Learning to Rank Typed Graph Walks: Local and Global Approaches

Einat Minkov
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA
einat@cs.cmu.edu

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

ABSTRACT

We consider the setting of lazy random graph walks over directed graphs, where entities are represented as nodes and typed edges represent the relations between them. This framework has been used in a variety of problems to derive an extended measure of entity similarity. In this paper we contrast two different approaches for applying supervised learning in this framework to improve graph walk performance: a gradient descent algorithm that tunes the transition probabilities of the graph, and a reranking approach that uses features describing global properties of the traversed paths. An empirical evaluation on a set of tasks from the domain of personal information management and multiple corpora show that reranking performance is usually superior to the local gradient descent algorithm, and that the methods often yield best results when combined.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Data Mining*; H.3.3 []: Information Search and Retrieval—*Retrieval Models, Search Process*

General Terms

Algorithms, Experimentation

Keywords

Entity relation graphs, learning, personal information management

1. INTRODUCTION

Relational structured data as well as semi-structured data is naturally represented by an entity-relation (ER) graph schema, where nodes represent entities and directed typed edges represent the relations between them. Such graphs are heterogenous, in the sense that they include different types of nodes, and different types of links. Examples of entity-relation graphs include citation networks where the nodes consist of authors, papers, institutions, journals, and conferences (e.g., [3]); these networks can be viewed also as rich social networks, where persons are interconnected via their relation to other entities, explicitly represented in the graph (e.g., persons may participate in the same *conferences*, coauthor *papers* etc.). Other examples of heterogenous social networks are the Internet Movie Database where nodes representing persons (actors, directors, etc.) are interconnected via a diverse set of relations to other entities such as movies and studios (e.g., [21]); in the domain of Homeland Security, affiliation networks that include diverse sets of persons and events interlinked in a variety of relationships are used for predicting threat vulnerabilities (e.g., [28]). Recently, it has been suggested to represent personal management information as entity-relation graphs, in which email messages, meeting entries, persons, email addresses, text and date nodes are interconnected via relations derived from the textual and structural information residing in a personal work station or a corporate database [19, 6].

Given an ER graph, a question of interest is how to determine the nature of relationship between two entities that are not directly connected in the graph (e.g., how to recognize the relationship *advisor-of* between person entities that are connected only indirectly, e.g., via links to coauthored paper entities). It has been suggested to construct a subgraph consisting of the shortest paths between two nodes for relationship detection [15]. Researchers have pointed out that the search for such extended semantic relations can be improved given knowledge about which link types in the graph are relevant for the relationship of interest [3].

In this paper we consider an extended similarity metric in entity-relation graphs based on lazy random graph walks. Given the networked data model and some initial distribution over graph nodes, a similarity (or relatedness) score of a node is defined as the probability of reaching that node in a random walk process [14, 22, 26, 27, 13, 20, 16]. This paradigm is closely related to the PageRank algorithm [23] and its "personalized" variants (e.g., [14]) that are based on a graph walk of infinite length with random resets. (In a "personalized" PageRank these resets are biased towards a distribution of iterest.) In a *lazy* graph walk, there is a fixed probability of halting the walk at each step. This halting probability means that short walks will be more probable, as it reduces the probability of reaching nodes distant from the starting points of the walk. Rather than modeling "centrality" of nodes (as in the PageRank algorithm), a lazy graph walk can be viewed as propagating "similarity" from

Joint 9th WEBKDD and 1st SNA-KDD Workshop '07 (WebKDD/SNA-KDD'07), August 12, 2007, San Jose, California, USA.
Copyright 2007 ACM 978-1-59593-848-0 ...$5.00.

a start node through edges in the graph—incidentally accumulating evidence of similarity over multiple connecting paths. This similarity metric can be viewed as a *tool for performing search* across the nodes in the graph. The result of the graph walk is a list of graph nodes, ranked by their probability mass cumulated in the graph walk process.

In a previous work [26] lazy random walks over graphs have been used for estimating word dependency distributions: in this case, the constructed graph represented different flavors of word-to-word similarity. Other papers have also used lazy walks over graphs for query expansion [27, 13]. Recently, this approach has been shown to be effective for contextual search and disambiguation in email processing tasks [20], as well as for protein name disambiguation [8].

