Term-Specific Eigenvector-Centrality in Multi-Relation Networks

François Bry, Fabian Kneißl, Klara Weiand
Institute for Informatics
Ludwig-Maximilian University of Munich
80538 Munich, Germany

Tim Furche
Oxford University Computing Laboratory
Wolfson Building, Parks Road,
Oxford, OX1 3QD, UK

Abstract: Fuzzy matching and ranking are two information retrieval techniques widely used in web search. Their application to structured data, however, remains an open problem. This article investigates how eigenvector-centrality can be used for approximate matching in multi-relation graphs, that is, graphs where connections of many different types may exist. Based on an extension of the PageRank matrix, eigenvectors representing the distribution of a term after propagating term weights between related data items are computed. The result is an index which takes the document structure into account and can be used with standard document retrieval techniques. As the scheme takes the shape of an index transformation, all necessary calculations are performed during index time.

Keywords: approximate matching, structured data, eigenvector, indexing methods, keyword search, pagerank, social networks, business process models, semantic web, ranking

1 Introduction

Approximate (or *fuzzy*) matching is a technique widely used when data are to be retrieved that resemble a given sample, called a *query*: the search engine sifts through the data and retrieves all items approximately matching the query. The assessed relevance of data items to the query is nuanced, allowing to differentiate looser results from results that adhere strictly to the query. For example, returned data items might not contain a word occurring in the query, but instead its plural or a synonym or a slightly differently spelled word. Approximate matching is often used in conjunction with a ranking scheme that orders the answers with respect to a relevance measure. The result of such a search is then an ordered list of data

items that are judged to be sufficiently relevant to the query, regardless of whether or not they constitute a strict match.

This article investigates how eigenvector-centrality can be used for approximate matching when the data searched takes the shape of a multi-relation network where data items may be related to each other by one or more of several different types of relations. The approximate matching considered here is furthermore term-based, that is, terms expressed in the query are to be approximately matched with terms in the data searched.

Approximate querying in multi-relation networks and structure-aware document querying are closely related. Indeed, a portion of a document might serve as a relevant approximate answer because of its structural embedding within the document. The same is true for documents inter-related by hyperlinks: A document might constitute a convenient approximate answer because of its hypertextual surrounding. As far as approximate querying is concerned, structure does not necessarily only refer to the document structure (in sections, paragraphs, etc.) and to the hyperlink structure of a collection of documents. Instead, structure might refer as well to semantic relationships between (components of) documents or to social relationships in a digital social network. Furthermore, the structure that may be considered in computing approximate answers can be a mix of document structure, semantic (or ontological) structure, and social structure. In such cases, multi-relation networks are indispensable.

Combining document structure with other relations opens the path to term-specificity: A term in a data item (such as a document in a hypertext application or a person's description in a social network) may be considered a vertex of the network related to other vertices not only indirectly through the relations of its embedding document, but also directly through its owns relations.

This article presents the concept of *term-specific multi-relation eigenvector centrality*, a scheme that uses the structural properties of vertices and networks to re-estimate the weights of terms occurring in data items such as documents or personal pages, thereby realizing term-based and multi-relation-based fuzzy matching and ranking for networked data.

The approach presented in this article is based on *content graphs* (see Section 4), a simple, but flexible model for multi-relation networks that arguably captures a wide range of knowledge management systems and (tree- or graph-shaped) structured data applications. Content graphs consist of vertices, representing data items, and edges, representing (structural or semantic) relationships between vertices. Both vertices and edges in a content graph are typed and the strength of the relation depends on the weight assigned to the type of connection, that is, the types of the vertices and of the edge connecting them. Edge weights can be seen to represent the degree of (structural or semantic) relatedness indicated by a connection of a specific type. For example, in a hypertext network, the connection between a web page and a tag that has been assigned to it can be seen to be closer than that between two web pages where one links to the other – tags are a kind of semantic annotation describing content, while web pages may link to other web pages without their content being similar or even related. Consequently, the approach presented in this article can accommodate rich graphs that represent, for example, both hypertext linking as well as semantic and social relationships.

The approach presented in this article investigates how eigenvector-centrality can be extended to yield a term-specific ranking which takes multi-relational structure into account. The aim of the research reported in this article is to accommodate within a single eigenvector-centrality measure rich graphs with different types of vertices and edges representing for example hypertextual linking, document structure, semantic relationships, and social connections.

In contrast to many other fuzzy matching approaches for structured data (see Section 3), the approach presented in this article relies solely on modifying term weights in a data index and requires no further computations at query time. Furthermore, it can be implemented using existing information retrieval tools such as Lucene. Finally, as the modified term weights computed by the proposed approach represent how well a data item is connected to others within the network, with the proposed approach a separate adjustment of the answer ranking as in PageRank is not necessary.

Though the matrix used for computing the proposed novel eigenvector-centrality measure is inspired by the Google Matrix used in PageRank, the approach presented here deviates from, and generalizes, PageRank in two significant aspects:

1. Our approach uses the network structure not only for ranking but also for answer approximation: it finds even relevant answers that do not explicitly contain the considered keyword. To this aim, we propagate terms between vertices using structural relationships and an *informed leap* based on the term weight distribution rather than a random leap. The informed leap is similar to personalised PageRank's random leap probability which is based on users' preferences, but with different goal and outcome; where personalized PageRank only computes a ranking, our approach computes a new vector-space representation of the data items that integrates propagated term weights with relevance based on the relations of a data item.

