relations like *sent-by*. We view the similarity metric as a *tool for performing search* across this structured dataset, in which related entities that are not directly similar to a query can be reached via a multi-step graph walk.

In this paper, we formulate and evaluate this extended similarity metric. The principal problem we consider is *disambiguating personal names in email*, which we formulate as the task of retrieving the person most related to a particular name mention. We show that for this task, the graph-based approach improves substantially over plausible baselines. After retrieval, learning can be used to adjust the ranking of retrieved names based on the edges in the paths traversed to find these names, which leads to an additional performance improvement. As a demonstration of generality, we also show performance improvements on a second email-related task—recovering messages from the same email thread. Name disambiguation and email threading are particular applications of the suggested general framework,

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

which is also applicable to any real-world setting in which structural data is available as well as text.

This paper proceeds as follows. Sections 2 and 3 formalize the general framework and its instantiation for email. Section 4 gives a short summary of the learning approach. Section 5 includes experimental evaluation, describing the corpora and results for the person name disambiguation as well as threading tasks. The paper concludes with a review of related work, summary and future directions.

2. EMAIL AS A GRAPH

A graph G consists of a set of nodes, and a set of labeled directed edges. Nodes will be denoted by letters such as x, y, or z, and we will denote an edge from x to y with label ℓ as $x \xrightarrow{\ell} y$. Every node x has a type, denoted $T(x)$, and we will assume that there is a fixed set of possible types. We will assume for convenience that there are no edges from a node to itself (this assumption can be easily relaxed).

We will use these graphs to represent real-world data. Each node represents some real-world entity, and each edge $x \xrightarrow{\ell} y$ asserts that some binary relation $\ell(x, y)$ holds. The entity types used here to represent an email corpus are shown in the leftmost column of Table 1. They include the traditional types in information retrieval systems, namely *file* and *term*. In addition, however, they include the types *person*, *email-address* and *date*. These entities are constructed from a collection of email messages in the obvious way—for example, a recipient of "Einat Minkov <einat@cs.cmu.edu>" indicates the existence of a person node "Einat Minkov" and an email-address node "einat@cs.cmu.edu". (We assume here that person names are unique identifiers.)

The graph edges are directed. We will assume that edge labels determine the source and target node types: i.e., if $x \xrightarrow{\ell} z$ and $w \xrightarrow{\ell} y$ then $T(w) = T(x)$ and $T(y) = T(z)$. However, multiple relations can hold between any particular pair of nodes types: for instance, it could be that $x \xrightarrow{\ell} y$ or $x \xrightarrow{\ell'} y$, where $\ell \neq \ell'$. (For instance, an email message x could be *sent-from* y, or *sent-to* y.) Note also that edges need not denote functional relations: for a given x and ℓ, there may be many distinct nodes y such that $x \xrightarrow{\ell} y$. For instance, for a file x, there are many distinct terms y such that $x \xrightarrow{has\text{-}term} y$ holds.

In representing email, we also create an *inverse label* ℓ^{-1} for each edge label (relation) ℓ. Note that this means that the graph will definitely be cyclic. Table 1 gives the full set of relations used in our email represention scheme.

source type	edge type	target type
file	sent-from	*person*
	sent-from-email	*email-address*
	sent-to	*person*
	sent-to-email	*email-address*
	date-of	*date*
	has-subject-term	*term*
	has-term	*term*
person	sent-from^{-1}	*file*
	sent-to^{-1}	*file*
	alias	*email-address*
	includes-term	*term*
email-address	sent-to-email^{-1}	*file*
	sent-from-email^{-1}	*file*
	alias^{-1}	*person*
	is-email^{-1}	*term*
term	has-subject-term^{-1}	*file*
	has-term^{-1}	*file*
	is-email	*email-address*
	includes-term^{-1}	*person*
date	date-of^{-1}	*file*

Table 1: Graph structure: Node and relation types

3. GRAPH SIMILARITY

3.1 Edge weights

Similarity between two nodes is defined by a lazy walk process, and a walk on the graph is controlled by a small set of parameters Θ. To walk away from a node x, one first picks an edge label ℓ; then, given ℓ, one picks a node y such that $x \xrightarrow{\ell} y$. We assume that the probability of picking the label ℓ depends only on the type $T(x)$ of the node x, i.e., that the outgoing probability from node x of following an edge type ℓ is:

$$Pr(\ell \mid x) = Pr(l \mid T_i) \equiv \theta_{\ell,T_i}$$

Let S_{T_i} be the set of possible labels for an edge leaving a node of type T_i. We require that the weights over all outgoing edge types given the source node type form a probability distribution, i.e., that

$$\sum_{\ell \in S_{T_i}} \theta_{\ell,T_i} = 1$$

In this paper, we will assume that once ℓ is picked, y is chosen uniformly from the set of all y such that $x \xrightarrow{\ell} y$. That is, the weight of an edge of type l connecting source node x to node y is:[1]

$$Pr(x \xrightarrow{\ell} y \mid \ell) = \frac{\theta_{\ell,T_i}}{\mid y : x \xrightarrow{\ell} y \mid}$$

This assumption could easily be generalized, however: for instance, for the type $T(x) = \textit{file}$ and $\ell = \textit{has-term}$, weights for terms y such that $x \xrightarrow{\ell} y$ might be distributed according to an appropriate language model [12].

