A Novel Metric for Information Retrieval in Semantic Networks

Joshua L. Moore[1,2], Florian Steinke[1], and Volker Tresp[1]

[1] Siemens AG, Corporate Technology, München, Germany
[2] Cornell University, Ithaca, NY
jlmo@cs.cornell.edu,{Florian.Steinke,Volker.Tresp}@siemens.com

Abstract. We propose a novel graph metric for semantic entity-relationship networks for solving two tasks. First, given a semantic entity-relationship graph such as for example DBpedia we find relevant neighbors for a given query node. Second, we search for paths between two given nodes in order to discover interesting links between the nodes. Compared to using the default step metric our approach yields more specific and informative results, as we demonstrate for two semantic web datasets. Moreover, we show that our proposed metric can intuitively be interpreted in terms of random walks. The distances are defined via paths that maximize the log-likelihood of a restricted round trip in such a random process. This also yields a link to the commute distance, which is highly plausible for the described tasks but prohibitively expensive to compute. In comparison, our metric can be calculated efficiently using standard graph algorithms, rendering the approach feasible also for the very large graphs of the semantic web.

Keywords: Entity-relationship graph, information retrieval, random walk, commute distance, graph metric, path finding

1 Introduction

Large entity-relationship (ER) graphs have recently become available on the semantic web. Sources like DBpedia (Auer et al., 2008), YAGO (Suchanek et al., 2007), OpenCyc[3] or Linked Life Data[4] (Momtchev et al., 2009) now encode useful information on large scale, and simple and efficient information retrieval methods for these data sources are a pressing need.

Given an ER graph, there are many interesting questions to be answered. We focus on the following two. First, given an entity node in the graph, e.g. a person or a category in DBpedia, which other nodes in the graph represent entities that are most likely to be useful in the context of the given query node? Answers might be other concepts that could be used to refine or extend an interactive search session. As another problem, consider selecting two nodes from the graph and asking how they are related. For example, via which people or concepts are Albert Einstein and Niels Bohr related in DBpedia? In the field of bioinformatics such

[3] http://www.cyc.com/opencyc
[4] http://linkedlifedata.com/

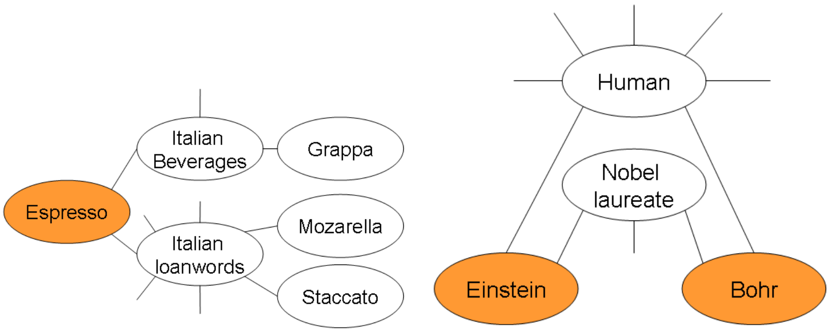

Fig. 1. Stylized examples of the two proposed tasks (data taken from DBpedia) and the associated challenges. In both examples there are links of equal length which are of very different informative value. Paths through nodes with many links are often too broad to be informative. Query nodes are marked orange, black lines denote links to nodes which are not depicted.

a link query between genes and diseases might well be used to discover unknown pathways from the existing literature (Antezana et al., 2009), given that such knowledge is extracted into semantic graph form as is done, for example, in Bundschus et al. (2008).

The two described tasks can both be solved with shortest-path search on the ER graph. Most simply, one assumes a step metric, i.e. one assigns every edge in the graph a unit cost, and computes the shortest-path distance from the query node to all other nodes for the first task, e.g. using Dijkstra's algorithm (Dijkstra, 1959), or the k-shortest paths connecting the two given nodes for the second task, e.g. by using the k-shortest path algorithm of (Yen, 1971). While this is straightforward and efficient to implement, it often returns highly irrelevant results. Consider for example a graph which contains a vertex which is connected to nearly all other vertices, such as a broad category that encompasses most entities in the graph. In the first problem we defined, this method would return the high-degree vertex with high rankings for almost any vertex queried. This lack of discrimination hinders context search, since a broad topic would lack specific relevance to any one node in the graph. Also, most of the vertices in the graph would be returned with distance two, although many of them are unrelated to the query. In the second problem, consider a database in which every person is connected to a "human" category node. In this case, the fact that Einstein and Bohr are both humans is less informative than the fact that they are both Nobel laureates, but the "human" and "Nobel laureates" nodes would have equal ranking. These problems are schematically depicted in Figure 1.