2. Since the term weight distribution and thus the informed leap is in general different for each term, the approach needs to consider term propagation separately for each term. Though this increases index time compared to PageRank, the term-dependent computation is only a small part of the overall indexing time and can be computed independently for each term (and is thus easily parallelized).

Contributions

The approach described in this article improves on existing approaches for approximate keyword search over structured data (see Section 3) in several respects:

(1) It is based on content graphs, a simple, but flexible novel model for multi-relation networks that captures a wide range of applications. Section 2 discusses a number of areas in which the application of our approach can improve search results and help users locate information relevant to them. Content graphs are introduced in Section 4. In Section 4.3, we how social networks can be captured as content graphs.

(2) Another contribution of this article is the matrix of Section 5 for propagating term weights in a multi-relation network. This matrix allows the propagation of term weights at *index time* and yields a modified vector space representation that can be used by any information retrieval engine based on the vector space model.

(3) In a first experimental evaluation, we show that an implementation of our approach behaves as expected, returning not only strict matches, but also other relevant results, while also improving the result ranking.

2 Motivating Scenarios

Data retrieval in multi-relation networks is relevant to a wide range of application domains and different types of data, that is, networks. The authors of this article are convinced that the eigenvector-centrality measure proposed in this article can improve search results in a wide range of applications. Three application scenarios described in the following sustain this conviction.

2.1 Social Networks

Over the past years, digital social networks have evolved from relatively simple networks based on a small and limited number of relations to rich multi-relation networks encompassing more and more relations. The various types of relations serve to express different kinds of social relationships (for example different kinds of personal relationships and group memberships) as well as interests in various subjects. Each service added to a digital social network usually enriches it with additional relations. Furthermore, the activity on the digital social network can be tapped, yielding additional relations like co-movements, useful in tracking various social phenomena such as influence. So far, however, the search facilities offered on digital social networks mostly perform exact matching and do not fully exploit the relation richness of the networks. Thus, much of the information available on the digital social network remains unused.

With the social networks analyzed in social sciences and biology, the situation is different. The multiple relations of a social network are usually all considered for network analysis. Indeed, if not for being exploited the data would not have been collected in the first place. However, the standard social network analysis methods make it difficult, or at least rather technical, to consider more than one type of relation at a time. Thus, a global analysis of the multi-relation network is difficult.

The eigenvector-centrality approach proposed in this paper makes it possible to consider multi-relation networks as a whole. Furthermore, it provides approximate querying and term-specific ranked answers. The authors of this article are therefore convinced that it provides a very appropriate basis both for search services in the current rich and evolving digital social networks and for global analysis methods tuned to multi-relation social networks.

2.2 Semantic Web

The Semantic Web vision is that of web pages embedded in a multi-relation semantic network with each type of relation expressing different and

complementary semantic notions. The RDF language, which is aimed to be an essential vector of the Semantic Web vision, is basically a language for expressing all binary semantic relations application developers might wish.

So far, querying on the Semantic Web is limited to either logic-based deduction (with logic languages such as the various flavors of OWL), or database-like querying of RDF specifications. Approximate answers taking into account the semantic embedding of web pages or RDF concepts do not yet exist.

The eigenvector-centrality approach proposed in this paper provides a convenient basis for semantic-aware approximate querying on the Semantic Web because it copes with multi-relation networks, and because it delivers ranked and term-specific answers.

2.3 Business Process Management

Business Process Management (BPM) is about composing relatively generic software components into complex systems relying on rather simple descriptions of the software components. In other words, BPM can be seen as gathering components from term-based specifications. BPM, however, does not assume a global approach to specifying software components one must perfectly adhere to. BPM is for the open Web where even if standards and conventions are agreed upon, no authority can, or wishes, to enforce them. Thus, BPM is much about delivering approximate answers to queries.

In BPM, directed graphs of activities, events and decision points are used to describe the work flow in a process such as a loan application. Previous approaches to querying BPM graphs such as [14, 4] rely on query languages with expressive graph pattern languages, e.g., to express queries such as "In every process execution (path) from an *implemented component* activity to an *integrate component* activity there must be a *perform test component* activity.". However, formulating such queries is difficult and prone to omitting all constraints relevant to a given, e.g., "perform test component". Approximate keyword queries can serve as a means to explore large BPM graphs with little knowledge about the structure, e.g., a search for "test component" will return not only the actual test components but also components that most of these rely on. This allows engineers unfamiliar with a BPM to gain a first insight into the model. With ever growing process models [14] and thousands of such models managed by a single company [4], efficient exploration of BPM models is essential. We believe that the approach proposed in this article provides a very convenient basis for exploratory search targeted at BPM. Arguably, ranked approximate answers taking the global multi-relation nature of software specifications into account are an essential ingredient for such search.

2.4 Combined Use Cases

Use cases like the three outlined above can be combined. For example, the search in a digital social network and the search for software components can be improved with ontologies. With the approach proposed in this article, adding one or several ontologies to a multi-relation network is no more than an extension of this network with new vertices and new edges between existing and new vertices. The

eigenvector-centrality computation proposed in this article requires no adaptation and is directly applicable to the extended network.

3 Related Work: Approximate Matching on Structured Data

Approximate matching on data structure has been researched mainly in the context of XML [26]. The majority of work belongs to one of two classes: Tree edit distance approaches [25, 10, 6, 22] and related approaches [1, 2, 24] are based on a quantification of the similarity between XML trees through the number of steps and types of operations needed to eliminate the differences between them. These approaches require expensive calculations at query time. Another class of approaches [19, 8, 23] aims, as ours, to adapt the vector space model, a well-established IR technique, to the application on XML data.