Typically, the individual transitions in the random walk are determined by a set of parameters, assigning a fixed weight for each edge type (relation) in the graph, designating its relevancy or importance. As mentioned above, it is reasonable that a particular subset of link types be relevant for extraction of semantic relations between nodes that are not directly linked in the graph. Furthermore, different edge types may have varying importance in the context of different types of queries (e.g., recognizing different relationships like advisor-of versus spouse-of). This suggests the goal of learning the graph edge weights for a particular class of queries, such that probability propagated by the graph walk is directed to those nodes that demonstrate the relation of interest. Several schemes have been suggested for adjusting the set of edge weights using hill-climbing methods, which proved successful in personalized PageRank settings (see Section 2). A different approach suggested recently learns to re-order an initial ranking, using features describing the edges in the traversed graph paths [20, 8]. This method parameterizes the graph walk with a set of representative features, and thus loses some information; however, such features allow one to capture certain "global" properties of the graph walk. For example, they allow considering edge *sequences* that are encountered in traveling from the source nodes to a target node. The graph walks and the algorithms which tune the graph edge weights, on the other hand, are based on "local" information, where only information about the next immediate step of the walk is accessible. Given this qualitative difference, the focus of this paper is to conduct an empirical evaluation of the two approaches for a shared set of tasks, using authentic data. We will show that reranking is in many cases a preferable *alternative* to direct weight learning. However, the two approaches are complementary, usually yielding improvements in performance when combined.

The paper proceeds as follows. In the next section we review related research. We then describe the lazy random walk framework and each of the learning approaches in detail. In the empirical evaluation section, we describe a set of tasks from the personal information management domain; we then give the results of applying a weight tuning algorithm and reranking separately, and in combination, on these tasks, for multiple real-world corpora. The paper concludes with a discussion of the results and possible future directions.

2. RELATED WORK

Performing search via finite graph walks is closely related to *spreading activation* over semantic or association networks: there the underlying idea is to propagate "activation" from source nodes via weighted links through the network (e.g., [4, 24]). 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 of lazy graph walks discussed here is similar in character, but is less constrained; rather, it relies on learning to optimize how similarity "spreads" through the graph.

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. For example, 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. Recently, the idea of PageRank has been applied to keyword search in structured databases, where edges correspond to inter-entity relations [5, 2].

Several methods have been developed that automatically tune edge weight parameters in extended PageRank models. These include exhaustive local search over each edge type [22], a gradient descent algorithm [7], a hill-climbing error backpropagation algorithm [14], and a hill-climbing approximation algorithm adapted for partial order preferences [1]. A common property of these methods is that they decompose the graph walk into single steps, thus performing optimization "locally". As a case study of these methods, we follow closely in this paper on the error backpropagation gradient descent algorithm [14], applying it to lazy graph walks.

An alternative approach for improving graph walk performance is learning to re-order an initial ranking. A reranking approach has been used in the past for meta-search [10] and also several natural-language related tasks (e.g., [12, 11]). Recently, it has been suggested to apply reranking to improve on graph walks in entity-relation networks [20, 8].

While node reranking can be used as an alternative to weight manipulation, it can readily be used as complementary approach, as the techniques can be naturally combined by first tuning the model parameters, 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 [12]. We follow this paradigm as well in our experiments.

3. FRAMEWORK AND NOTATION

In this section we first provide detailed definitions of the assumed entity-relation graph scheme, and then specify how probability propagates in the graph given a set of edge weights parameters, to derive a measure of similarity between the graph nodes.

A graph G consists of a set of nodes, and a set of labeled directed edges. We will denote graph nodes by letters such as x, y, or z, and an edge from x to y with label ℓ as $x \xrightarrow{\ell} y$. Every node x has a type, denoted $\tau(x)$, where the graph scheme includes a fixed set of possible node types. For example, consider the graph scheme for email data representation described in Figure 1. This graph scheme includes five different node types, denoting *person*, *email-address*, *file*, *term* and *date* entities.