3.2 Graph walks

Conceptually, the edge weights above define the probability of moving from a node x to some other node y. At each step in a lazy graph walk, there is also some probability γ of staying at x. Putting these together, and denoting by $\mathbf{M}_{xy}$ the probability of being at node y at time $t+1$ given that one is at x at time t in the walk, we define

$$\mathbf{M}_{xy} = \begin{cases} (1-\gamma)\sum_\ell Pr(x \xrightarrow{\ell} y|\ell) \cdot Pr(\ell|T(x)) & \text{if } x \neq y \\ \gamma & \text{if } x = y \end{cases}$$

If we associate nodes with integers, and make $\mathbf{M}$ a matrix indexed by nodes, then a walk of k steps can then be defined by matrix multiplication: specifically, if V_0 is some initial probability distribution over nodes, then the distribution after a k-step walk is proportional to $V_k = V_0\mathbf{M}^k$. Larger values of γ increase the weight given to shorter paths between x and y. In the experiments reported here, we consider small values of k, and this computation is carried out

[1] If no such y exists, then the outgoing probability mass θ_{ℓ,T_i} associated with node x is absorbed into a "null" state.

directly using sparse-matrix multiplication methods.[2] If V_0 gives probability 1 to some node x_0 and probability 0 to all other nodes, then the value given to y in V_k can be interpreted as a similarity measure between x and y.

In our framework, a *query* is an initial distribution V_q over nodes, plus a desired output type T_{out}, and the answer is a list of nodes y of type T_{out}, ranked by their score in the distribution V_k. For instance, for an ordinary *ad hoc* document retrieval query (like "economic impact of recycling tires") would be an appropriate distribution V_q over query terms, with $T_{out} = \textit{file}$. Replacing T_{out} with *person* would find the person most related to the query—e.g., an email contact heavily associated with the retread economics. Replacing V_q with a point distribution over a particular document would find the people most closely associated with the given document.

3.3 Relation to TF-IDF

It is interesting to view this framework in comparison to a more traditional IR setting, which can be viewed as a special case. Suppose we restrict ourselves to only two types, terms and files, and allow only *in-file* edges. Now consider an initial query distribution V_q which is uniform over the two terms "the aardvark". A one-step matrix multiplication will result in a distribution V_1, which includes file nodes. The common term "the" will spread its probability mass into small fractions over many file nodes, while the unusual term "aardvark" will spread its weight over only a few files: hence the effect will be similar to use of an IDF weighting scheme.

4. LEARNING

As suggested by the comments above, the described graph framework could be used for many types of tasks, and it is unlikely that a single set of parameter values θ will be best for all tasks. It is thus important to consider the problem of *learning* how to better rank graph nodes.

Previous researchers have described schemes for adjusting the parameters θ using gradient descent-like methods [14, 24]. In this paper, we suggest an alternative approach of learning to re-order an initial ranking. This reranking approach has been used in the past for meta-search [8] and also several natural-language related tasks [10, 9]. The advantage of reranking over parameter tuning is that the learned classifier can take advantage of "global" features that are not easily used in a walk.

Note however that node reranking, while can be used as an alternative to weight manipulation, it is better viewed as a complementary approach, as the techniques can be naturally combined by first tuning the parameters θ, and then reranking the result using a classifier which exploits non-local features. This hybrid approach has been used successfully in the past on tasks like parsing [10].

We here give a short overview of the reranking approach, which is described in more detail elsewhere [10]. The reranking algorithm is provided with a training set containing n examples. Example i (for $1 \le i \le n$) includes a ranked list of l_i nodes. Let w_{ij} be the jth node for example i, and let $p(w_{ij})$ be the probability assigned to w_{ij} by the graph walk.

[2]We have also explored an alternative approach based on sampling; this method scales better but introduces some additional variance into the procedure, which is undesirable for experimentation.