Further, Anh et al. [3] use activation propagation for approximate matching over XML structure. This approach resembles ours in that activation propagation and the vector space are used to realize approximate matching over structure. However, here, propagation happens upon query evaluation and is unidirectional.

The approach presented in this paper differs from the majority of approximate matching approaches including those mentioned above in the following in several important aspects:

- It does not realize fuzzy matching by defining a structural distance function and ranking results according to how close they are to a strict match. Instead, it uses the structure of the data to determine which terms are relevant to a document, regardless of whether or not they explicitly occur in it. As a consequence, not only are new matches introduced, but strict matches may also be re-ranked depending on their structural connections.
- It is designed for *graph-shaped data* rather than purely hierarchical data like the XML-based approaches discussed in the following.
- It can be used with any information retrieval engine based on the vector space model. The only modification to the evaluation process is the computation of the actual term weights. Otherwise existing technology (such as Lucene or similar search engines) can be utilized. In particular, the propagation matrix is query independent and can be computed at *index time*. No additional computations such as query transformations are needed during query evaluation.

PageRank: Where the above approaches for approximate (keyword) search and querying on XML data are similar in aim, PageRank is closely related to our approach in technique, but differs considerably in aim and scope. The original PageRank article [7] suggests to exploit anchor-tags for web search. The anchor text of a link to a web page is treated as if it is part of the text of that web page. This suggestion can be seen as a special case of the approach suggested in this paper where the only kind of propagation is that from anchor tags to linked pages and where links weights are ignored. The application of this approach is limited to anchor tags and does not apply to general graphs or generalize to different link

types. However, there are a number of extensions [5] of the original PageRank that share more of the characteristics of the eigenvector centrality approach presented in this article.

PageRank is based on the intuition that a link from one webpage to another can be seen as an endorsement of the linked page. A page then is important if many important pages link to it, even more so if the number of pages linked to by these important pages is low. This idea is implemented by transforming the link graph into a transition matrix which is augmented by a random leap component that ensures that the probability to transition from any state, that is, page, to any other state is nonzero. The contribution of the random leap relative to the transition values is determined by the factor α. As a consequence of introducing the random leap and setting α to a non-zero value, the normalized matrix is stochastic and strictly positive and the principal eigenvector (called PageRank vector) for eigenvalue 1 exists and is unique. In standard PageRank, the random leap operates using a uniform distribution, that is, the likelihood to transition to a state is identical for all states. For a set of N pages, the corresponding leap or teleportation vector can be seen to consist of N entries with value $\frac{1}{N}$. Brin and Page [7] point out the possibility to realize a personalized version of PageRank by using a non-uniform leap vector.

Several schemes for improving the scalability of personalized PageRank have been presented in recent years [17, 18, 15, 20] which are discussed in the following.

Topic-sensitive PageRank [15] builds on the idea of a personalized teleportation vector by introducing a number of topic-specific leap vectors, each assigning a uniform value to all pages relevant to the respective topic and 0 otherwise. The topic-dependent importance scores for each page are calculated offline. At query time, a weighted classification of the query into topics is computed. A query-specific PageRank can then be calculated as a mixture of topic-specific scores. The motivation behind topic-sensitive PageRank is to avoid generally important pages getting highly ranked despite not containing information relevant to the query.

Query-dependent PageRank [20] is another extension of PageRank which is based on the idea that webpages matching a query that are connected to other matching webpages should be ranked higher. The PageRank algorithm is adapted in such a way that the probability of a transition from one webpage to another is determined by how relevant the target webpage is to the query. Towards this end, both the distribution underlying the leap vector as well as the mode of calculating transition probabilities are adjusted. In both cases, the probability to transition to a webpage is given as the proportion between that webpage's relevance score and the sum of all matching pages' relevance scores. When a webpage has no non-zero outlinks, the leap vector is used to jump to another webpage. The transition matrix is not strictly positive and vertices which do not contain the relevant term are ignored. The PageRank vector is calculated for each term at index time, the scores for each term in the query are combined upon query evaluation.

The approach presented in this article differs from the methods described in various ways. In contrast to PageRank (but similar to topic-sensitive and query-dependent PageRank), several matrices and eigenvectors are calculated, namely one per term, each using a term-dependent leap vector. In contrast to topic-sensitive PageRank as well as PageRank, the leap vectors do not use a uniform distribution.

Most importantly, differences to the three described approaches consist in the following ways:

- None of the approaches implement approximate matching over structured data or generally add additional relevant results; they are purely approaches to *ranking* sets of webpages.
- The *assignment of edge weights is more flexible* in that edge weights can be set explicitly and individually or different weights can be chosen depending on the type of edge. In contrast, in PageRank and topic-sensitive PageRank edge weights are uniform and in query-dependent PageRank, edge weights are derived from keyword matches.
- While our approach can be used on webpages with linking as the only relation between pages, its versatile and extensible data model allows for the application to many *other types of graph-shaped data* such as a fine-grained modeling of structured web data.
- The *probability of a leap is variable depending on the number and weight of outgoing edges of a vertex*, thus encoding the intuition that a user jumps to a new page when he cannot find what he is looking for by following links.