One way to allow for differently weighted nodes is to transfer PageRank-like concepts (Brin and Page, 1998) to ER graphs. However, a simple implementation of this might have an adverse effect in our setting: nodes are deemed popular and thus important if many links point to them. Thus high-level, highly connected nodes would become more important, although they are often uninformative as argued above. A different popularity-based ranking concept assumes that facts that have many witnesses in a corpus are highly informative (Kasneci et al., 2008). This, however, requires additional input apart from the graph, and the assumption that important facts are expressed more often than others may not hold in curated data stores like wikipedia or company networks.

Another approach based on the graph structure alone is to use properties of random walks on the graph. It has been argued that the commute time between nodes in an ER graph is a useful distance measure to find relevant neighboring nodes (Baluja et al., 2008). In each step, the random walk jumps from one node to any neighboring node. The commute time is the expected number of steps for a random walk starting from one node to reach another before returning to the first. Using this metric, the problems with the step distance are alleviated in that the commute distance decreases not only if there is a short path between two nodes, but also if there are several short paths between them. Thus, a single link over a high-degree hub is not likely to yield a small distance. Moreover, if two nodes are joined by a path containing a high-degree node, a random walk is likely to "get lost" at the high-degree node, taking steps into unrelated regions of the graph, increasing the commute time between the two nodes.

While these are strong intuitive arguments for the commute distance, it is very difficult to compute. There exists an analytic formula for the commute distance in terms of the pseudoinverse of the graph Laplacian (Klein and Randić, 1993). This, however, is computationally prohibitive for the large graphs encountered in the semantic web. The pseudoinverse of the sparse graph Laplacian matrix is in general not sparse, and a square matrix of size of the number of nodes in the graph can typically not be stored or worked with efficiently. More efficient approximations of the commute distance have been developed for citation graphs in Sarkar et al. (2008). Their method, however, still needs 4 seconds for a graph of 600k nodes, which is only a moderate size in the context of the semantic web. Moreover, it is not clear how it would perform on structurally more complex graphs such as DBpedia.

In order to combine the simplicity and speed of shortest path finding with the properties of the commute distance, we propose our novel approach. We also perform shortest path finding, but with a problem adapted graph metric that assigns each edge a weight dependent on the degrees of its endpoints. Finding shortest paths in our novel metric can then be interpreted in terms of maximizing the log-likelihood of the path between the two vertices in a random walk on the graph. It can be seen as an optimally adapted first order approximation to the commute distance, and thus experimentally inherits many of the favorable properties of the commute distance. At the same time the computations are very efficient since they reduce to purely local shortest path searches that can be performed with standard graph algorithms.

In the next section we describe exactly the approach we propose to solve the two proposed tasks. In Section 3 we explain its justification in terms of random walks, and how it can be seen as an approximation to the commute distance. In Section 4 we show a number of examples and a numeric evaluation on several semantic datasets, demonstrating the superior behavior both in comparison to the step distance path finding approach and to another simple approximation of the commute distance.

2 Proposed Approach

Let the semantic ER graph be represented as $G = (V, E)$ where V is the set of nodes or entities and E the set of edges or relations holding between the entities. We do not distinguish between different relation types for the edges, opting to treat them all equally. Moreover, we also symmetrize the graphs, since the semantic direction of a relation statement is often not syntactically obvious; for example, “buys” or “is bought by” might both appear in a graph.

For each edge $(u, v) \in E$, we then define a weight

$$w_{(u,v)} = \log(\deg(u)) + \log(\deg(v)),$$

where $\deg(u)$ is the degree of the node u. If G is connected, then the degree $\deg(u)$ of all nodes u is greater than or equal to one and thus $w_{(u,v)} \geq 0$ for all $(u, v) \in E$. The weights are therefore a valid positive semi-definite path metric on G, and the two described tasks can be solved using these novel edge weights in standard shortest paths routines.

In our first task, a node is specified as input, and we must retrieve a set of other nodes that are ranked based on how relevant and related they are to the query node. The results of this search might include, for example, topics that are contained within the query topic, topics which contain the query node, or topics that are related by common membership within a category or broader topic. In order to solve this task, we find the shortest path between the query node and all other nodes and rank the results. Note that using Dijkstra’s algorithm allows to directly retrieve the top ranked nodes without computing the shortest path to all other nodes.

In our second task, we are given two distinct nodes as input and wish to find paths between both that, ideally, provide unique insight into the relationship between the two nodes. This might include interesting or distinct ways that the two nodes are related. We solve this by finding the k shortest paths between the two nodes in the weighted graph, where k is a free parameter. We return the sequence of nodes in each of the k shortest paths.

The proposed metric can be justified intuitively: The distance to high-degree nodes which carry potentially very unspecific information, e.g. the “human” node, is large. This problem can be avoided as long as more specific, low degree nodes are within reach. This means that we are effectively searching in compactly connected, local subgraphs, assumed to carry context specific information.

