A Graphical Framework for Contextual Search and Name Disambiguation in Email

Einat Minkov
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213
einatm@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 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 the task of disambiguating names in email documents. 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.

1 Introduction

Many tasks in information retrieval can be performed by clever application of textual similarity metrics. In particular, The canonical IR problem of *ad hoc* retrieval is often formulated as the task of finding documents "similar to" a query. 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 in the same thread 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 (Page et al., 1998). PageRank and its variants 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 (Toutanova et al., 2004), 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 (Xi et al., 2005; Collins-Thompson and Callan, 2005). 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 multi-step graph walk.

In this paper, we formulate and evaluate this extended similarity metric. The principal problem we

Workshop on TextGraphs, at HLT-NAACL 2006, pages 1–8,
New York City, June 2006.

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. Name disambiguation is a particular application of the suggested general framework, 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 task. 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 like 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 are 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 $x \xrightarrow{\textit{has-term}} y$ 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.

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:

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

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:

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

This assumption could easily be generalized, however: for instance, for the type $T(x) = \textit{file}$ and

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 inv.	*file*
	sent-to^{-1}	*file*
	alias	*email-address*
	has-term	*term*
email-address	sent-to-email^{-1}	*file*
	sent-from-email^{-1}	*file*
	alias-inverse	*person*
	is-email^{-1}	*term*
term	has-term^{-1}	*file*
	has subject-term^{-1}	*file*
	is-email	*email-address*
	has-term^{-1}	*person*
date	date-of^{-1}	*file*

Table 1: Graph structure: Node and relation types

$\ell =$ *has-term*, weights for terms y such that $x \xrightarrow{\ell} y$ might be distributed according to an appropriate language model (Croft and Lafferty, 2003).

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

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

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 directly using sparse-matrix multiplication methods.[1] 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 $T_{out} = \mathit{file}$. 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 more traditional IR methods. Suppose we restrict ourselves to 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, this graph framework could be used for many types of tasks, and it is unlikely that a single set of parameter values 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 (Diligenti et al., 2005; Nie et al., 2005). 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 (Cohen et al., 1999) and also several natural-

[1] 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.

language related tasks (Collins and Koo, 2005). The advantage of reranking over parameter tuning is that the learned classifier can take advantage of "global" features that are not easily used in walk.

Note 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 (Collins and Koo, 2005).

We here give a short overview of the reranking approach, that is described in detail elsewhere (Collins and Koo, 2005). The reranking algorithm is provided with a training set containing n examples. Example i (for $1 \leq i \leq n$) 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. 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. 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-value 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 (Collins and Koo, 2005), 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, a correct target node. 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 updates and the impact per feature (Schapire and Singer, 1999).

5 Evaluation

We experiment with three separate corpora.

The **Cspace** corpus contains email messages collected from a management course conducted at Carnegie Mellon University in 1997 (Minkov et al., 2005). 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. The **Enron** corpus is a collection of mail from the Enron corpus that has been made available for the research community (Klimt and Yang, 2004). Here, we used the saved email of two different users.[2] 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 "enron.announcement", to "all.employees@enron.com" etc.). Text from forwarded messages, or replied-to messages were also removed from the corpus.

Table 2 gives the size of each processed corpus, and the number of nodes in the graph representation of it. In deriving terms for the graph, terms were Porter-stemmed and stop words were removed. The processed Enron corpora are available from the first author's home page.

	corpus		Person set	
	files	**nodes**	train	test
Cspace	821	6248	26	80
Sager-E	1632	9753	11	51
Shapiro-R	978	13174	11	49

Table 2: Corpora Details

5.1 Person Name Disambiguation

5.1.1 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 prefered 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.

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

However, although the referent of the name is 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.

5.1.2 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 (Minkov et al., 2005). 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"), unusual nicknames (e.g., "Kai" for "Keiko"); or American names adopted in place of a foreign name (e.g., "Jenny" for "Qing").

For Enron, two datasets were generated automatically. We collected name mentions which correspond uniquely a names that is 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.