We will assume that the graph edge labels determine the source and target node types: i.e., if $x \xrightarrow{\ell} z$ and $w \xrightarrow{\ell} y$ then $\tau(w) = \tau(x)$ and $\tau(y) = \tau(z)$. For example, according to the scheme defined in Figure 1, an edge $x \xrightarrow{sent-from} y$ implies that node x is of type *file* and node y is of type *person*. However, multiple relations can hold between any particular pair of nodes: that is, it could be that $x \xrightarrow{\ell} y$ and $x \xrightarrow{\ell'} y$, where $\ell \neq \ell'$. For instance, in the email domain, it could be that a particular file was *sent-from* and also *sent-to* the same person. Note 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 example, an email *file* is typically linked to multiple recipients via a *sent-to* relation, a particular *file* entity includes multiple *terms* etc. Finally, for every edge in the graph there is an edge going in the other direction, denoting an inverse relation. (The inverse edges were omitted from Figure 1, for clarity of presentation.) Thus, the graph is cyclic, and there are no sink nodes in the graph.

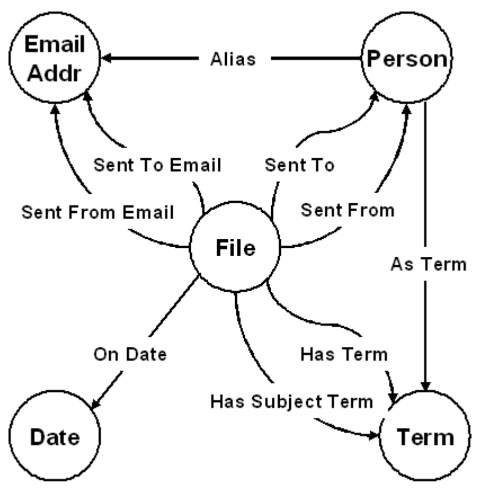

Figure 1: Email ER graph scheme

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 set of parameters Θ. An edge of type ℓ is assigned an edge weight θ_ℓ. For example, the graph scheme shown in Figure 1 includes 18 types of graph edges (including the inverse edge types that are not shown), for which distinct weights are specified.

Let L_{xy} denote the set of edge types of the outgoing edges from x to y. The probability of reaching node y from node x over a single time step is defined as:

$$Pr(x \longrightarrow y) = \frac{\sum_{\ell \in L_{xy}} \theta_\ell}{\sum_{y' \in ch(x)} \sum_{\ell' \in L_{xy'}} \theta_{\ell'}}$$

where $ch(x)$ denotes the set of immediate children of x (the set of nodes that are reachable from x in one time step). That is, the probability of reaching node y from x is defined as the proportion of total edge weights from x to y out of the sum of weights of all the outgoing edges from x.

3.2 Lazy Graph Walks

Conceptually, the edge weights above define the probability of moving from 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

$$\mathbf{M}_{xy} = \begin{cases} (1-\gamma) Pr(x \longrightarrow y) & \text{if } x \neq y \\ \gamma & \text{if } x = y \end{cases}$$

If we associate nodes with integers, and make $\mathbf{M}$ a matrix indexed by nodes, then a walk of k steps can 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 $V_k = V_0 \mathbf{M}^k$. Larger values of γ increase the weight given to shorter paths between x and a destination node z. 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 z in V_k can be interpreted as a similarity measure between x and z.

In this framework, a *query* is an initial distribution V_q over nodes, plus a desired output type τ_{out}, and the answer is a list of nodes z of type τ_{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") the corresponding graph query would be an appropriate distribution V_q over the query terms, with $\tau_{out} = file$. Replacing τ_{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.

4. LEARNING

The described graph framework can be used for many types of tasks (i.e., many flavors of extended semantic similarity), and it is unlikely that a single set of parameter values Θ will be best for all of them. In this section we give a detailed overview of two methods that represent different approaches for the problem of *learning* to better rank graph nodes: a hill-climbing method that tunes the graph weights and the reranking method.

4.1 Problem Setting

By problem definition, we are given a graph G and an initial set of graph parameters Θ. We are also given a set of labeled example queries, for which the identity of at least one graph node that is considered to be correct or relevant is known. (A detailed description of queries per our case study of personal information management and the definition of their respective correct answers is given in Section 5.1.) For every example query V_q, we are given a ranked list of nodes that is output by a graph walk of pre-defined length k. Learning is to be applied, such that the relevant

[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.

output nodes be retrieved at the top of the final ranked list returned, for the given class of queries.

