Mach Learn (2010) 81: 53–67
DOI 10.1007/s10994-010-5205-8

Relational retrieval using a combination of path-constrained random walks

Ni Lao · William W. Cohen

Received: 30 April 2010 / Accepted: 20 June 2010 / Published online: 22 July 2010

Abstract Scientific literature with rich metadata can be represented as a labeled directed graph. This graph representation enables a number of scientific tasks such as *ad hoc* retrieval or named entity recognition (NER) to be formulated as *typed proximity queries* in the graph. One popular proximity measure is called *Random Walk with Restart* (RWR), and much work has been done on the supervised learning of RWR measures by associating each edge label with a parameter. In this paper, we describe a novel learnable proximity measure which instead uses one weight per edge label *sequence*: proximity is defined by a weighted combination of simple "path experts", each corresponding to following a particular sequence of labeled edges. Experiments on eight tasks in two subdomains of biology show that the new learning method significantly outperforms the RWR model (both trained and untrained). We also extend the method to support two additional types of experts to model intrinsic properties of entities: *query-independent experts*, which generalize the PageRank measure, and *popular entity experts* which allow rankings to be adjusted for particular entities that are especially important.

Keywords Entity relation graph · Random walk · Learning to rank · Relational model · Filtering and recommending

1 Introduction

Most past research on accessing the scientific literature has focused on a small-number of well-defined tasks which represent the scientific literature as a set of documents: such tasks include *ad hoc* retrieval based on keyword queries, or named entity recognition (NER) and

Editors: José L Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag.

N. Lao (✉) · W.W. Cohen
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
e-mail: nlao@cs.cmu.edu

W.W. Cohen
e-mail: wcohen@cs.cmu.edu

normalization. In fact, scientific literature naturally includes substantial metadata such as author names, citations, and publication venues, as well as derived metadata (such as gene and protein names, in the biomedical literature). An alternative way to represent the scientific literature is as a labeled directed graph, with typed nodes representing documents, terms, and metadata, and labeled edges representing the relationships between them (e.g., "authorOf", "datePublished", etc.). This graph represents not only text, but also implicitly includes social-network information (via co-authorship, and paths between authors and conference venues), expertise information (via paths between an author and entities mentioned in her publications). Domain knowledge can also be easily added to the graph (e.g., adding known relationships between entities, such as protein-protein interaction information).

Representing the scientific literature as a labeled graph enables a number of scientific tasks to be formulated as *typed proximity queries* in the graph, in which the user provides as input a set of *query nodes* and *answer type*, and receives as output a list of nodes of the desired answer type, ordered by *proximity* to the query nodes. For instance, traditional keyword-based ranked retrieval of documents can be formulated as a proximity query where the query nodes are term nodes, and the answer type is "document"; also, in past research, future collaborations between scientists have been predicted by proximity queries on a co-authorship graph (Liben-Nowell and Kleinberg 2007), document-level gene annotations have been generated using proximity queries on a graph of based on NER-annotated documents and known entity synonyms, and publications involving new gene-protein entities have been predicted using proximity queries on co-authorship graph including document-level metadata on entities (Arnold and Cohen 2009). In general, the appropriate notion of "proximity" may be task- or user-specific, and hence must be learned or engineered; however, there are also general-purpose graph proximity measures such as *random walk with restart* (RWR) (also called personalized PageRank) which are fairly successful for many types of tasks.

In this paper we will consider the use of typed proximity queries to solve four tasks: a "gene recommendation" task considered by Arnold and Cohen (2009), and three additional tasks we call *venue recommendation*, *reference recommendation*, and *expert-finding*. As we will argue below, all of these tasks are plausible surrogates for tasks commonly performed by scientists, and data for them is readily obtainable, making them suitable for learning and evaluating task-specific proximity measures for the scientific literature. We evaluate these four tasks, in two subdomains of biology each, and evaluate performance on 2000 test queries for each of these eight tasks.

The principle contribution of this paper is the development of a new method for learning proximity measures on labeled graphs. In particular, we describe a novel scheme for parameterizing such a measure, in which a proximity measure is defined by a weighted combination of simple "path experts", each of which corresponds to a particular labeled path through the graph. The new learning method outperforms untrained RWR on all eight tasks, achieving an improvement in MAP scores of up to 43%. The new learning method also outperforms a widely-used simpler parameterization in which a weight is associated with each label in the graph, again producing high MAP scores on all eight tasks.

Another contribution of the paper is extension of the method to support two additional types of experts, which we call *query-independent experts* and *popular entity experts*. query-independent experts provide a rich set of query-independent ranking schemes similar to the PageRank measure. Popular entity experts allow rankings to be adjusted for particular entities that are especially important: for instance, an popular entity expert might assign a higher weight to the specific venue "PKDD" when the query contains the keyword "mining".

The work of this paper is most closely related to other systems that learn task-specific proximity measures on labeled graphs. Most of these systems have used some variant of the simpler one-weight-per-edge-label parameterization scheme which we use as our baseline (e.g., Diligenti et al. 2005; Chakrabarti and Agarwal 2006; Toutanova et al. 2004). One line of work that uses a richer "feature set" is described in Minkov et al. (2006), which explored using n-grams of edge labels as features for re-ranking results of an RWR-based system, and Minkov and Cohen (2008), who proposed a method that upweights RWR-paths which are more likely to reach relevant entities. Our approach can be viewed as principled discriminative version of this algorithm—one important advantage of which is the ability to easily incorporate additional types of information, such as the query-independent and popular entity experts described above.