ObjectRank [16] is a system for authority-based ranking for keyword search in databases that, like our eigenvector centrality approach, uses PageRank to exploit the connections between data items for propagating authority with respect to a keyword across a data graph. Given a database modeled as a labeled graph and a schema graph that assigns bidirectional authority transfer rates to the different types of edges, an Authority Transfer Data Graph is derived. The weight of a vertex with respect to a given keyword is then established by a modified version of PageRank where a random surfer walks across the graph either traversing edges or jumping to any of the vertices that literally contain the keyword. The probability for following any outgoing edge or jumping to one of the keyword vertices is steered by the damping factor d: The probability to follow an edge is given as the product of d and the authority transfer rate of the edge, while the probability to randomly jump to one of the vertices containing the keyword is $(1 - d)$.

While ObjectRank clearly shares characteristics, it differs in various significant ways and has various drawbacks:

- ObjectRank uses a binary measure to represent literal keyword containment. Differences in the frequency of the keyword between documents are ignored and do not factor into the ranking. However, term frequencies are an important factor for ranking, particularly in text-heavy areas of application where it is of high relevance whether a term only occurs once or is frequently used.
- There are no jumps to vertices that do not contain the keyword and those vertices can only be reached through an edge traversal. Therefore, ObjectRank cannot be applied to graphs that are not strongly connected. While this may be of less concern in the area of databases, this constraint severely limits the possibility of applying ObjectRank to other types of graph-shaped data such as web or wiki pages which are frequently not strongly connected.

- Unlike our approach, ObjectRank is not a simple modification of term frequency distributions. As such, it cannot be directly used with standard information retrieval engines and easily used in conjunction with the vector space model.
- Queries are limited to simple keywords combined using disjunction and conjunction and it remains unclear whether ObjectRank could be combined with more powerful query languages.
- The probability to make a leap is steered only by the damping factor d and thus remains constant regardless of whether there is a promising edge that could be followed or not. Moreover, due to the way that authority transfer weights are assigned and normalized, the probability of all possible events may not add up to 1, which is unintuitive in terms of the random surfer model and the idea of spreading authority.

There is a significant body of research [11, 9, 21] on ranking entities in, e.g., entity-relationship graphs. These approaches share the aim to rank connected items by considering not only local, but also global properties, viz. what other items they are related to. However, they differ from the approach presented in this article in two main aspects: (1) They require a new query engine with sophisticated, multi-part ranking functions, whereas our approach computes a modified vector space model that can be used with any existing vector-space IR system. (2) They are tailored to ranking of entities such as dates, prices etc. where our eigenvector centrality approach is tailored to domains such as semantic wikis or concept search in ontologies where the items of interest are self-contained and clearly identified.

The approach presented in this article can be applied to any type of structured data, including RDF ontologies. Ranking of RDF query results is discussed, e.g., in [12] and [13]. However, these works differ by focusing on general RDF data and by using statistical language models with limited propagation.

4 A Formal Model for Structured Data

In this section, we formally define a generic graph-based model of structured content that is capable of capturing the rich representation features of a wide range of multi-relation networks such as social networks, the Semantic Web etc. Furthermore, we extend this model with a weight function so that different strenghts of connections in the graph can be modeled.

4.1 Content Graph

A content graph is defined based on a **type structure** $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of types for data items and $\mathcal{E}$ the set of edge types.

Content graph: *For a given type structure $\mathcal{T}$, a **content graph** is a tuple $G = (V, E, type, T, w_T)$ where V is the set of vertices (or data items) in G and $E \subseteq V \times V \times \mathcal{E}$ its set of typed edges. $type : V \to \mathcal{V}$ assigns a type to each vertex. The textual*

content of data items is represented by a set T of terms and a function $w_t : V \times T \to \mathbb{R}$ that assigns a weight to each pair of a vertex and a term. We assume that the term weights for each vertex v form a stochastic vector (i.e., $\sum_{\tau \in T} w_t(v, \tau) = 1$).

By using different types of data items and edges, this model becomes very flexible and can be used for a variety of purposes without adaption. In a social network as described in Section 2.1, for example, $\mathcal{V}$ consists of person and group and $\mathcal{E}$ of friendship and member-of-group. Then, a link between a person and a group through a member-of-group connection can be weighted differently than a friendship connection and thus the propagation effect adapted adequately in its strength.

4.2 Weighted Propagation Graph

By extending the content graph with a weight function for edges, we ascertain that differences in the importance of connections are accounted for during the ranking process.

Weighted propagation graph: *A **weighted propagation graph** is a content graph extended with a function $w_e : E \to \mathbb{R}^2$ for assigning weights to edges.*

In this article, we set edge weights according to the type of the edge and the types of the two vertices connected by the edge. We call a weighted propagation graph *type-weighted*, if for any two edges $e_1 = (v_1, w_1, t), e_2 = (v_2, w_2, t) \in E$ it holds that, if $type(v_1) = type(v_2)$ and $type(w_1) = type(w_2)$, then $w_e(e_1) = w_e(e_2)$. In other words, the weights of edges with the same type and with start and end vertices of the same type respectively must be the same in a type-weighted propagation graph. In the following, we only consider such graphs.

4.3 Specific Adjustments for Social Networks

As the approach is not tailored to one specific type of network, to achieve the best results, adjustments have to be made to account for peculiarities of a specific type of multi-relation network. In the following, the formalization of social networks a content graphs is exemplarily discussed.