4.2 Error Backpropagation

We follow closely on the algorithm proposed by [14], making slight modifications to adapt it to finite lazy graph walks. They derive a gradient descent algorithm using the paradigm of error backpropagation in neural networks. The target cost function is the following:

$$E = \frac{1}{N}\sum_{i\in N} e_z = \frac{1}{N}\sum_{i\in N}\frac{1}{2}(p_z - p_z^{Opt})^2$$

where e_z is the error for a target node z, defined as the squared difference between the final score assigned to z by the graph walk p_z and some ideal score according to the example's labels, p_z^{Opt}. Specifically, p_z^{Opt} is set to 1 in case that the node z is relevant or 0 otherwise. The error is averaged over a set of example instantiations of size N. We want to minimize this cost function by gradient descent with respect to every edge weight $\theta_{\ell'}$, as follows:

$$\theta_{\ell'} = \theta_{\ell'} - \eta\frac{\partial E}{\partial \theta_{\ell'}} = \theta_{\ell'} - \eta\frac{1}{N}\sum_{i\in N}\frac{\partial e_i}{\partial \theta_{\ell'}}$$

The derivative of the error is a summation over each of the graph walk's time steps, where the final error is propagated backward, weighted by the relative contribution of every intermediate node to the final node score, as follows:

$$\frac{\partial e}{\partial \theta_{\ell'}} = (p_z - p_z^{opt})\sum_{t=0}^{T-1}\sum_{y\in U_z(t+1)} P(y, t+1 \to z, T)\cdot\frac{\partial p_y(t+1)}{\partial \theta_{\ell'}}$$

where $U_z(t+1)$ denotes the set of graph nodes that are in the set of connecting paths leading to z, at time $t+1$; and, given that node y belongs to this set, $P(y, t+1 \to z, t)$ is the total probability of reaching z at time T starting from y at time $t+1$. The derivative of the node y with respect to an edge weight $\theta_{\ell'}$ is:

$$\frac{\partial p_y(t+1)}{\partial \theta_{\ell'}} = \sum_{x\in pa(y)} p_x(t)\gamma\cdot\frac{\partial\frac{{}_{\ell\in L_{xy}}\theta_\ell}{{}_{y'\in ch(x)}\quad{}_{\ell'\in L_{xy'}}\theta_{\ell'}}}{\partial\theta_{\ell'}}$$

where $pa(y)$ is the set of nodes linked by an outgoing edge to y (its parents). Specifically, denoting as $C(\ell', L_{xy})$ the count of edge type ℓ' in the set of connecting paths L_{xy}, the explicit derivative is:

$$\sum_{x\in pa(y)} p_x(t)\gamma\cdot\frac{C(\ell', L_{xy})O_x - C(\ell', L_{xy'})\theta_\ell}{O_x^2}$$

where we use the abbreviation O_x for the total outgoing weight from node x, i.e. $O_x = \sum_{y'\in ch(x)}\sum_{\ell'\in L_{xy'}}\theta_{\ell'}$.

4.3 Node Reranking

Following is a short overview of the reranking approach, described in more detail elsewhere [12]. For every example i $(1 \leq i \leq N)$, the reranking algorithm is provided with the corresponding output ranked list of l_i nodes. Let z_{ij} be the output node ranked at rank j in l_i, and let $p_{z_{ij}}$ be the probability assigned to z_{ij} by the graph walk. Each output node z_{ij} is represented through m features, which are computed by pre-defined feature functions $f_1, \ldots, f_m$. The *ranking function* for node z_{ij} is defined as:

$$F(z_{ij}, \bar{\alpha}) = \alpha_0 log(p_{z_{ij}}) + \sum_{k=1}^{m}\alpha_k f_k(z_{ij})$$

where $\bar{\alpha}$ is a vector of real-valued parameters. Given a new test example, the output of the model is the output node list reranked by $F(z_{ij}, \bar{\alpha})$.