There is an interesting connection between the Relational Retrieval (RR) problems considered in this work and Statistical Relational Learning (SRL) problems (Getoor and Taskar 2007). RR and SRL have slightly different task definitions: retrieval *vs.* classification, and their underlying inference methods have different complexities: RWR is generally more efficient than the inference for graphical models (e.g. Markov Logic Networks (Richardson and Domingos 2006)). However, RR and SRL are based on the same data model—entity relation graphs, and they share a set of related challenges: efficient parameters estimation, efficient structure learning (or ILP), and hidden concept discovery (or predicate invention).

In the remainder of the paper, we first describe the tasks and the datasets we will use in our experiments in more detail. We next present baseline results using RWR, a robust general-purpose proximity measure. In the next section, we present the path ranking algorithm, describing first the way in which path experts are enumerated, then the learning algorithm, and finally the two extensions of query-independent experts and popular entity experts. We then describe the experimental results with the new learning methods and conclude.

2 The dataset and tasks

2.1 Tasks

We consider here three new tasks that are well-suited to solution by typed proximity queries.

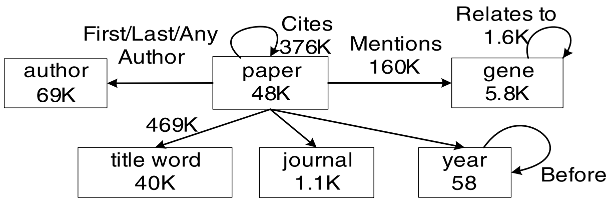
Venue recommendation is the problem of finding a venue to publish a new research paper. Here the query is a heterogeneous set of nodes: the terms in the title of the new paper, the set of entities (genes or proteins) associated with the paper, and the current year. The answer type is "journal", so the answer will be a list of biological journals, ranked by suitability for the new paper.

Reference recommendation (or citation recommendation) is the problem of finding relevant citations for a new paper. The query is, as in venue recommendation, the title terms and relevant entities for the new paper, the current year, and the answer type is "paper". The desired answer is a list of papers ranked by appropriateness as citations in the new paper. This task is similar to The TREC-CHEM Prior Art Search Task (Lupu et al. 2009), and can also be seen as a simplified version of the context-aware citation recommendation task (He et al. 2010).

Expert finding is the problem of finding a domain expert for a particular topic. The query is again a list of terms and relevant entities, the current year, and the answer type is "person". The desired answer is a list of people with expertise on this topic.

The first two of these tasks are encountered in preparing a new paper, and the third is encountered in finding reviewers, or new collaborators. To evaluate performance on these tasks, we will compare the ranked list from a query associated with a paper to the actual

Fig. 1 Schema of the yeast data



metadata associated with the paper: specifically, we will compare actual venue to the recommended venues, the actual citations to the recommended citations. Perhaps more speculatively, we will also compare the authors of a paper to the experts recommended by the query based on the title and related-entity set for the paper. In each case the predictions will be made using a graph that does not contain the actual paper in question—see the next subsection for details.

As a fourth task, we will consider the *gene recommendation* task considered by Arnold and Cohen (2009)—i.e., predicting, given past publishing history, which genes an author will publish about over the next year. Here the query nodes are an author and a year, and the answer type is "gene". This task is an approximation to predicting future interests.

Because the fly data is larger than the yeast data, and has an extra entity type protein, we do not use the publication year as part of the query for fly data.

2.2 Datasets

Most previous work on supervised training of RWR-based proximity measure have used a small number of training queries—for instance, Minkov and Cohen used more than 30 queries (Minkov et al. 2006)—or else used artificially generated document orderings (Tsoi et al. 2003; Agarwal et al. 2006). Using large amounts of realistic data in this study makes it possible to learn more complex models. We created two publication data sets (Yeast and Fly) in the biological domain. Paper content and metadata information are crawled from two resources: *PubMed*[1] is a free on-line archive of over 18 million biological abstracts for papers published since 1948; *PubMed Central* (PMC)[2] contains full-text and references to over one million of these papers.

Figure 1 shows the schema of the yeast corpus. We extracted gene mentions from the *Saccharomyces Genome Database(SGD)*,[3] which is a database of various types of information concerning the yeast organism Saccharomyces cerevisiae, including about 48K papers, each annotated with the genes it mentions. The title words are filtered by a stop word list of size 429. The *Authorship* relations are further distinguish into three sub-types: any author, first author, and last author. We extracted gene-gene relations from *Gene Ontology* (GO),[4] which is a large ontology describing the *properties* of and *relationships* between various biological entities across numerous organisms.

Figure 2 shows the schema of the fly corpus. It is extracted from *Flymine*,[5] which is an integrated database for Drosophila and Anopheles genomics, and contains about 127K

[1] www.ncbi.nlm.nih.gov/pubmed.

[2] www.ncbi.nlm.nih.gov/pmc.

[3] www.yeastgenome.org.

[4] www.geneontology.org.

[5] www.flymine.org.