We define a social network as a graph consisting of vertices, representing e.g. persons, groups, company profiles, or single web pages, and edges between them that express, e.g., friendship, membership, employment, or authorship. The result of the algorithm presented in this paper depends on how the data items and connections are formalized in terms of the data model.

As an example, consider a social networking site like Facebook[1]. Here, users can describe themselves on their personal pages, join interest groups, publish content (e.g., text or photos), annotate it in form of comments, and keep in touch with friends. These multi-relational data make it difficult to locate relevant information through simple full-text search. For example, joined groups or comments on photos provide valuable information about a user, but data retrieval is only performed on single data items. Consequently, when searching for a specific topic of interest, e.g. *linguistics*, only the pages of users and groups are returned that directly contain

the queried term. In contrast, our propagation algorithm ensures that a user whose personal page does not contain the term *linguistics* but who is engaged in several groups about linguistics is returned as a result to the query.

Several approaches exist to formalize a social network in such a way that the algorithm presented in this article is applicable and can improve search results as discussed above. One could, for example, take only users and groups into account and attach information to these two data items, allowing group memberships and friendships as relations. The number of data items is of a manageable size and if the interest is focused on users or groups, the relevant pieces of information can be retrieved. Another possibility consists in adding comments and other published content to the graph structure and interlink all vertices accordingly. The exhaustive graph can produce more fine-grained result rankings as multiple relations are used and can be balanced more exactly. Furthermore, different queries can be evaluated that, for example, return photos that could be seen as best matching for a specific topic (defined by one or several terms) as they are posted in several groups of the topic and mentioned by people interested in this topic. Thus, the decision on what data items to include not only influences the result ranking but furthermore extends or restricts it.

Depending on the types of vertices in the graph, the extension with additional vertices and edges helps to improve the propagation of terms through the graph and can facilitate searching for certain data items. For example, when a query language is used that allows to select specific types of data items, empty comment data items should be added to users and groups which do not yet have a comment attached in case a search restricted only to comments is performed. Then, appropriate links have to be added to the graph.

One further important aspect of our approach consists in the propagation of *terms* through the graph. Therefore, terms appropriate for propagation have to be chosen from many possibilities. The terms in a web page yield a good starting point, and also title or tags are valid choices. The selection of these terms is dependent on the allowed search queries and should be adapted to these.

5 Computing the Matrix for the Propagation of Term-Weights

Based on the model above, we now formally define the propagation of term-weights over multi-relation networks represented in a content graph by means of an eigenvector computation.

As a common property of the scenarios described in Section 2, an association between data items always carries information that can be seen as contexual similarity. We thus suggest to exploit these relationships for approximate matching over multi-relation networks by using them to propagate the data items' contents. A data item thereby is extended by the terms contained in other data items it is related to. Then, standard information retrieval engines based on the vector space model can be applied to find and rank results oblivious to the underlying structure of term-weight propagation.

In analogy to the *random surfer* of PageRank, the term-weight propagation can be explained in terms of a *semi-random reader* who is navigating through the content graph looking for documents relevant to his information need expressed by

a specific term τ (or a bag of such terms). He has been given some—incomplete—information about which vertices in the graph are relevant to τ. He starts from one of the vertices and reads on, following connections to find other documents that are also relevant for his information need (even if they do not literally contain τ). When he becomes bored or loses confidence in finding more matches by traversing the structure of the graph, he jumps to another vertex that seems promising and continues the process.

To encode this intuition in the matrix, we first consider which connections are likely to lead to further matches by weighting the edges occurring in a content graph. Let $\mathbf{H}$ be the transposed, normalized adjacency matrix of the resulting graph. Second, we discuss how to encode, in the leap matrix $\mathbf{L}_\tau$, the jump to a *promising* vertex for the given term τ (rather than to a random vertex as in PageRank)

The overall matrix $\mathbf{P}_\tau$ is computed as (where α is the leap factor)

$$\mathbf{P}_\tau = (1-\alpha)\mathbf{H} + \mathbf{L}_\tau.$$

Each entry $m_{i,j} \in \mathbf{P}_\tau$, that is, the probability of transitioning from vertex j to vertex i, is thus determined primarily by two factors, the normalized edge weights of any edge from j to i and the term weight of τ in j.

5.1 Normalizing the Weighted Propagation Graph's Adjacency Matrix

Let $\mathbf{A}_w$ be the weighted adjacency matrix of a weighted propagation graph G defined in Section 4.2. Then we normalize and transpose $\mathbf{A}_w$ to obtain the transition matrix $\mathbf{H}$ for G as follows (where $outdegree(v)$ denotes the number of outgoing edges of v):

$$\mathbf{H} = \left(\frac{1}{outdegree(v_i)} \left(\mathbf{A}_w^T\right)_{i,j} \right)_{i,j}$$

By normalizing with the number of outgoing links rather than the total weight of the outgoing edges, edge weights are preserved to some extent. At the same time, vertices with many outgoing edges are still penalized. Normalization with the out-degree proved the most effective in our experiments compared to, e.g., normalizing with the maximum sum of outgoing term weights (over all vertices) or with the sum of outgoing term weights for each vertex. Different choices for normalization preserve different properties of the original matrix and, for other applications, a different choice of normalization may be advisable.

5.2 Informed Leap

Given a leap factor $\alpha \in (0, 1]$, a leap from vertex j occurs with a probability

$$P(\text{leap}|j) = \alpha + (1-\alpha)(1 - \sum_i \mathbf{H}_{i,j})$$

A leap may be *random* or *informed*. In a random leap, the probability of jumping to some other vertex is uniformly distributed and calculated as $l^{\text{rnd}}(i,j) = \frac{1}{|V_d \cup V_t|}$ for each pair of vertices (i,j).