To learn the parameter weights $\bar{\alpha}$, we use a boosting method [12], which minimizes the following loss function on the training data:

$$ExpLoss(\bar{\alpha}) = \sum_i\sum_{j=2}^{l_i} e^{-(F(z_{i1},\bar{\alpha}) - F(z_{ij},\bar{\alpha}))}$$

where z_{i1} is, without loss of generality, the correct target node.[2] 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. Provided that the features are binary, closed form formulas exist for calculating the optimal additive parameter updates [25].

4.3.1 *Features*

While typically the ranked list of candidates was generated using local search methods, reranking features can represent global phenomena that was not captured in the local model, using this information to discriminate between the top ranked candidates. For example, previous work [11] applied a MaxEnt learner to perform named entity tagging; then, reranked high-probability annotations using features describing the entity boundaries predicted.

In general, the scope of features possible is unlimited, allowing one to incorporate many types of relevant information. In this paper, however, we compare reranking with graph parameter tuning as *alternative* learning methods that improve on graph walk performance. The features we use for reranking will therefore be derived from the set of paths leading to every candidate node (that is, the same information available to the error backpropagation algorithm), describing non-local properties of these paths. In particular, we will evaluate the following three types of feature templates:

- *Edge label n-grams* - features indicating whether a particular sequence of n edge labels $(n < k)$ occurred within the set of paths leading to the node.

- *Top edge label n-grams* - these features are similar to the previous feature type. However, here we consider the subset of top t paths that had the largest contribution to the final node probability score.

- *Source count* - In case that the query included multiple nodes, this feature indicates the number of different source nodes in the set of connecting paths leading to the candidate node. This feature models the assumption that nodes reachable from multiple query source nodes are more relevant to the query.

[2]If there are $K > 1$ target nodes in a ranking, we split the ranking into K examples.

5. EMPIRICAL EVALUATION

We evaluate the algorithms on a set of common tasks from the domain of personal information management, where the underlying graph describes an email corpus (jointly with meeting entries if applicable [19]). Figure 1 illustrates the relational schema used; using this schema, a whole email corpus can be represented as a linked network of entities such as email files, person, email-addresses etc. Below is a short description of the tasks and corpora included in the evaluation (see also [20, 19]).

5.1 Tasks

We define a task as a query class, where independently of the query parameter values, a particular type of similarity or association between objects is sought. For example, in the task of *threading*, a user (human, or an automatic email processing agent) looks for messages that are adjacent to a given message in a thread. Following is a description of the tasks considered in our experiments, and their corresponding representation as queries in the graph walk framework. Table 1 gives a summary of the mappings between task and query representation for each of these tasks.

5.1.1 Person Name Disambiguation

Consider an email message containing a term that is identified to be a person's name. It can be non-trivial for an automated system to find out which person node maps to the given term. The task is especially difficult when the mentioned person does not appear in the email header, or when the name mention is ambiguous. For example, common names like "Andrew" may map to multiple persons names in the corpus. Graph walks address this problem, as they encode entity cooccurences. In addition, starting the walk from a distribution that gives equal weight to the term node and the relevant email message node provides natural context and allows to resolve ambiguous instances. Formally, this problem is defined as the following search task: given a *term* node that corresponds to a name-mention and the relevant *email file* node, we formulate a uniform query distribution V_q, and retrieve a ranked list of nodes of type *person*.

5.1.2 Threading

This is the problem of retrieving messages in an email thread, given a message from that thread. As threads are indicated by multiple linkage types, including similar text, common social network and time proximity, this task benefits 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-file* nodes, where the immediate parent and child of the given file are considered to be "correct" answers.

5.1.3 Email Alias Finder

The task considered is automatic assistance in finding a person's set of email-addresses. We consider the following setting: given a person's first name, retrieve a ranked list of *email-address* nodes. For this particular task, we added "similarity" edges to the graph schema, where *email-address* node pairs for which a Jaro string similarity score [9] exceeded a certain threashold have been connected with this edge type.

5.2 Corpora

task	V_q	τ_{out}
Name disambig.	*term* (name mention) *+file*	*person*
Threading	*file*	*file*
Email aliases	*term* (person's name)	*email-address*

Table 1: Query realizations of the considered tasks

We evaluated each of the tasks above using multiple corpora, where we used the following corpora:

The **Cspace** corpus, containing email messages collected from a management course conducted at Carnegie Mellon University in 1997 . In this course, MBA students, organized in teams of four to six members, ran simulated companies in different market scenarios. The corpus used includes the emails of all teams over a period of four days.