Fig. 2 Schema of the fly data

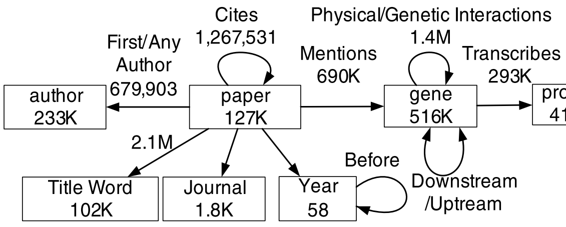


Table 1 Corpus statistics

	Graph size			No. query		
	Paper	Node	Edge	Train	Dev	Test
Yeast	48K	164K	2.8M	2K	2K	2K
Fly	127K	770K	3.5M	2K	2K	2K

papers tagged with genes and proteins. The schema is similar to that of the yeast data, except for a new entity type *Protein*,[6] and several relations among genes. *Downstream* and *Upstream* relation connect a gene to its two neighbors on the DNA strand.

Each paper can be used to simulate a query and relevance judgments for any of the four above mentioned tasks. However, we need to prevent the system from using information obtained later than the query's date. Therefore, we define a *time variant graph* in which each edge is tagged with a time stamp (year). When doing random walk for a query generated from a particular paper, we only consider edges that are earlier than the publication date of that paper.

For each task on any of the two corpora, we randomly hold out 2000 queries for development, and another 2000 queries for testing. We evaluate models by Mean Average Precision (MAP).

2.3 Baseline results

The RWR based retrieval model is commonly used and studied by today's research community and is serving as our baseline. The two recent extensions with richer "feature set" (Minkov et al. 2006; Minkov and Cohen 2008), however, are not selected as baselines for this work, mainly because we are focusing on large scale problems here. These two extensions are designed for smaller scale problems, therefore not efficient enough to deal with the Fly and Yeast data sets we used in this study.

Table 2 shows the result of our two baseline methods: untrained RWR model with all edges set to uniform weight 1.0, and trained RWR model (detail of which will be described in Sect. 3.2). Basically, a random walker can follow any type of edge at each step in a RWR model. While in a trained RWR model, the walker can have preference over different type of edges which is expressed as edge weights. We can see that except on the gene recommendation tasks, supervised training can significantly improve retrieval quality. By comparing four tasks we can see that venue and gene recommendation are relatively easier

[6]In yeast, there is a nearly one-to-one relationship between genes and proteins, as most genes are transcribed to a unique protein; in flies, alternative splicing means that a gene can be transcribed to several different proteins.

Table 2 MAP of the baseline RWR model, and RWR model with learning based on one weight per edge label (see Sect. 3.2 for details). The *numbers in parenthesis* are relative improvement of MAP(%). Except these[†], all improvements are statistically significant at $p < 0.0001$ using paired t-test

Corpus	Task	RWR	
		Untrained	Trained
Yeast	Venue recommendation	40.4	44.2 (+9.4)
Yeast	Reference recommendation	11.8	16.0 (+35.6)
Yeast	Expert finding	9.9	11.1 (+12.1)
Yeast	Gene recommendation	14.4	14.4 (+0.0)[†]
Fly	Venue recommendation	45.4	48.3 (+6.4)
Fly	Reference recommendation	18.8	20.5 (+9.0)
Fly	Expert finding	5.6	7.2 (+28.6)
Fly	Gene recommendation	18.7	19.2 (+2.7)[†]

tasks because they have smaller number of candidate answers. Although the reference recommendation task has large number of candidate entities, the models effectively leverage the citation links among papers to achieve reasonably good retrieval accuracy. Among all four tasks, expert finding is the hardest one.

3 The path ranking algorithm (PRA)

3.1 Basic path experts

One-parameter-per-edge label RWR proximity measures are limited because the *context* in which an edge label appears is ignored. For example, in the reference recommendation task, one of the query nodes is a year. There are two ways in which one might use a year y to find candidate papers to cite: (H1) find papers published in year y, or (H2) find papers frequently cited by papers published in year y. Intuitively, the second heuristic seems more plausible than the first; however, a system that insists on a using a single parameter for the "importance" of the edge label *PublishedIn* cannot easily encode this intuition.

To define heuristics of this sort more precisely, let R be a binary relation. We write $R(e,e')$ if e and e' are related by R, and define $R(e) \equiv \{e' : R(e,e')\}$. We use $dom(R)$ to denote the domain of R, and $range(R)$ for its range. A *relation path* P is a sequence of relations $R_1 \dots R_\ell$ with constraint that $\forall i : 1 < i < \ell - 1, range(R_i) = dom(R_{i+1})$. We define $dom(R_1 \dots R_\ell) \equiv dom(R_1)$ and $range(R_1 \dots R_\ell) \equiv range(R_\ell)$, and when we wish to emphasize the types associated with each step in a path, we will write the path $P = R_1 \dots R_\ell$ as

$$T_0 \xrightarrow{R_1} \cdots \xrightarrow{R_\ell} \cdots T_\ell$$

where $T_0 = dom(R_1) = dom(P)$, $T_1 = range(R_1) = dom(R_2)$ and so on. In this notation,the two heuristics suggested above would be written as:

$$H1:\ year \xrightarrow{PublishedIn^{-1}} paper$$

$$H2:\ year \xrightarrow{PublishedIn^{-1}} paper \xrightarrow{Cite} paper$$

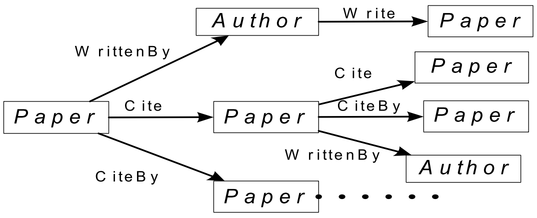


Fig. 3 A 2-level relation tree for a simple schema of paper and author

This notation makes it clear that the range of each relation path is *paper*, the desired type for reference recommendation. We use $^{-1}$ to denote the inverse of a relation, which is considered as a different relation: for instance, *PublishedIn* and *PublishedIn*$^{-1}$ are considered as different relations.