A candidate node w_{ij} is represented through m features, which are computed by m feature functions $f_1, \ldots, f_m$. We will require that the features be binary; this restriction allows a closed form parameter update [10]. The *ranking function* for node x is defined as:

$$F(x, \bar{\alpha}) = \alpha_0 L(x) + \sum_{k=1}^{m} \alpha_k f_k(x)$$

where $L(x) = \log(p(x))$ and $\bar{\alpha}$ is a vector of real-valued parameters. Given a new test example, the output of the model is the given node list re-ranked by $F(x, \bar{\alpha})$.

To learn the parameter weights $\bar{\alpha}$, we use a boosting method [10], which minimizes the following loss function on the training data:

$$ExpLoss(\bar{\alpha}) = \sum_i \sum_{j=2}^{l_i} e^{-(F(x_{i,1},\bar{\alpha}) - F(x_{i,j},\bar{\alpha}))}$$

where $x_{i,1}$ is, without loss of generality, the correct target node.[3] The weights for the function are learned with a boosting-like method, where in each iteration the feature f_k that has the most impact on the loss function is chosen, and α_k is modified. Closed form formulas exist for calculating the optimal additive parameter updates [10, 29].

5. EVALUATION

There are currently no available annotated email corpora for evaluation of email-related queries. In this paper we evaluate the system on two tasks: person name disambiguation, and email threading. The key property of these tasks is that a non-subjective correct answer set can be constructed per query. Each task was evaluated on three corpora.

5.1 Corpora

Each corpus is of moderate size—representative, we hope, of an ordinary user's collection of saved mail.

The **Cspace** corpus contains email messages collected from a management course conducted at Carnegie Mellon University in 1997 [23]. In this course, MBA students, organized in teams of four to six members, ran simulated companies in different market scenarios. The corpus we used here includes the emails of all teams over a period of four days, plus all messages that were replied to in the four-day period. This subcorpus is convenient for the task of name disambiguation for several reasons, which are outlined below.

The **Enron** corpus is a collection of mail from the Enron corpus that has been made available to the research community [19]. This corpus can be easily segmented by user: here, we used the saved email of four different users.[4] To eliminate spam and news postings we removed email files sent from email addresses with suffix ".com" that are not Enron's; widely distributed email files sent from addresses such as "enron.announcement" or "enron.chairman" at "enron.com"; and emails sent to "all.employees@enron.com" etc. Text from forwarded messages, or replied-to messages

[3]If there are $k > 1$ target nodes in a ranking, we split the ranking into k examples. Note also that it is possible to incorporate weights into this formula, e.g., to assign higher weight to nodes earlier in the ranking; however we assign all nodes equal importance.

[4]Specifially, we used the "all_documents" folder, including both incoming and outgoing files.

were also removed from the corpus. In deriving terms for the graph, terms were Porter-stemmed and stop words were removed.

Table 2 (leftmost columns) gives the size of each processed corpus, and the number of nodes in the graph representation of it. The processed Enron-derived corpora are available from the first author's home page.[5]

	corpus		Person set		Thread set	
	files	**nodes**	train	test	train	test
Cspace	821	6248	26	80	30	100
Sager-E	1632	9753	11	51	-	-
Shapiro-R	978	13174	11	49	-	-
Germany-C	2657	12730	-	-	27	54
Farmer-D	2548	14082	-	-	29	98

Table 2: Corpora Details

5.2 Person Name Disambiguation

5.2.1 *Task definition*

Consider an email message containing a common name like "Andrew". Ideally an intelligent mailer would, like the user, understand which person "Andrew" refers to, and would rapidly perform tasks like retrieving Andrew's preferred email address or home page. Resolving the referent of a person name is also an important complement to the ability to perform named entity extraction for tasks like social network analysis or studies of social interaction in email.

However, while the referent of the name is usually unambiguous to the recipient of the email, it can be non-trivial for an automated system to find out which "Andrew" is indicated. Automatically determining that "Andrew" refers to "Andrew Y. Ng" and not "Andrew McCallum" (for instance) is especially difficult when an informal nickname is used, or when the mentioned person does not appear in the email header. As noted above, we model this problem as a search task: based on a name-mention in an email message m, we formulate query distribution V_q, and then retrieve a ranked list of *person* nodes.

5.2.2 *Data preparation*

Unfortunately, building a corpus for evaluating this task is non-trivial, because (if trivial cases are eliminated) determining a name's referent is often non-trivial for a human other than the intended recipient. We evaluated this task using three labeled datasets, as detailed in Table 2.