An informed leap by contrast takes the term weights, that is, the prior distribution of terms in the content graph, into account. It is therefore term-dependent and given as $l^{\inf}_{\tau}(i,j) = \frac{w_t(i,\tau)}{\sum_k w_t(k,\tau)}$ for a term $\tau \in \mathcal{T}$. Thus, when the probability of following any of the outgoing edges of a vertex decreases, in turn a leap becomes more likely.

In preliminary experiments, a combination of random and informed leap, with heavy bias towards an informed leap, proved to give the most desirable propagation behavior. The overall leap probability is therefore distributed between that of a random leap and that of an informed leap occurring according to the factor $\rho \in (0,1]$, which indicates which fraction of leaps are random leaps.

Therefore, we obtain the leap matrix $\mathbf{L}_\tau$ for term τ as

$$\mathbf{L}_\tau = \Big(P(\text{leap}|j) \cdot \big((1-\rho) \cdot l^{\inf}_{\tau}(i,j) + \rho \cdot l^{\text{rnd}}(i,j)\big) \Big)_{i,j}$$

5.3 Properties of the Propagation Matrix

Definition 5.1: *Let $\alpha \in (0,1]$ be a leap factor, $\mathbf{H}$ be the normalized transition matrix of a given content graph (as defined in Section 4.2) and $\mathbf{L}_\tau$ the leap matrix (as defined in Section 5.2) for $\mathbf{H}$ and term τ with random leap factor $\rho \in (0,1]$. Then the matrix $\mathbf{P}_\tau$ is the matrix*

$$\mathbf{P}_\tau = (1-\alpha)\mathbf{H} + \mathbf{L}_\tau.$$

Theorem 5.2: *The matrix $\mathbf{P}_\tau$ for any content graph and term τ is column-stochastic and strictly positive (all entries > 0).*

Proof: It is easy to see that $\mathbf{P}_\tau$ is strictly positive as both α and ρ are > 0 and thus there is a non-zero random leap probability from each vertex to each other vertex.

$\mathbf{P}_\tau$ is column stochastic, as for each column j

$$\begin{aligned}
\sum_i (\mathbf{P}_\tau)_{i,j} &= \sum_i \big((1-\alpha)\mathbf{H}_{i,j} + (\mathbf{L}_\tau)_{i,j}\big) \\
&= (1-\alpha)\sum_i \mathbf{H}_{i,j} + \Big(\big(\alpha + (1-\alpha)(1 - \sum_l \mathbf{H}_{l,j})\big) \cdot \\
&\quad \big((1-\rho) \cdot \underbrace{\sum_i l^{\inf}_{\tau}(i,j)}_{=1} + \rho \underbrace{\sum_i l^{\text{rnd}}(i,j)}_{=1}\big) \Big) \\
&= (1-\alpha)\sum_i \mathbf{H}_{i,j} + (1-\alpha)(1 - \sum_l \mathbf{H}_{l,j}) + \alpha \\
&= 1 - \alpha + \alpha = 1
\end{aligned}$$

□

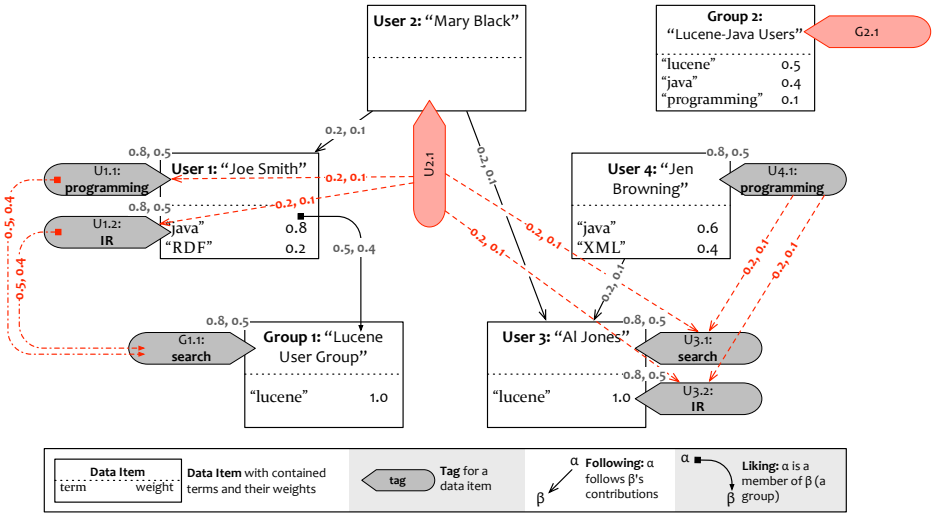

Figure 1: Weighted propagation graph of a social network. Edge weights are given as pairs of numbers, one for traversing the edge in its direction, one for traversing it against its direction.

Corollary 5.1: *The matrix* $\mathbf{P}_\tau$ *has eigenvalue* 1 *with unique eigenvector* $\mathbf{p}_\tau$ *for each term* τ.

The resulting eigenvector $\mathbf{p}_\tau$ gives the new term-weights for τ in the vertices of the content graph after term-weight propagation. It can be computed, e.g., using the power method (which is guaranteed to converge due to Theorem 5.2).