For any relation path $P = R_1 \dots R_\ell$ and set of query entities $E_q \subset dom(P)$, we define a distribution $h_{E_q,P}$ as follows. If P is the empty path, then define

$$h_{E_q,P}(e) = \begin{cases} 1/|E_q|, & \text{if } e \in E_q \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

If $P = R_1 \dots R_\ell$ is nonempty, then let $P' = R_1 \dots R_{\ell-1}$, and define

$$h_{E_q,P}(e) = \sum_{e' \in range(P')} h_{E_q,P'}(e') \cdot \frac{I(R_\ell(e',e))}{|R_\ell(e')|},$$

where $I(R(e',e))$ is an indicator function that takes value 1 if $R(e',e)$ and 0 otherwise. If we assume that $I(R(e',e)) = 0$ when e' is not in $dom(R)$, then the definition naturally extends to the case where E_q is not a subset of $dom(P)$.

Given these definitions, the intuition that "heuristic H1 is less useful than H2" could be formalized as follows: for reference recommendation queries E_q, T_q, where E_q is a set of title words, gene-protein entities, and a year y, entities e_1 with high weight in $h_{E_q,PublishedIn^{-1}}$ are not likely to be good citations, where as entities e_2 with high weight in $h_{E_q,PublishedIn^{-1}.Cite}$ are likely to be good citations. More generally, given a set of paths $P_1, \dots, P_n$, one could treat these paths as features for a linear model and rank answers e to the query E_q by

$$\theta_1 h_{E_q,P_1}(e) + \theta_2 h_{E_q,P_2}(e) + \dots + \theta_n h_{E_q,P_n}(e)$$

where the θ_i are appropriate weights for the paths.

In this paper, we consider learning such linear weighting schemes over all relation paths of bounded length ℓ. For small ℓ (e.g., $\ell \le 4$), one can easily generate $\mathcal{P}(q,l) = \{P\}$, the set of all type-correct relation paths with range T_q and length $\le l$. The distributions defined by all the relation paths can be summarized as a prefix tree (Fig. 3), where each node corresponds to a distribution $h_P(e)$ over the entities. A PRA model ranks $e \in I(T_q)$ by the scoring function

$$s(e;\theta) = \sum_{P \in \mathcal{P}(q,l)} h_{E_q,P}(e)\theta_P \tag{2}$$

In matrix form this could be written $s = A\theta$, where s is a sparse column vector of scores, and θ is a column vector of weights for the corresponding paths P. We will call A the *feature matrix*, and denote the i-th row of A as A_i.

We found that, because some of the relations reflect one-to-one mapping, there are paths give exactly the same distribution over the target entities. For example, the following three paths among years are actually equivalent, where *Before*$^{-1}$ is the inverse of relation *Before*:

$$year \xrightarrow{Before^{-1}} year \xrightarrow{Before} year \xrightarrow{Before} year$$
$$year \xrightarrow{Before} year \xrightarrow{Before^{-1}} year \xrightarrow{Before} year$$
$$year \xrightarrow{Before} year \xrightarrow{Before} year \xrightarrow{Before^{-1}} year$$

To avoid creating these uninteresting paths, we add constraint to the following relations that they cannot be immediately preceded by its inverse:

$$year \xrightarrow{Before^{-1}} year \xrightarrow{Before} year$$
$$year \xrightarrow{Before} year \xrightarrow{Before^{-1}} year$$
$$journal \xrightarrow{PublishedBy^{-1}} paper \xrightarrow{PublishedBy} journal$$
$$year \xrightarrow{PublishedIn^{-1}} paper \xrightarrow{PublishedIn} year$$

3.2 Parameter estimation

There have been much previous work in supervised learning of random walk models. Nie et al. (2005) use exhaustive local search over each edge type, which is only applicable when the number of parameters is very small. Diligenti et al. (2005) and its follow up (Minkov et al. 2006) optimize weights on the relations using back-propagation, which has linear convergence, therefore requires many iterations to reach convergence. Recent works (Agarwal et al. 2006; Chakrabarti and Agarwal 2006) use more efficient second order optimization procedure like BLMVM for numerical optimization. In this study, we use L-BFGS (Andrew and Gao 2007), one commonly used second order optimization procedure in many machine learning problems, and binomial log-likelihood loss functions.

The training data can be represented as $\mathcal{D} = \{(q^{(m)}, r^{(m)})\}$, $m = 1 \dots M$, where $r^{(m)}$ is a binary vector. $r_e^{(m)} = 1$ if entity e is relevant to the query $q^{(m)}$, and $r_e^{(m)} = 0$ otherwise. Given the training data, parameter estimation can be formulated as maximizing a regularized objective function

$$O(\theta) = \sum_{m=1\dots M} o^{(m)}(\theta) - \lambda|\theta|_2/2 \tag{3}$$

where λ is a regularizer, and $o^{(m)}(\theta)$ is a per-instance objective function. In this paper we use binomial log-likelihood (the loss function for logistic regression); however, negative hinge loss (for SVM), negative exponential loss (for boosting), and many other functions could be used instead. Binomial log-likelihood has the advantage of being easy to optimize, and also does not penalize outlier samples too harshly, as exponential loss does. For a training instance $(q^{(m)}, r^{(m)})$, let $A^{(m)}$ be its corresponding feature matrix, $\mathcal{R}^{(m)}$ be the index set of the relevant entities, and $\mathcal{N}^{(m)}$ the index set of the irrelevant entities. In order to balance uneven number of positive and negative entities, we use the average log-likelihood of positive and negative entities as the objective