The Cspace corpus has been manually annotated with personal names [23]. Additionally, with the corpus, there is a great deal of information available about the composition of the individual teams, the way the teams interact, and the full names of the team members. Using this extra information it is possible to manually resolve name mentions. We collected 106 cases in which single-token names were mentioned in the the body of a message but did not match any name from the header. Instances for which there was not sufficient information to determine a unique person entity were excluded from the example set. In addition to names that refer to people that are simply not in the header, the names in this corpus include people that are in the email header, but cannot be matched because they are referred to using: *initials*–this is commonly done in the sign-off to an email; *nicknames*, including common nicknames (e.g., "Dave" for "David"), and unusual nicknames (e.g., "Kai" for "Keiko"); or American names that were adopted by persons with foreign-language names (e.g., "Jenny" for "Qing").

For Enron, two datasets were generated automatically. We collected name mentions which correspond uniquely to names that are in the email "Cc" header line; then, to simulate a non-trivial matching task, we eliminate the collected person name from the email header. We also used a small dictionary of 16 common American nicknames to identify nicknames that mapped uniquely to full person names on the "Cc" header line.

Table 3 gives the distribution of name mention types for all datasets. For each dataset, some examples were picked randomly and set aside for learning and evaluation purposes (see Table 2).

	initials	**nicknames**	**other**
Cspace	11.3%	54.7%	34.0%
Sager-E	-	10.2%	89.8%
Shapiro-R	-	15.0%	85.0%

Table 3: Person Name Disambiguation Datasets

5.3 Results for person name disambiguation

5.3.1 *Evaluation details*

All of the methods applied generate a ranked list of person nodes, where there is exactly one correct answer per example.[6] Figure 1 gives results[7] for two of the datasets as a function of recall at rank k, up to rank 10. Table 4 shows the mean average precision (MAP) of the ranked lists as well as *accuracy*, which we define as the percentage of correct answers at rank 1 (i.e., precision at rank 1).

5.3.2 *Baseline method*

To our knowledge, there are no previously reported experiments for this task on email data. As a baseline, we apply a reasonably sophisticated string matching method [7]. Each name mention in question is matched against all of the person names in the corpus. The similarity score between the name term and a person name is calculated as the maximal Jaro similarity score [7] between the term and any single token of the personal name (ranging between 0 to 1). In addition, we incorporate a nickname dictionary,[8] such that if the name term is a known nickname of the person name, the similarity score of that pair is set to 1.

The results are shown in Figure 1 and Table 4. As can be seen, the baseline approach is substantially less effective for the more informal Cspace dataset. Recall that the Cspace corpus includes many cases such as initials, and also nicknames that have no literal resemblance to the person's name (section 5.2.2), which are not handled well by the string similarity approach. For the Enron datasets, the baseline approach perfoms generally better (Table 4). In all the corpora there are many ambiguous instances, e.g., common names like "Dave" or "Andy" that match many people with equal strength.

[5]Unfortunately, due to privacy issues, the CSpace corpus can not be distributed in this way.

[6]If a ranking contains a block of items with the same score, a node's rank is counted as the average rank of the "block".

[7]Results refer to test examples only.

[8]The same dictionary that was used for dataset generation.

5.3.3 Graph walk methods

We perform two variants of graph walk, corresponding to different methods of forming the query distribution V_q. Unless otherwise stated, we will use a uniform weighting of labels—i.e., $\theta_{\ell,T} = 1/S_T$; $\gamma = 1/2$; and a walk of length 2.

In the first variant, we concentrate all the probability in the query distribution on the name term. The column labeled **term** gives the results of the graph walk from this probability vector. Intuitively, using this variant, the name term propagates its weight to the files in which it appears. Then, weight is propagated to person nodes which co-occur frequently with these files. Note that in our graph scheme there is a direct path between terms to person names, so that person nodes may recieve weight vis this path as well.

As can be seen in the results, this leads to very effective performance: e.g., it leads to 61.3% vs. 41.3% accuracy for the baseline approach on the CSpace dataset. However, it does not handle ambiguous terms as well as one would like, as the query does not include any information of the *context* in which the name occurred: the top-ranked answer for ambiguous name terms (e.g., "Dave") will always be the same person. To solve this problem, we also used a **file+term** walk, in which the query V_q gives equal weight to the name term node and the file in which it appears.

We found that adding the file node to V_q provides useful context for ambiguous instances—e.g., the correct "David" would in general be ranked higher than other persons with this same name. On the other hand, though, adding the file node reduces the the contribution of the term node. Although the MAP and accuracy are decreased, file+term has better performance than term at higher recall levels, as can be seen in Figure 1.

5.3.4 Reranking the output of a walk

We now examine reranking as a technique for improving the results. We formed the following types of features f for a node x. *Edge unigram features* indicate, for each edge label ℓ, whether ℓ was used in reaching x from V_q. *Edge bigram features* indicate, for each pair of edge labels ℓ_1, ℓ_2, whether ℓ_1 and ℓ_2 were used (in that order) in reaching x from V_q. *Top edge bigram features* are similar but indicate if ℓ_1, ℓ_2 were used in one of the two highest-scoring paths between V_q and x (where the "score" of a path is the product of $\Pr(y \xrightarrow{\ell} z)$ for all edges in the path).