The vector space representation of the content graph *after term-weight propagation* is the document-term matrix using the propagation vectors $\mathbf{p}_\tau$ for each term τ as columns.

6 Social Networks – a Proof of Concept

In order to confirm that the application of the scheme presented here extends and ranks results in a useful manner, a prototype implementation was devised and small-scale experiments were conducted in order to verify the algorithm's viability.

In Figure 1, the weighted propagation graph of a social network consisting of four users and two groups is displayed. Each data item (i.e., user, group, or tag) contains a list of terms of a certain term weight which relate to information displayed on the user's or group's page; in case of tags, the term weight 1 for the only term is omitted. Furthermore, tags, indicating the users' interests and the groups' topics, are attached to users and groups. The remaining edges represent a *"following"* resp. *"liking"* relationship with the former receiving more weight than the latter as the authors derive *"liking"* to be more important in terms of its propagation influence. The tags *U2.1* and *G2.1* have been added to the weighted propagation graph as part of the specific adjustments process (see Section 4.3) in

	RDF	*XML*	*IR*	*programming*	*java*	*lucene*	*search*
U1	**0.46**	0.03	**0.11**	**0.11**	**0.26**	**0.08**	0.07
U1.1	**0.11**	0.02	0.05	**0.23**	0.07	0.04	0.07
U1.2	**0.11**	0.02	**0.24**	0.04	0.07	0.04	0.07
U2	0.03	0.02	0.03	0.02	0.03	0.03	0.02
U2.1	0.03	0.02	0.04	0.03	0.02	0.02	0.03
U3	0.02	**0.08**	**0.09**	0.04	0.04	**0.22**	**0.09**
U3.1	0.02	0.04	0.03	0.04	0.02	0.06	**0.22**
U3.2	0.02	0.04	**0.23**	0.04	0.02	0.06	0.03
U4	0.01	**0.53**	0.02	**0.08**	**0.17**	0.02	0.02
U4.1	0.01	**0.12**	0.02	**0.22**	0.04	0.02	0.02
G1	**0.10**	0.02	0.05	0.05	0.06	**0.21**	**0.09**
G1.1	0.06	0.02	0.06	0.06	0.04	0.06	**0.24**
G2	0.01	0.02	0.01	0.03	**0.11**	**0.11**	0.01
G2.1	0.01	0.01	0.01	0.02	0.03	0.03	0.01

Table 1 Document-term matrix after term-weight computation with all values ≥ 0.08 highlighted.

order to ease direct propagation of terms between tags. In the same step, (dashed) edges are added between tags which are indirectly connected through the data items they are attached to.

The evaluation of the approach for this examplary multi-relation network and the calculation of the modified term weights for all terms appearing in the network results in the document-term matrix shown in Table 1. To verify the veracity of our approach, let us consider a number of desirable properties an approach to fuzzy matching on a multi-relation network should exhibit:

1. Users or groups containing a term directly (e.g., *"java"*) with a significant term weight should still be ranked highly after propagation. This should hold to guarantee that direct search results (that would have been returned without fuzzy matching) are retained.

   Indeed, User 1 and 4 and Group 2, all containing *"java"*, are highest ranked for that term, though the tags of User 1 come fairly close. This result is desired, as User 1 contains *"java"* with high term weight and tag-user associations are among the closest relations.

2. A search for a term τ should also yield data items not containing τ but data items directly connected to ones containing τ. Their rank should depend on the weight of τ in the connected data item and the type (and thus propagation strength) of the connection.

   Looking at the results for *"lucene"*, "Joe Smith" receives weight from the liked "Lucene User Group" and thus becomes a valid search result as well.

3. Searching for *"IR programming"* should also rank highly data items that do not include these terms, but that are tagged with *"IR"* and *"programming"*.

User 1 is such a case and is indeed the next highest ranked data item for such a query after the three data items directly containing *"IR"* or *"programming"* (using either boolean or cosine similarity).

Though this evaluation can, by design, only illustrate the effectiveness of the proposed term-weight propagation approach for fuzzy matching, we believe that it is a strong indication that it will prove efficient and effective also for larger and more diverse document collections.

Perspectives and Conclusion

In this paper, we have proposed a unique approach to fuzzy matching that combines the principles of structural relevance from approaches such as PageRank with the standard vector space model. Its particular strength is that it runs entirely at index time and results in a modified vector space representation. We show that it is applicable in a wide-range of scenarios, including social network analysis. It allows us to find, e.g., a user group based on interests of its members.

There are two major open questions in evaluating the proposed approach:

1. A **large-scale evaluation** in multiple scenarios of the proposed approach, including a user study. Preliminary evaluations point toward the veracity of the approach but further experimental validation is needed. In particular, the effect of the shape of the data (e.g., social network) on the quality of the approach is to be investigated.

   In particular, we are currently estimating the values for α and ρ as well as for the edge weights manually rather than by empirical observation. A guide to choosing these values might be possible to derive from studying the behaviour of our approach on data with different characteristics.

2. A main concern about the approach is its **scaling** to very large datasets. First, we need to compute one propagation matrix for each term. Fortunately, in homogeneous data sets after an initial warm-up phase, the number of distinct terms increases only slowly with increasing data size. We plan to investigate the effect of this observation on our approach as well as a number of simple heuristics for avoiding the computation for low entropy terms and to cut off very low term weights.