$$o^{(m)}(\theta) = \sum_{i\in\mathcal{R}^{(m)}} \frac{\ln p_i^{(m)}}{|\mathcal{R}^{(m)}|} + \sum_{i\in\mathcal{N}^{(m)}} \frac{\ln(1-p_i^{(m)})}{|\mathcal{N}^{(m)}|} \tag{4}$$

where $p_i^{(m)} = p(r_i^{(m)} = 1; \theta) = \sigma(\theta^T A_i^{(m)})$, σ is the sigmoid function $\sigma(x) = \exp(x)/(1 + \exp(x))$, and the gradient is

$$\frac{\partial o^{(m)}(\theta)}{\partial \theta} = \sum_{i \in \mathcal{R}^{(m)}} \frac{(1 - p_i^{(m)}) A_i^{(m)}}{|\mathcal{R}^{(m)}|} - \sum_{i \in \mathcal{N}^{(m)}} \frac{p_i^{(m)} A_i^{(m)}}{|\mathcal{N}^{(m)}|}. \tag{5}$$

For most retrieval tasks, there are just a few positive entities but thousands (or millions) of negative ones. Therefore using all of them in the objective function is expensive. Here we used a simple strategy similar to stratified random sampling (Pavlu 2008). First, we sort all the negative entities using PRA model without training (i.e., all feature weights are set to 1.0). Then, entities at the $k(k+1)/2$-th positions are selected as negative samples, where $k = 0, 1, 2, \ldots$. This is helpful because, in generally, non-relevant entities highly ranked by some weak ranking function are more important than lower ranked ones: for in-depth comparisons of different selection strategies we refer the reader to Aslam et al.'s work (Aslam et al. 2009).

For parameter estimation of the one-weight-per-edge-label RWR model, we use the same log-likelihood objective function and LBFGS optimization procedure as for PRA. Since a RWR can be seen as the combination of all the PCRWs with each path having its weight set to the product of all the edge weights along the path, we can calculate the gradient of edge weights by first calculating the gradient w.r.t. the paths, and then applying the chain rule of derivative.

3.3 Query-independent experts

The features above describe a entity only in terms of its position in the graph relative to the query entities. However, the relevance of an entity may also depend on query-independent qualities—for instance, its recency of publication, its citation count, or the authoritativeness of the venue in which it was published. To account for these intrinsic properties of entities, we extend every query set E_q to include a special entity e^*. We then extend the graph so that for each type T, there is a relation *AnyT* such that *AnyT*(e^*, e) is true for every $e \in T$. For example, the relation *AnyPaper* maps e^* to each paper, and the relation *AnyYear* maps e^* to each year.

For example, the path $e^* \xrightarrow{AnyPaper} paper \xrightarrow{Cite} paper$ defines this random-walk process: start from any paper with equal probability, and then jump to one of its referenced papers. This results in higher probability mass to the papers with high citation count. A path that starts with *AnyPaper* and then follows two *Cite* edges assigns weight to papers frequently cited by other highly-cited papers: as path length increases, a combination of this variety of *query-independent paths* begins to approximate the PageRank for papers on the citation graph.

These scores can be seen as a rich set of query-independent features, which can be combined with query-dependent path features to rank the target entities. To use them, the scoring function in (2) remains unchanged. However, since these paths are query-independent, we improve performance by computing their values for every entity offline. In particular, using the time-variant-graph described in Sect. 2.1, we calculate, for each year, the h score for all query-independent paths, using only edges earlier than that year.

3.4 Popular entity experts

Previous work in information retrieval has shown that entity specific characteristics can be leveraged for retrieval. For the ad hoc retrieval task, some lower ranked document under a query may be interesting to the users and got clicked very often because of features not captured by the ranking function of the system. In this case, promoting these popular documents to higher rank would result in better user experience (White et al. 2007). For personalized search (Dou et al. 2007), different users may have different information needs under the same query: for instance, the word "mouse" can mean different things for a biologist and a programmer. In this case, modeling the correlation between query entities (users) and target entities (documents) can be useful.

In this work, we provide a simple yet general way of modeling entity popularities by adding biases and query-conditioned biases to the target entities. For a task with query type T_0, and target type T_q, we introduce a *popular entity* bias θ_e^{pop} for each target entity $e \in T_q$. We also introduce a *conditional popular entity* bias $\theta_{e',e}$ for each query-target entity pair (e', e), where $e' \in T_0$, $e \in T_q$. The scoring function in (2) is extended to

$$s(e;\theta) = \sum_{P \in \mathcal{P}(q,l)} h_{E_q,P}(e)\theta_P + \theta_e^{pop} + \sum_{e' \in E_q} \theta_{e',e}^{pop}, \tag{6}$$

or in matrix form $s = A\theta + \theta^{pop} + \Theta q$, where θ^{pop} is an concatenation of all bias parameters, Θ is an matrix of all conditional bias parameters, and q is a binary vector indicating whether each entity is included in the query.