We believe that these features could all be computed using dynamic programming methods. Currently, however, we compute features by using a method we call *path unfolding*, which is similar to the *back-propagation through time* algorithm [15, 14] used in training recurrent neural networks. Graph unfolding is based on a backward breadth-first visit of the graph, starting at the target node at time step k, and expanding the unfolded paths by one layer per each time step. This procedure is more expensive, but offers more flexibility in choosing alternative features, and was useful in determining an optimal feature set.

In addition, we used for this task some additional problem-specific features. One new feature indicates whether the set of paths leading to a node originate from one or two nodes in V_q. (We conjecture that in the file+term walk, nodes that are connected to both the source term and file nodes are more relevant than nodes that are connected to only the file node or only the term node.) We also form features that indicate whether the given term is a nickname of the person

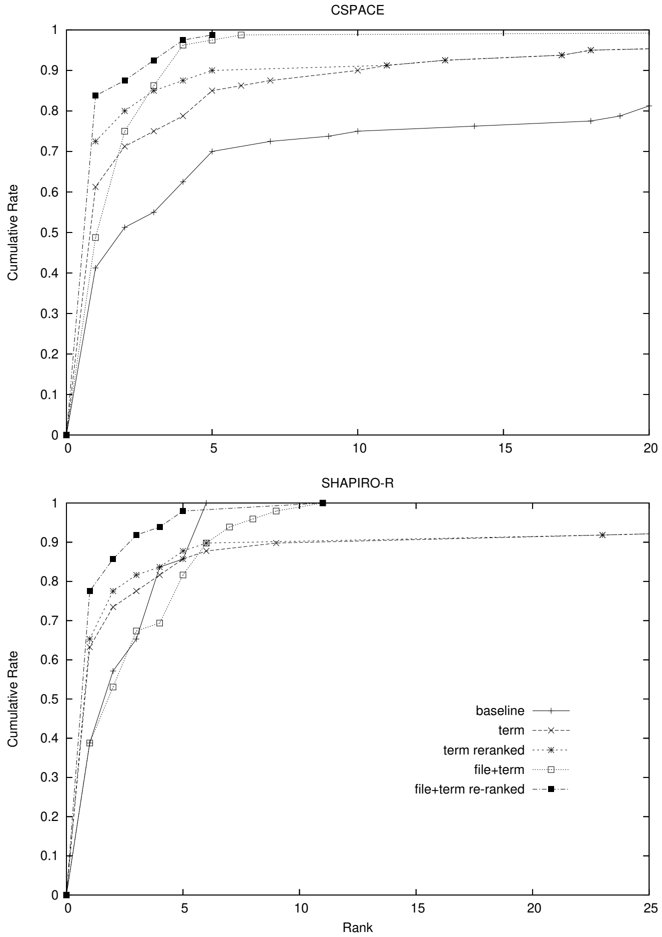

Figure 1: Person name disambiguation results: Recall at rank k

name, per the nicknames dictionary; and whether the Jaro similarity score between the term and the person name is above 0.8. This information is similar to that used by the baseline ranking system.

The results (for the test set, after training on the train set) are shown in Table 4 and (for two representative cases) Figure 1. In each case the top 10 nodes were reranked. Reranking substantially improves performance, especially for the file+term walk. The accuracy rate is higher than 75% across all datasets. The features that were assigned the highest weights by the re-ranker were the literal similarity features and the *source count* feature.

5.4 Threading

5.4.1 Task Description

As a test of the generality of our approach, we also considered a second task. *Threading* is the problem of retrieving other messages in an email thread given a single message from the thread. Threading is a well known task for email, although there are only few relevant works published [21, 27]. As has been pointed out before [21], users make inconsistent use of the "reply" mechanism, and there are

	MAP	Δ_{MAP}	Acc.	Δ_{Acc}
Cspace				
Baseline	49.0	-	41.3	-
Graph - term	72.6	48.2%	61.3	48.4%
Graph - file+term	66.3	35.3%	48.8	18.2%
Reranking - term	85.6	74.7%	72.5	75.5%
Reranking - file+term	**89.0**	**81.6%**	**83.8**	**102.9%**
Sager-E				
Baseline	67.5	-	39.2	-
Graph - term	82.8	22.7%	66.7	70.2%
Graph - file+term	61.7	8.6%	41.2	5.1%
Reranking - term	83.2	23.3%	68.6	75.0%
Reranking - file+term	**88.9**	**31.7%**	**80.4**	**105.1%**
Shapiro-R				
Baseline	60.8	-	38.8	-
Graph - term	84.1	38.3%	63.3	65.3%
Graph - file+term	56.5	(7.1%)	38.8	1.3%
Reranking - term	**87.9**	**44.6%**	65.3	70.5%
Reranking - file+term	85.5	40.6%	**77.6**	**102.6%**

Table 4: Person Name Disambiguation Results

frequent irregularities in the structural information that indicates threads; thus, thread discourse arguably should be captured using an intelligent approach. It has also been suggested [19] that once obtained, thread information can improve message categorization into topical folders.