The **Enron** corpus, including emails of Enron Corp. employees, which has been made available to the research community [18]. Here, we use the saved email of these users: Sager, Shapiro, Farmer and Germany.[3]

Finally, the **Meetings** corpus includes messages from this paper's second author's email Meetings folder, over a period of several months. This corpus includes also meeting entries, as maintained in a PalmPilot handset, over the same period.

Table 2 details the set of experiments conducted by corpus, corpus size (number of nodes in the corresponding graph representation) and dataset size. The datasets, including labelled queries, were split into training, development and test sets, as specified in the table.

5.3 Experiments

In all experiments, we assign stay probability $\gamma = 0.5$, leading to fast convergence of node probabilities. In addition, the graph scheme used (Figure 1) gives high node connectivity (i.e., small graph diameter), such that a small number of graph walk steps give good performance. In the experiments we used $k = 2$ for most tasks and $k = 3$ steps for the *Alias* task.

We compare the gradient descent and reranking as follows. The gradient algorithm is given the *train* data, where we also compute the resulting error (cost function value) for the *development* set after every iteration. The gradient learning is terminated when either the train set error converges or when the development set error starts rising. (This procedure is intended to prevent over-fitting, a potential problem with small datasets). Since the gradient descent algorithm is prone to converge to a local minima (unfortunately, the target function is not smooth), we ran the algorithm for every task and corpus (train set) combination for 10 randomly generated initial graph parameter sets, out of which we consider the parameters for which the best end result is reached[4] by the gradient algorithm, Θ^0. The output of this procedure is a modified set of weights Θ^G; we then applied graph walks using Θ^G to evaluate performance on the test set queries.

Reranking was trained separately, using both the *train* and *development* sets, where for comparison reasons, the same set of selected initial graph weights was used to generate the graph walk output (thus, both methods are compared

[3]The Enron corpus is available from the first author's homepage.

[4]Results will be presented in terms of Mean Average Precision (MAP). We found that the error function and MAP are well-correlated.

Task		**Corpus**	**Size**	**Train**	**Dev.**	**Test**
Disambig.	D1	M.Game	6248	20	25	60
	D2	Sager	9753	15	12	35
	D3	Shapiro	13174	15	10	35
Threading	T1	M.Game	6248	20	25	80
	T2	Farmer	14082	22	23	93
	T3	Germany	12730	24	21	42
Alias	A1	Meetings	3239	20	-	15

Table 2: Datasets' statistics

	Baseline	**Gradient**	**Reranking**	**Combined**
D1	0.54	0.63*	0.64	0.67
D2	0.66	0.68	0.67	0.68
D3	0.47	0.46	0.63*+	0.70*+
T1	0.51	0.62*	0.72*+	0.77*+
T2	0.65	0.76*	0.83*	0.84*+
T3	0.65	0.67*	0.75*	0.74*
A1	0.62	0.79	0.58	0.73

Table 3: Results (MAP)

against the same baseline ranking, generated by the graph walk with parameters Θ^0). For every query, the top 50 nodes (or less, using the maximum available) were reranked; for the *Alias* task we rerank the top 100 nodes.

In addition, we considered a combination of the two learners in a pipeline fashion, as follows: graph rankings were generated using the set of weights as modified by the gradient learner, Θ^G; then, reranking was applied given the generated node lists.

5.4 Results

The results are presented in Table 3 in terms of Mean Average Precision (MAP), which is a common measure for answer set relevance in Information Retrieval. Results marked with an asterisk are significantly different from the baseline (using a two-sided Wilcoxon test at 95% confidence level). Results marked with a plus sign are significantly different from the gradient method's performance. Figure 2 provides a different view of these results for some of the evaluated datasets, showing the curve of recall ratio at rank K up to rank 10 for the set of evaluated examples. For example, recall level of 0.3 at rank 1 would mean that for 30% of the evaluated queries, the top rank in the returned list contained a relevant answer. Obviously, a higher curve is preferable, where one is interested in maximizing performance primarily at the top ranks.