We can see that the number of parameters is potentially very large. For example, θ^{pop} has the length of the total number of entities of the target type, and Θ is a huge matrix with number of rows and columns equal to the number of entities in the target and query entity type. Since it is impractical to include all of them to the model (consider the task of retrieving documents using words), we use an efficient induction strategy which only add the most important features (Perkins et al. 2003). At each LBFGS training iteration, we add to the model the top J popular entity expert parameters which have the largest gradient (in magnitude) w.r.t. the objective function in (3). We call J the *batch size*. In our experiment, we found $J = 20$ gives relatively good performance. We also restrict the induction to be applied no more than 20 times during training. In this way, the computation cost is not bounded by the size of θ^{pop} and Θ, but the number of non-zero elements in them. In practice, we found that training a PRA model with popular entity experts is not much more expensive than training a regular PRA model, and the details will be given in the next section.

4 Experiment

Biologists currently spend a lot of time in searching for relevant information about specific bioentities (e.g. a set of genes). Here we explore how relational retrieval can help biologists in various tasks. We report empirical results of comparing PRA with unsupervised RWR model, and its supervised version (RWR+train). We also compare to PRA with query-independent path experts (PRA+qip), and PRA with popular entity experts (PRA+pop).

4.1 Parameter tuning on development data

In this subsection, we show the parameter swiping for reference recommendation task on the yeast data. Other tasks have similar trend, but their plots are not shown here due to space limitation.

Figure 4 shows the relation between path tree depth and model complexity. For PRA model we can see that both model complexity (measured by number of features) and query

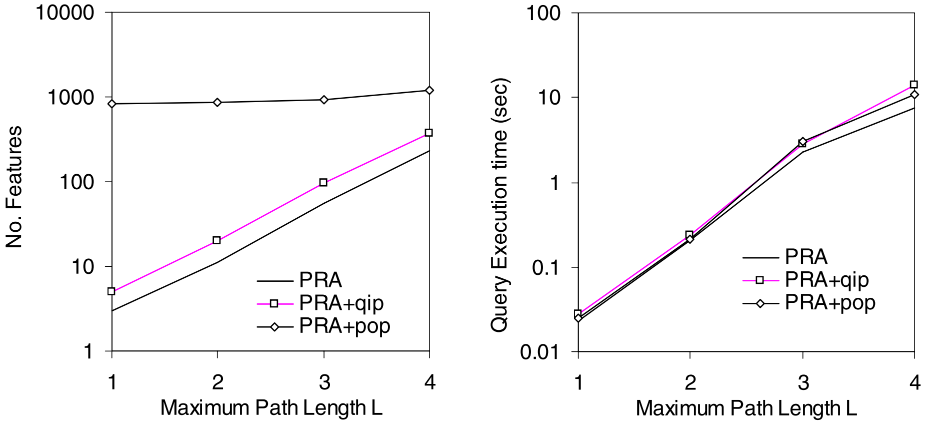


Fig. 4 Model complexity verses maximum path length L for reference recommendation task on the yeast data. Execution time is an average of 2000 test queries

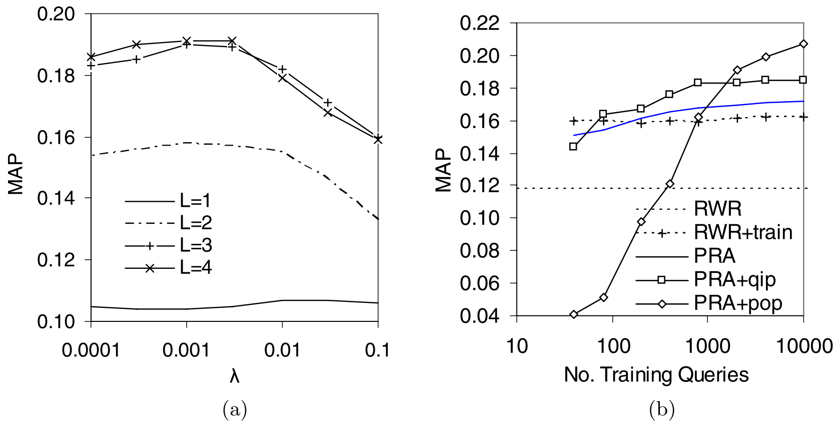


Fig. 5 (**a**) Compare different regularization parameter λ and different path length l. (**b**) Effect of training data size for reference recommendation task on yeast data

execution time are exponential to the path length. The query independent path (qip) extension introduces about twice number of paths than the basic PRA algorithm. However, since these query independent random walks are performed offline, they do not significantly affect query execution time. Although the popular entity experts introduce a large number of features, they are easy to calculate, therefore do not significantly affect query execution time.

Figure 5a shows the effect of L_2-regularization and path length on retrieval quality. We can see that a small amount of L_2-regularization can slightly improve MAP, and longer path lengths give better performances, but only to a certain level. In order to balance between retrieval quality and computational complexity, we fix max path length, for the rest of the experiment, to 4 for the venue recommendation task and 3 for the other three tasks.