Our primary interest in this task is that threading is an easily-evaluated proxy for the task of finding similar messages in a corpus. Finding related messages would be both a useful operation for users, and is also important for automatic email processing at the corpus level. As threads (and more generally, similar messages) are indicated by multiple types of relations including text, social network information, and timing information, we expect this task to benefit from the graph framework.

More precisely, we formulate threading as follows: given an email file as a query, produce a ranked list of related email files, where the immediate parent and child of the given file are considered to be "correct" answers. We limit the answer set to the adjacent files because of our more general interest in finding related messages: while consecutive thread messages can be assumed to be related to each other, this assumption is weaker if applied on the entire thread. This definition does, however, make the task somewhat more challenging.

5.4.2 Data

We created three datasets for task evaluation, again from the Cspace and Enron corpora. The number of queries for each dataset are given in Table 2. For each relevant message, its parent was identified by using the subject line and time stamp. About 10-20% of the messages have both parent and child messages available, otherwise only one file in the thread is a correct answer.

We used a series of variants of this data, in which we varied the amount of message information that is available. Specifically, several information types are available in these corpora: the email *header*,including sender, recipients and date; the *body*, i.e., the textual content of an email, excluding any quoted reply lines or attachements from previous messages; *reply lines*, i.e., quoted lines from previous messages; and *the subject*, i.e., the content of the subject line.

We compared several combinations of these components, as detailed in Table 5. Of particular interest is the task which considers header and body information alone, since it best reflects the situation for the more general task of finding "related" messages.

5.4.3 Baseline method

The baseline approach generates a list of files, ranked by similarity scores using the vector space model, in which a document is represented as a weighted vector in a term space and a document similarity score is the cosine similarity of their vectors. TF-IDF term weighting is commonly used for document representation; to apply the TF-IDF scheme here, we simply consider all available information as text.

The results (Table 5) show that this approach performs reasonably well. Due to space limitations full results are given as MAP scores; in addition, Figure 2 shows the recall-at-k curve for the Cspace dataset, using header and text. Recall of about 55% at rank 5 is reached using header and text information for Cspace using the baseline approach. As one might expect, adding information, in particular the subject and reply lines, improves performance substantially.

5.4.4 Graph walk methods

To formulate this as a problem in the graph model, we let V_q assign probability 1 to the *file* node corresponding to the original message, and let T_{out} = *file*. In addition to using uniform graph weights, we also use an extremely simple weight-tuning method: specifically, we evaluated 10 randomly-chosen sets of weights and pick the one that performs best (in terms of MAP) on the CSpace training data. We repeated this procedure separately for every experiment setting, so a total of four "random" weight vectors were used. Performance for this weight set is shown as "Graph-Random" in the table.

The results show that the graph walk and the TF-IDF are comparable when identical chunks of text, such as subject lines, are present in both the query message and the "target". However, the graph walk performs better using only header and body text information, with an absolute improvement of 4.1% to 24.7% in MAP across corpora. Note that the "random" weights outperform uniform weights and TF-IDF substantially on CSpace, and often also on other corpora. This is especially true when reply and subject lines are not available. This suggests that even very simple weight-tuning methods are likely to improve performance.

5.4.5 Reranking the output of walks

We applied reranking on top of the random-weighted graph walk results. The top 50 file nodes were given to the reranker. The features applied are *edge unigram*, *edge bigram* and *top edge bigram* (described in section 5.3.4). We found that the edge bigram features are most informative, leading to large improvement rates. Overall, reranking the graph walk almost always yields the best results.[9] For example, recall of 75% at rank 5 is achieved for the CSpace dataset, with only header and text available, compared to 58% using the TF-IDF based method and 54% prior to reranking. Most features that were assigned high weight by the learner were bigrams: some examples are: *sent from* $\rightarrow$ *sent to*$^{-1}$, *date*

[9] Performance degraded in only one of the ten cases, for Farmer-D dataset using header only. This is probably due to over-fitting; performance improved slightly on the train set.