Overall, the results show consistent trends over tasks and corpora. Both the gradient descent learner and reranking improve results in most of the experiments; in most cases, the reranking method is superior to the weight-tuning learner, yielding higher improvment rates. Reranking failed to improve results for the alias task. The gradient learner slightly degraded performance on one of the tasks, and improved performance on the remaining tasks. The right-most column in the table ("Combined") shows the results of applying reranking using the weights modified by the gradient as the start point. Usually, this hybrid method gives the best performance.

There are several reasons for the observed trends. A main reason for the superiority of reranking for some of the datasets is that reranking features capture global information, describing sequences of relations. In the *threading* task, for example, an adjacent message in a thread is often a reply-to message, where a recipient becomes the sender and vice versa, etc. This composite relation is captured by edge bigrams such as *sent-to*$\rightarrow$ *sent-from-inverse*. The gradient descent, however, does not model multi-steps dependencies, and therefore yields smaller improvements for this task over all the evaluated corpora. In addition, reranking can reward nodes for which the set of connecting edge sequences is diverse. Thus, it may prefer a node that is connected to the source distribution through multiple different connecting paths to a node that got a higher probability score by the graph walk due to a single or fewer connecting paths. This property appears to be beneficial in the explored datasets. The *source count* feature makes a similar contribution in the *person-disambiguation* task, where the initial query includes multiple nodes, and the relevant nodes are often those that are reachable from all of the source nodes.

There are several possible reasons why reranking degraded performance for the alias finding task. First, the set of *n-gram* features derived for this task, for which we applied a 3-step walk, is larger than the set of the features derived for a walk of 2 steps (that is, here trigrams of edges are considered as well). A possible solution for a disproportion between a large feature set size and a relatively small number of training examples is feature selection. The data we had available for the *Alias* task was not sufficient for effectively applying such a procedure. Another possible explanation is that the Meetings corpus is small and not sufficiently dense to demonstrate consistent trends over sets of connecting paths. We conjecture that a larger corpus and as well as a larger training set would be beneficial in this case for reranking.

Finally, it is observed from the results that the combination of local tuning and reranking gives the best performance in the majority of benchmarks, probably due to the reduced noise level of the initial ranking. (Although, the combined approach was not found to be significantly different from reranking.)

6. CONCLUSION

We discussed two approaches for improving graph walks performance, for the general task of searching for entities that demonstrate a particular extended semantic similarity to a given distribution in an entity-relation graph. In particular, we considered a hill-climbing error backpropagation method, which considers local information; and discriminative reranking that uses global graph walk features, including sequences of the traversed edge types. Empirical evaluation using real-world data and multiple corpora showed that while the relations explicitly represented in the graph are of first order, considering global features using the reranking approach is often advantageous, as global features can capture high-level properties of the set of connecting paths between entities, which are informative for various tasks. We believe that the graph scheme and tasks considered in this paper are representative of many real-world problems. The empirical evaluation conducted in this work thus suggests that the reranking approach may be preferable to a weight tuning approach in optimizing ranking performance in similar settings. Furthermore, we have shown that the combination of the two learning approaches usually leads to further improvements.

In the future we would like to address scalability issues of the reranking method. In particular, the suggested feature