In Fig. 5b, we vary the number of training queries to see how training data size affects the quality of the model. we can see that all learning methods benefit from more training data, and it is especially evident when popular entity experts are used. This is due to the fact that

Table 3 Subset of features from a PRA+qip+pop model trained for the reference recommendation task on the yeast data. *In* is a shorthand for the *PublishedIn* relation

ID	Weight	Feature
1	272.4	$word \xrightarrow{HasTitle^{-1}} paper \xrightarrow{Cite^{-1}} paper \xrightarrow{Cite} paper$
2	156.7	$word \xrightarrow{HasTitle^{-1}} paper \xrightarrow{Cite} paper$
3	100.5	$gene \xrightarrow{HasGene^{-1}} paper \xrightarrow{Cite^{-1}} paper \xrightarrow{Cite} paper$
4	83.7	$word \xrightarrow{HasTitle^{-1}} paper \xrightarrow{Cite^{-1}} paper$
5	50.2	$gene \xrightarrow{HasGene^{-1}} paper \xrightarrow{Cite} paper$
6	41.4	$word \xrightarrow{HasTitle^{-1}} paper$
7	29.3	$year \xrightarrow{In^{-1}} paper \xrightarrow{Cite} paper$
8	13.0	$year \xrightarrow{Before^{-1}} year \xrightarrow{In^{-1}} paper \xrightarrow{Cite} paper$
	⋮	
9	3.7	$e* \xrightarrow{AnyPaper} paper \xrightarrow{Cite} paper$
10	2.9	GAL4>Nature. 1988. GAL4-VP16 is an unusually potent transcriptional activator
11	2.1	CYC1>Cell. 1979. Sequence of the gene for iso-1-cytochrome c in Saccharomyces cerevisiae
	⋮	
12	−5.4	$year \xrightarrow{Before^{-1}} year \xrightarrow{In^{-1}} paper$
13	−39.1	$year \xrightarrow{In^{-1}} paper$
14	−49.0	$e* \xrightarrow{AnyYear} year \xrightarrow{In^{-1}} paper$

they have a large number of parameters to estimate, and we need at least a thousand training queries to prevent over fitting and to get good results.

4.2 Examples of important path features

In the TREC-CHEM Prior Art Search Task (Lupu et al. 2009), people found that instead of directly searching for patents with the query words, it is much more effective to first find patents with similar topic, then aggregate these patents' citations. The relation path of this strategy can be expressed as "$query\ word \xrightarrow{ContainedBy} patent \xrightarrow{Cite} patent$". In our experiment, the PRA model based on PCRW not only successfully identifies this paths as an important feature in scientific literature domain (path #2 in Table 3), but also finds several other useful paths.

Table 3 shows a subset of features for a PRA + qip + pop model trained for the reference recommendation task on the yeast data. Feature #1–#8 are regular path features. Among them, feature #6 resembles what most ad-hoc retrieval systems would do to find relevant papers: finding papers with many words overlapping with the query. However, we can see that this feature is not considered the most important by the model. Instead, the model favors the papers that are well cited by on-topic papers (#2), and the papers cited together with the

Table 4 Subset of features from a PRA+qip+pop model trained for the venue recommendation task on the fly data. *In* is a shorthand for the *PublishedIn* relation

ID	Weight	Feature
1	26.9	$word \xrightarrow{HasTitle^{-1}} paper \xrightarrow{In} journal$
2	4.5	$word \xrightarrow{HasTitle^{-1}} paper \xrightarrow{FirstAuthor} author \xrightarrow{FirstAuthor^{-1}} paper \xrightarrow{In} journal$
3	2.8	$word \xrightarrow{HasTitle^{-1}} paper \xrightarrow{AnyAuthor} author \xrightarrow{AnyAuthor^{-1}} paper \xrightarrow{In} journal$
4	1.1	$gene \xrightarrow{GeneticallyRelated} gene \xrightarrow{HasGene^{-1}} paper \xrightarrow{In} journal$
5	0.9	$gene \xrightarrow{HasGene^{-1}} paper \xrightarrow{In} journal$
6	0.6	$e* \xrightarrow{AnyPaper} paper \xrightarrow{Cite} paper \xrightarrow{In} journal$
	⋮	
7	3.5	*virus > J_Virol*
8	2.7	*deficiency > Am_J_Hum_Genet*
9	2.0	*drosophila > Dev_Biol*
	⋮	
10	−2.6	*> J_Virol*
11	−3.2	*drosophila > J_Bacteriol*
12	−3.5	*drosophila > Am_J_Hum_Genet*
13	−3.6	*drosophila > J_Med_Genet*
14	−35.8	$e* \xrightarrow{AnyYear} year \xrightarrow{In^{-1}} paper \xrightarrow{In} journal$

on-topic papers (#1). Papers cited during the past two years (#7, #8) are also favored. In contrary, general papers published during the past two years (#12, #13) are disfavored.

Features starting with $e*$ are query-independent path features. We can see that well cited papers are generally favored (#9). Since the number of papers published is increasing every year, feature #14 actually disfavors old papers.

Features of the form "> *XXX*" are popular entity biases on specific entities. Features of the form "*XXX* > *XXX*" are conditional popular entity biases that associate a query entity with a target entity. We can see that papers about specific genes (e.g. CAL4, CYC1) often cite specific early works (#10, #11).

Table 4 shows a subset of features for a PRA+qip+pop model trained for the venue recommendation task on the fly data. We can see that different journals have different preferred topics (#7–#9), and some journals are less likely to accept drosophila related papers (#11–#13). Although journals of old papers are disfavored (#14), journals of popular papers are favored (#6). Interestingly, journals with many on-topic first authors (#2) are more favored than those with just any on-topic authors (#3).

4.3 Main results

Table 5 compares the effectiveness of different ranking algorithms on all four tasks and two copora. We can see that PRA performs significantly better than RWR under most tasks. The query-independent path experts (PRA+qip) manage to improve over basic PRA model in all tasks, and especially in reference recommendation and expert finding tasks. The popular entity experts (PRA+pop) also manage to improve over basic PRA model in all tasks, and the different is very significant on yeast tasks.