header	$\surd$	$\surd$	$\surd$	$\surd$
body	$\surd$	$\surd$	$\surd$	-
subject	$\surd$	$\surd$	-	-
reply lines	$\surd$	-	-	-
Cspace				
TF-IDF	58.4	50.2	36.2	43.7
Graph - Uniform	59.6	54.8	40.2	40.6
Graph - Random	61.9	62.1	49.3	46.8
Graph - Reranked	**73.8**	**71.5**	**60.3**	**55.0**
Germany-C				
TF-IDF	-	58.2	37.5	28.1
Graph - Uniform	-	50.5	41.6	33.9
Graph - Random	-	49.7	41.6	46.7
Graph - Reranked	-	**68.6**	**49.1**	**57.5**
Farmer-D				
TF-IDF	-	65.7	36.1	30.8
Graph - Uniform	-	66.7	57.1	50.3
Graph - Random	-	63.8	60.8	**56.6**
Graph - Reranked	-	**79.8**	**65.1**	48.4

Table 5: Threading Results: MAP

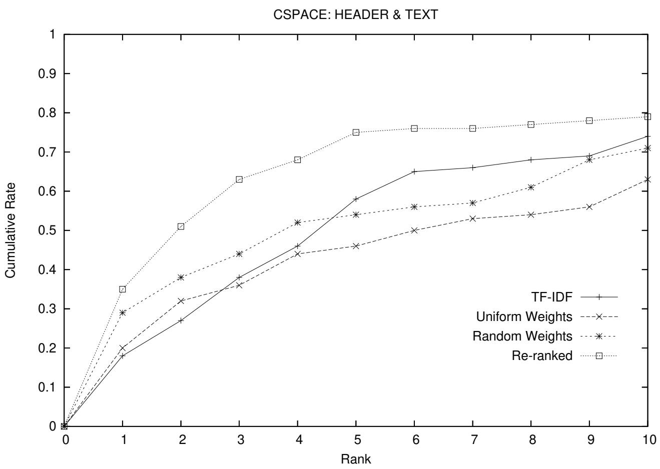

Figure 2: Threading results: Recall at rank k

of $\rightarrow$ *date of*$^{-1}$, and *has term* $\rightarrow$ *has term*$^{-1}$. These paths are indeed characteristic of a thread: e.g., the sender of a message is likely to be a recipient of a reply message, there is high temporal proximity between messages in a thread, and some textual overlap.

Note that while such sequences of relations can be readily identified as important in our framework, they cannot be even modeled easily in a flat representation. Sequential aspects of a corpora have been shown to be important for other email-related tasks, e.g., workflows and social interaction [5].

6. RELATED WORK

As noted above, the similarity measure we use is based on graph-walk techniques which have been adopted by many other researchers for several different tasks. In the information retrieval community, infinite graph walks are prevalent for determining document centrality (e.g., [25, 14, 20]). Another related line of research is of *spreading activation* over semantic or association networks: here the underlying idea is to propagate "activation" from source nodes via weighted links through the network (e.g., [4, 26]). Spreading activation methods are parameterized by user-provided threshold functions for node activation, limits on node distance, preferences over paths, and other constraints. The framework presented here is similar, but operates through unconstrained lazy graph walks, where path preferences are learned from data.

The idea of representing structured data as a graph is widespread in the data mining community, which is mostly concerned with relational or semi-structured data. Recently, the idea of PageRank has been applied to keyword search in structured databases [2]. Analysis of inter-object relationships has been suggested for entity disambiguation for entities in a graph [18], where the graph edges are undirected and edge weights represent confidence in having a connecting path between the entities. It has been suggested to model similarity between objects in relational data in terms of structural-context similarity [17], where the similarity measure corresponds to the expected number of steps required for a random surfer to cross the graph from one object to the other. The latter did not consider edge weights.

In this paper we propose the use of learned re-ranking schemes to improve performance of a lazy graph walk. Earlier authors have considered instead using hill-climbing approaches to adjust the parameters of a graph-walk [14]. We have not compared directly with such approaches; it may be that the performance gain of such methods is limited, due to their inability to exploit the global features we used for these tasks. In preliminary experiments, reranking using a set of simple locally-computable features only modestly improved performance of the "random" weight set for the CSpace threading task. Another related line of research explores random walks for semi supervised learning [34, 33].

As mentioned earlier, not much work has been done that integrates meta-data and text in email. One example examines clustering using multiple types of interactions in cooccurence data [3]. Another recent paper [1] proposes a graph-based approach for email classification. They represent an individual email message as a structured graph representing both content and header, and find a graph profile for each folder; incoming messages are classified into folders using graph matching techniques.

The task of person disambiguation has been studied in the field of social networks and applied also to email data (e.g., [22, 13]). In particular, it has been recently suggested to perform name disambiguation in email using traffic information, as derived from the email headers [13]. Our approach differs in that it allows integration of email content and a timeline in addition to social network information in a unified framework. In addition, we use learning to tune the system parameters automatically.