For each dataset, some examples were picked randomly and set aside for learning and evaluation purposes.

	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

5.2 Results for person name disambiguation

5.2.1 Evaluation details

All of the methods applied generate a ranked list of person nodes, and there is exactly one correct answer per example.[3] Figure 1 gives results[4] 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.)

5.2.2 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 (Cohen et al., 2003). 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 (Cohen et al., 2003) between the term and any single token of the personal name (ranging between 0 to 1). In addition, we incorporate a nickname dictionary[5], such that if the name term is a known nickname of a 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

[3] 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".

[4] Results refer to test examples only.

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

5.1.2), 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.

5.2.3 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 they recieve weight 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.

5.2.4 Reranking the output of a walk

We now examine reranking as a technique for improving the results. After some preliminary experimentation, we adopted the following types of features f for a node x. The set of features are fairly generic. *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 (Haykin, 1994; Diligenti et al., 2005) 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 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 are connected to both the source term and file nodes are more relevant comparing to nodes that are reached from the file node or term node only.) We also form features that indicate whether the given term is a nickname of the person 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 method.

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.

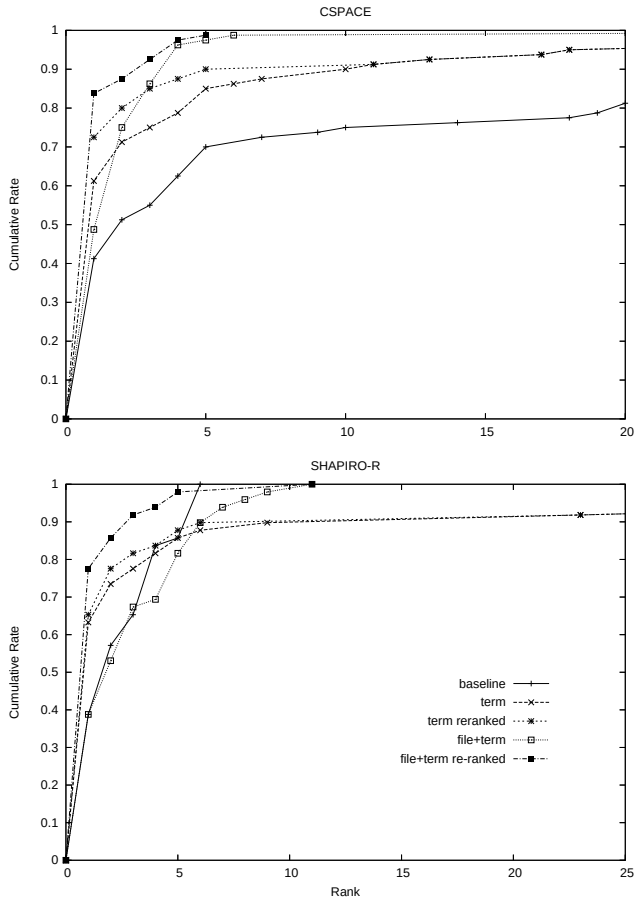

Figure 1: Person name disambiguation results: Recall at rank k

	MAP	Accuracy
Cspace		
Baseline	49.0	41.3
Graph - term	72.6	61.3
Graph - file+term	66.3	48.8
Reranking - term	85.6	72.5
Reranking - file+term	**89.0**	**83.8**
Sager-E		
Baseline	67.5	39.2
Graph - term	82.8	66.7
Graph - file+term	61.7	41.2
Reranking - term	83.2	68.6
Reranking - file+term	**88.9**	**80.4**
Shapiro-R		
Baseline	60.8	38.8
Graph - term	84.1	63.3
Graph - file+term	56.5	38.8
Reranking - term	**87.9**	65.3
Reranking - file+term	85.5	**77.6**

Table 4: Person Name Disambiguation Results

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., (Page et al., 1998; Diligenti et al., 2005; Kurland and Lee, 2005)). A related venue of research is of *spreading activation* over semantic or association networks, where the underlying idea is to propagate activation from source nodes via weighted links through the network (Berger et al., 2004; Salton and Buckley, 1988).