Table 5 Compare baseline RWR with PRA and its two extensions: query-independent path experts (+qip) and popular entity experts (+pop). The tasks are Venue Recommendation (Ven), Reference Recommendation (Ref), Expert Finding (Exp), and Gene Recommendation (Gen). Performances are measured by MAP, and the numbers in the brackets are relative improvement (%) over the trained RWR model. Except these†, all improvements are statistically significant at $p < 0.05$ using paired t-test

Corpus	Task	RWR	PRA			
		Trained	Trained	+qip	+pop	+qip +pop
Yeast	Ven	44.2	45.7 (+3.4)	46.4 (+5.0)	48.7 (+10.2)	49.3 (+11.5)
Yeast	Ref	16.0	16.9 (+5.6)	18.3 (+14.4)	19.1 (+19.4)	19.8 (+23.8)
Yeast	Exp	11.1	11.9 (+7.2)	12.4 (+11.7)	12.5 (+12.6)	12.9 (+16.2)
Yeast	Gen	14.4	14.9 (+3.5)	15.1 (+4.9)	15.1 (+4.9)	15.3 (+6.3)
Fly	Ven	48.3	50.4 (+4.3)	51.1 (+5.8)	50.7 (+5.0)	51.7 (+7.0)
Fly	Ref	20.5	20.8 (+1.5)†	21.0 (+2.4)	21.6 (+5.4)	21.7 (+5.9)
Fly	Exp	7.2	7.6 (+5.6)†	8.3 (+15.3)	7.9 (+9.7)	8.5 (+18.1)
Fly	Gen	19.2	20.7 (+7.8)	21.1 (+9.9)	21.1 (+9.9)	21.0 (+9.4)

5 Conclusion and future work

We proposed a novel method for learning a weighted combination of path-constrained random walkers, which is able to discover and leverage complex path features of relational retrieval data. We also evaluate the impact of using query-independent path features, and popular entity features which can model per entity characteristics. Our experiment on several recommendation and retrieval tasks involving scientific publications shows that the proposed method can significantly outperforms traditional models based on random walk with restarts.

We are very interested in the generalization from simple relations to hyper-relations which are mappings from possibly more than one source types. For example, there is much incentive to express the AND relation (Balmin et al. 2004): e.g. consider the task of finding papers that are both written by certain author and recent. However, model complexity will be a major concern. Efficient structure selection algorithm is very important to make a system practical.

Furthermore, we are interested in algorithms that introduces new entities and edges to the graph. This can potentially be useful to improving retrieval quality or efficiency. For example, new entities can represent subtopics of research interests, and new links can represent memberships from words, authors or papers to these subtopics. In this way, a model might be able to replace some long paths which we have shown in the experiment with relatively shorter and more effective paths associated with the introduced structures.

References

Agarwal, A., Chakrabarti, S., & Aggarwal, S. (2006). Learning to rank networked entities. In *KDD*

Andrew, G., & Gao, J. (2007). Scalable training of l^1-regularized log-linear models. In *ICML*.

Arnold, A., & Cohen, W. W. (2009). Information extraction as link prediction: using curated citation networks to improve gene detection. In *ICWSM*.

Aslam, J. A., Kanoulas, E., Pavlu, V., Savev, S., & Yilmaz, E. (2009). Document selection methodologies for efficient and effective learning-to-rank. In *SIGIR*.

Balmin, A., Hristidis, V., & Papakonstantinou, Y. (2004). Objectrank: authority-based keyword search in databases. In *VLDB*.

Chakrabarti, S., & Agarwal, A. (2006). Learning parameters in entity relationship graphs from ranking preferences. In *PKDD*.

Diligenti, M., Gori, M., & Maggini, M. (2005). Learning web page scores by error back-propagation. In *IJCAI*.

Dou, Z., Song, R., & Wen, J.-R. (2007). A large-scale evaluation and analysis of personalized search strategies. In *WWW*.

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning*. Cambridge: MIT Press.

He, Q., Pei, J., Kifer, D., Mitra, P. & Giles, C. L. (2010). Context-aware citation recommendation.

Liben-Nowell, D., & Kleinberg, J. (2007) The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*.

Lupu, M., Piroi, F., Huang, X., Zhu, J., & Tait, J. (2009). Overview of the TREC 2009 chemical IR track. In *TREC-18*.

Minkov, E., & Cohen, W. W. (2008) Learning graph walk based similarity measures for parsed text. In *EMNLP*.

Minkov, E., Cohen, W. W., & Ng, A. Y. (2006). Contextual search and name disambiguation in email using graphs. In *SIGIR*.

Nie, Z., Zhang, Y., Wen, J.-R., & Ma, W.-Y. (2005). Object-level ranking: bringing order to web objects. In *WWW*.

Pavlu, V. (2008). Large scale IR evaluation. *PhD thesis, Northeastern University, College of Computer and Information Science*.

Perkins, S., Lacker, K., & Theiler, J. (2003). Grafting: fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*.

Toutanova, K., Manning, C. D., & Ng, A. Y. (2004). Learning random walk models for inducing word dependency distributions.

Tsoi, A. C., Morini, G., Scarselli, F., Hagenbuchner, M., & Maggini, M. (2003). Adaptive ranking of web pages. In *WWW*.

White, R. W., Bilenko, M., & Cucerzan, S. (2007). Studying the use of popular destinations to enhance web search interaction. In *SIGIR*.