7. CONCLUSION

We have presented a scheme for representing a corpus of email messages with a graph of typed entities, and an extension of the traditional notions of document similarity to documents embedded in a graph. This scheme provides good performance on two representative email-related tasks: disambiguating person names, and email threading. Using a boosting-based learning scheme to rerank outputs based on graph-walk related features provides an additional performance improvement. The final results are quite strong: for name disambiguation, the method yields MAP scores in the mid-to-upper 80's; and for threading, it produces substantial gains over a TF-IDF baseline. The person name identification task illustrates a key advantage of our approach—that context can be easily incorporated in entity disambiguation.

In future work, we plan to further explore the scalability of the approach, and also ways of integrating this approach with language-modeling approaches for document representation and document retrieval. A newer version of this system (not the one used in the experiments) uses a sampling-based approximation to iterative matrix multiplication. In preliminary timing experiments, the new system can very accurately approximate walks of the sort considered here in around 0.5 seconds, and can approximate 10-step walks on a million-node corpus (from a different domain) in around 10-15 seconds.

There are several strong motivations for using this framework for search-related tasks in email. First, preserving entity type allows one to formulate a broad range of problems as typed search queries—including, in this paper, name disambiguation and threading. Secondly, structural relation modeling provides a unified framework for integration of multiple types of information, including social networks, text, timelines,[10] and other information such as organization charts. With such additional information, many interesting information management tasks can be formulated as (or facilitated by) typed retrieval: for instance, retrieval of email addresses related to calendar entries could facilitate meeting rescheduling.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

9. REFERENCES

[1] M. Aery and S. Chakravarthy. EmailSift: Email classification based on structure and content. In *ICDM*, 2005.

[2] A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-based keyword search in databases. In *VLDB*, 2004.

[3] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *ICML*, 2005.

[4] H. Berger, M. Dittenbach, and D. Merkl. An adaptive information retrieval system. based on associative networks. In *APCCM*, 2004.

[5] V. R. Carvalho and W. W. Cohen. On the collective classification of email "speech acts". In *SIGIR*, 2005.

[6] W. W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, 2000.

[7] W. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWEB*, 2003.

[8] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research (JAIR)*, 10:243–270, 1999.

[10] In future work we plan to incorporate additional time information by adding edges between dates nodes.

[9] M. Collins. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *ACL*, 2002.

[10] M. Collins and T. Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69, 2005.

[11] K. Collins-Thompson and J. Callan. Query expansion using random walk models. In *CIKM*, 2005.

[12] W. B. Croft and J. Lafferty. *Language Modeling for Information Retrieval*. Springer, 2003.

[13] C. P. Diehl, L. Getoor, and G. Namata. Name reference resolution in organizational email archives. In *SIAM*, 2006.

[14] M. Diligenti, M. Gori, and M. Maggini. Learning web page scores by error back-propagation. In *IJCAI*, 2005.

[15] S. Haykin. *Neural Networks*. Macmillan College Publishing Company, 1994.

[16] M. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, 1997.

[17] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *SIGKDD*, 2002.

[18] D. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationship for domain independent data cleaning. In *SIAM*, 2005.

[19] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *ECML*, 2004.

[20] O. Kurland and L. Lee. Pagerank without hyperlinks: Structural re-ranking using links induced by language models. In *SIGIR*, 2005.

[21] D. E. Lewis and K. A. Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 1997.

[22] B. Malin, E. M. Airoldi, and K. M. Carley. A social network analysis model for name disambiguation in lists. *Journal of Computational and Mathematical Organization Theory*, 11(2), 2005.

[23] E. Minkov, R. C. Wang, and W. W. Cohen. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT-EMNLP*, 2005.

[24] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: Bringing order to web objects. In *WWW*, 2005.

[25] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Technical Report, Computer Science department, Stanford University*, 1998.

[26] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information retrieval. In *SIGIR*, 1988.

[27] G. Salton and C. Buckley. Global text matching for information retrieval. *Science*, 253:1012–1015, 1991.

[28] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–208, 1997.

[29] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[30] K. Toutanova, C. D. Manning, and A. Y. Ng. Learning random walk models for inducing word dependency distributions. In *ICML*, 2004.

[31] W. Xi, E. A. Fox, W. P. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. Simfusion: Measuring similarity using unified relationship matrix. In *SIGIR*, 2005.

[32] Y. Yang and C. Chute. An example-based mapping method for text classification and retrieval. *ACM Transactions on Information Systems*, 12(3), 1994.

[33] D. Zhou, B. Scholkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *NIPS*, 2005.

[34] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.