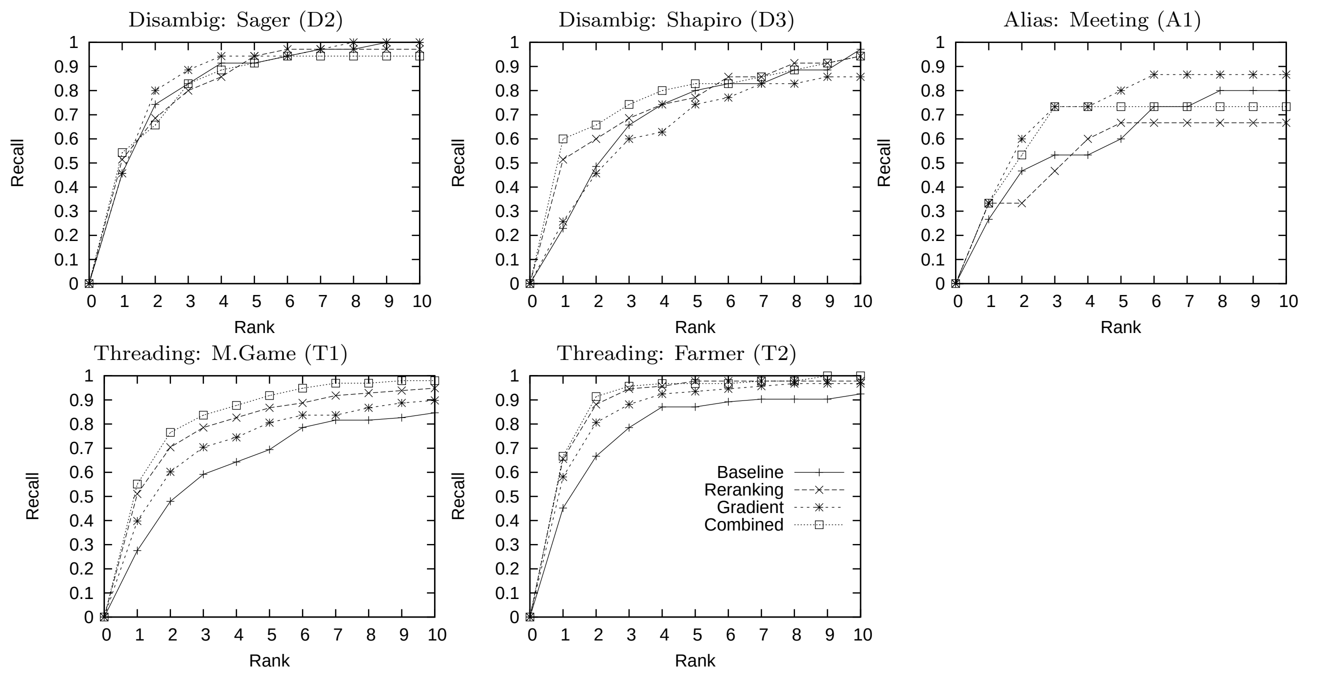

Figure 2: Results: Recall at Rank K

sets that capture n-gram sequences of edge types within the set of connecting paths grow exponentially with the length of the paths. We believe that feature selection may resolve this issue. Otherwise, there may be other feature templates that capture relevant global information in such settings more efficiently. The space of possible features that can describe the semantic relevancy of a graph node in terms of the graph walk has yet to be explored. Features can be suggested that capture pehnomena relevant for a particular type of semnatic similarity sought (for example, path symmetry), or that add information relevant for a particular domain.

7. ACKNOWLEDGMENTS

The authors wish to thank Andrew Y. Ng and Elchanan Mossel for their advice on the gradient descent algorithm implementaion. 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).

8. REFERENCES

[1] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entities. In *KDD*, 2006.

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

[3] M. Barthelemy, E. Chow, and T. Eliassi-Rad. Knowledge representation issues in semantic graphs for relationship detection. In *AAAI Spring Symposium*, 2005.

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

[5] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE*, 2002.

[6] S. Chakrabarti, J. Mirchandani, and A. Nandi. Spin: Searching personal information networks. In *SIGIR*, 2005.

[7] H. Chang and D. Cohn. Learning to create customized authority lists. In *ICML*, 2000.

[8] W. W. Cohen and E. Minkov. A graph-search framework for associating gene identifiers with documents. *BMC Bioinformatics*, 7(440), 2006.

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

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

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

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

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

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

[15] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD*, 2004.

[16] T. Hughes and D. Ramage. Lexical semantic

relatedness with random graph walks. In *EMNLP*, 2007.

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

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

[19] E. Minkov and W. W. Cohen. An email and meeting assistant using graph walks. In *CEAS*, 2006.

[20] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR*, 2006.

[21] J. Neville and D. Jensen. Data mining in social networks. In *National Academy of Sciences Symposium on Dynamic Social Network Analysis*, 2002.

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

[23] 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.

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

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

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

[27] 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.

[28] B. Zhao, P. Sen, and L. Getoor. Entity and relationship labeling in affiliation networks. In *ICML Workshop on Statistical Network Analysis*, 2006.