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 (Balmin et al., 2004). Analysis of inter-object relationships has been suggested for entity disambiguation for entities in a graph (Kalashnikov et al., 2005), where edges are unlabelled. It has been suggested to model similarity between objects in relational data in terms of structural-context similarity (Jeh and Widom, 2002).

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 (Diligenti et al., 2005). We have not compared directly with such approaches; preliminary experiments suggest that the performance gain of such methods is limited, due to their inability to exploit the global features we used here[6]. Related research explores random walks for semi supervised learning (Zhu et al., 2003; Zhou et al., 2005).

The task of person disambiguation has been studied in the field of social networks (e.g., (Malin et al., 2005)). In particular, it has been suggested to perform name disambiguation in email using traffic information, as derived from the email headers (Diehl et al., 2006). Our approach differs in that it allows integration of email content and a timeline in addition to social network information in a unified

[6]For instance, re-ranking using a set of simple locally-computable features only modestly improved performance of the "random" weight set for the CSpace threading task.

framework. In addition, we incorporate 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. Using a boosting-based learning scheme to rerank outputs based on graph-walk related, as well as other domain-specific, features provides an additional performance improvement. The final results are quite strong: for the explored name disambiguation task, the method yields MAP scores in the mid-to-upper 80's. 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 retrieval. An open question with regard to contextual (multi-source) graph walk in this framework is whether it is possible to further focus probability mass on nodes that are reached from multiple source nodes. This may prove beneficial for complex queries.

References

Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. 2004. ObjectRank: Authority-based keyword search in databases. In *VLDB*.

Helmut Berger, Michael Dittenbach, and Dieter Merkl. 2004. An adaptive information retrieval system. based on associative networks. In *APCCM*.

William W. Cohen, Robert E. Schapire, and Yoram Singer. 1999. Learning to order things. *Journal of Artificial Intelligence Research (JAIR)*, 10:243–270.

William W. Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *IIWEB*.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Kevyn Collins-Thompson and Jamie Callan. 2005. Query expansion using random walk models. In *CIKM*.

W. Bruce Croft and John Lafferty. 2003. *Language Modeling for Information Retrieval*. Springer.

Christopher P. Diehl, Lise Getoor, and Galileo Namata. 2006. Name reference resolution in organizational email archives. In *SIAM*.

Michelangelo Diligenti, Marco Gori, and Marco Maggini. 2005. Learning web page scores by error back-propagation. In *IJCAI*.

Simon Haykin. 1994. *Neural Networks*. Macmillan College Publishing Company.

Glen Jeh and Jennifer Widom. 2002. Simrank: A measure of structural-context similarity. In *SIGKDD*.

Dmitri Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen. 2005. Exploiting relationship for domain independent data cleaning. In *SIAM*.

Brown Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *ECML*.

Oren Kurland and Lillian Lee. 2005. Pagerank without hyperlinks: Structural re-ranking using links induced by language models. In *SIGIR*.

Bradely Malin, Edoardo M. Airoldi, and Kathleen M. Carley. 2005. A social network analysis model for name disambiguation in lists. *Journal of Computational and Mathematical Organization Theory*, 11(2).

Einat Minkov, Richard Wang, and William Cohen. 2005. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT-EMNLP*.

Zaiqing Nie, Yuanzhi Zhang, Ji-Rong Wen, and Wei-Ying Ma. 2005. Object-level ranking: Bringing order to web objects. In *WWW*.

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

Gerard Salton and Chris Buckley. 1988. On the use of spreading activation methods in automatic information retrieval. In *SIGIR*.

Robert E. Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML*.

Wensi Xi, Edward Allan Fox, Weiguo Patrick Fan, Benyu Zhang, Zheng Chen, Jun Yan, and Dong Zhuang. 2005. Simfusion: Measuring similarity using unified relationship matrix. In *SIGIR*.

Dengyong Zhou, Bernhard Scholkopf, and Thomas Hofmann. 2005. Semi-supervised learning on directed graphs. In *NIPS*.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.