   The latter is also necessary to maintain the high performance of typical vector space indices such as inverted files that rely on most terms occurring only in few documents. We plan to investigate of our approach with various cut-off points on the query performance of typical keyword queries.

In addition, there are a number of ways to improve the approach beyond what is outlined in this paper:

(1) Edge values, in particular, could also be amenable to various machine learning approaches, using, for example, average semantic relatedness as a criterion, or to semi-automatic approaches through user-feedback.

(2) Any fuzzy matching approach suffers from non-obvious *explanations* for returned answers: In the case of a boolean query semantics, the answer is obvious,

but when term propagation is used, a document might be a highly-ranked query result without as much as containing any query terms directly. In this case, providing an explanation, for example that the document in question is closely connected to many documents containing query terms, makes the matching process more transparent to users. However, automatically computing good, minimal explanations is far from a solved issue.

(3) We want to apply this approach to a wide range of applications: Ranking and fuzzy matching in *Semantic wikis* where documents, authors, and meta-data are connected with a number of relations and propagation allows authors to be considered relevant answers for keywords that occur frequently in documents they authored. Similarly, in *image search* for art history we can propagate tags (assigned, e.g., by users in a game-with-a-purpose) based on known relations between artefacts such as created by the same artist in the same period. Here, propagation strongly depends on the tag type (e.g., style of painting vs. content of painting). Finally, in *object search* we want to find structured objects, often extracted from web pages or deep web databases, based on attributes, but often also simple keyword queries (since users have little knowledge of the schemata used). Propagation is very useful here as information is often incomplete, but structurally related entities can provide a way of making an informed guess: E.g., if a users searches for a flat that is not a house share in Oxford, we would rank results from an agency with many house shares lower than those from an agency with few house shares even if we have no knowledge whether that specific flat is a house share or not.

References

[1] Sihem Amer-Yahia, SungRan Cho, and Divesh Srivastava. Tree pattern relaxation. In *Int'l Conf. on Extending Database Technology*, 2002.

[2] Sihem Amer-Yahia, Laks V. S. Lakshmanan, and Shashank Pandit. FleXPath: flexible structure and full-text querying for XML. In *ACM SIGMOD Int'l Conf. on Management of Data*, 2004.

[3] Vo Ngoc Anh and Alistair Moffat. Compression and an IR approach to XML retrieval. In *INEX Workshop*, 2002.

[4] Ahmed Awad and Sherif Sakr. Querying graph-based repositories of business process models. In Masatoshi Yoshikawa, Xiaofeng Meng, Takayuki Yumoto, Qiang Ma, Lifeng Sun, and Chiemi Watanabe, editors, *Database Systems for Advanced Applications*, volume 6193 of *LNCS*. Springer, 2010.

[5] P. Berkhin. A survey on PageRank computing. *Internet Mathematics*, 2(1), 2005.

[6] Philip Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3), 2005.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Int'l World Wide Web Conf.*, 1998.

[8] D. Carmel, YS Maarek, Y. Mass, N. Efraty, and GM Landau. An extension of the vector space model for querying XML documents via XML fragments. In *SIGIR Workshop on XML and Information Retrieval*, 2002.

[9] Soumen Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 571–580, New York, NY, USA, 2007. ACM.

[10] S.S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change detection in hierarchically structured information. In *ACM SIGMOD Int'l Conf. on Management of Data*, 1996.

[11] Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. Entityrank: searching entities directly and holistically. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 387–398. VLDB Endowment, 2007.

[12] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. Language-model-based ranking for queries on rdf-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, Hongkong, China, 2009. ACM.

[13] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, and Gerhard Weikum. Searching RDF Graphs with SPARQL and Keywords. *IEEE Data Enginerring Bulletin*, 33(1), 2010.

[14] Georg Grossmann, Michael Schrefl, and Markus Stumptner. Modelling inter-process dependencies with high-level business process modelling languages. In *APCCM*, 2008.

[15] T. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4), 2003.

[16] Heasoo Hwang, Vagelis Hristidis, and Yannis Papakonstantinou. Objectrank: a system for authority-based search on databases. In *SIGMOD Conference*, pages 796–798, 2006.

[17] G. Jeh and J. Widom. Scaling personalized web search. In *Int'l World Wide Web Conf.*, 2003.

[18] S. Kamvar, T Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing PageRank. Technical report, Stanford, 2003.

[19] Jaroslav Pokorný. Vector-oriented retrieval in XML data collections. In *Databáze, Texty, Specifikace a Objekty*, 2008.

[20] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. *Advances in Neural Information Processing Systems*, 2, 2002.

[21] Henning Rode, Pavel Serdyukov, and Djoerd Hiemstra. Combining document- and paragraph-based entity ranking. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 851–852, New York, NY, USA, 2008. ACM.

[22] T. Schlieder. Similarity search in XML data using cost-based query transformations. In *ACM SIGMOD Web and Databases Workshop*, 2001.

[23] Torsten Schlieder and Holger Meuss. Querying and ranking XML documents. *Journal of the American Society for Information Science and Technology*, 53(6), 2002.

[24] Dennis Shasha, Jason Tsong-Li Wang, Huiyuan Shan, and Kaizhong Zhang. Atreegrep: Approximate searching in unordered trees. In *Int'l Conf. on Scientific and Statistical Database Management*, 2002.

[25] K.C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3), 1979.

[26] Joe Tekli, Richard Chbeir, and Kokou Yetongnon. An overview on XML similarity: Background, current trends and future directions. *Computer Science Review*, 3(3), 2009.