As we will see in the experiments section, the proposed approach yields matches for queried nodes that are highly specific in subject matter and are very appropriate for someone who, for example, wants to explore a particular academic subject in detail. In addition, our metric facilitates the discovery of novel, distinct relations between nodes: vertices that are related to each other in some unique way (i.e. there is a path between them that is connected to relatively few other vertices outside of that path) are closer to each other than vertices that are linked by a very common relationship.

These intuitions can be further motivated by relating our approach with random walks on semantic graphs. Before we do so in the next section, note that the proposed approach only requires standard graph algorithms and is thus simple to implement. It also runs very efficiently even for large graphs. For example with Dijkstra's algorithm, we only have to visit k nodes to find the k closest neighbors.

3 The Connection to Random Walks

Consider a random walk on G. In each step the walk moves from one vertex v to an arbitrary adjacent node with probability $\deg(v)^{-1}$, where $\deg(v)$ is the degree of v. Denote the set of paths between two fixed vertices u and v by $\Pi_{u,v}$, i.e. $\pi = (\pi_1, \pi_2, .., \pi_{len(\pi)}) \in \Pi_{u,v}$ iff $\pi_1 = u$ and $\pi_{len(\pi)} = v$. The probability of the random walk following such a path and returning on the same route then is

$$\begin{aligned} p(\pi) &= \left(\prod_{i=1}^{n-1} \deg(\pi_i)^{-1}\right)\left(\prod_{i=2}^{n} \deg(\pi_i)^{-1}\right) \\ &= \deg(\pi_1)^{-1} \prod_{i=2}^{n-1} \deg(\pi_i)^{-2} \deg(\pi_n)^{-1}. \end{aligned}$$

The negative log-likelihood follows as

$$-\log p(\pi) = \log \deg(\pi_1) + 2\sum_{i=2}^{n-1} \log \deg(\pi_i) + \log \deg(\pi_n) = \sum_{i=1}^{n-1} w_{(\pi_i, \pi_{i+1})}.$$

This result shows that the negative log-likelihood of the path is exactly equal to the path length in our proposed metric. Moreover, shortest path finding between u and v using this metric is thus equivalent to finding that path in $\Pi_{u,v}$ that has minimal negative log-likelihood, or maximal probability, of a random walk following that path back and forth.

As argued in the introduction, random walks are very well suited to the proposed information retrieval task. At high-degree nodes, there are many possible nodes that could be visited next and the random walk might get lost in graph regions that often are relatively unrelated to the user's information needs. In contrast, on local, compactly connected subgraphs for a given topic, the random walk is highly likely to return quickly.

3.1 Approximation of the Commute Distance

Random walk probabilities also determine the commute time which has been proposed as an information metric on ER graphs before (Baluja et al., 2008). In contrast to our approach, the commute distance does not only measure whether there is a single high-probability connection between two nodes, but also takes into account how many such paths there are.

Since the commute distance uses more of the structure of the graph, it is potentially more robust. However, this comes at a huge computational cost and our approach is in comparison extremely efficient. Still, it can be seen as a first order approximation of the commute distance, as we will discuss now.

The commute time $C(u,v)$ between nodes u and v is

$$C(u,v) = \sum_{\pi:(\pi_1=u,\dots\pi_k=v,\dots\pi_{len(\pi)}=v)} len(\pi)\; p(\pi) = \sum_{\pi} len(\pi) \prod_{i=1}^{len(\pi)-1} \deg(\pi_i)^{-1}.$$

The sum goes over all paths that start and end at u and visit v in between. Since all terms are positive, a first order lower bound is to take into account only a single such path $\hat{\pi}$, i.e.

$$C(u,v) \geq len(\hat{\pi})\; p(\hat{\pi}).$$

Whether this is a tight bound depends on how concentrated the path probabilities are on a single term. While there are certainly situations where this is not the case, we would argue that for many semantic graphs the approximation might be acceptable. The reason is that the degree of the nodes enters multiplicatively into the sum. Consider query nodes that are both members of two categories of highly different sizes. Then the path through the smaller category and back on the same way is actually quadratically preferred over the one through the larger category.

Given above lower bound we now try to find the optimal lower bound for the commute distance $C(u,v)$. That is we search for that path $\hat{\pi}$ that contributes the most to the sum above. This then leads to

$$\max_{\hat{\pi}}\; len(\pi) p(\hat{\pi}) = \min_{\hat{\pi}}\; -\log len(\pi) + \sum_{i=1}^{n-1} \log \deg(\hat{\pi}_i).$$

The second term is additive in the length of the path and quickly dominates the first term whose magnitude increases sub-linearly. At the same time, for paths of equal length only the second term has to be considered for the minimization. Without too large an error we can thus neglect the first term in most cases. Moreover, we restrict the optimization set to those paths that go from u to v and return the same way. The result will still be a lower bound on the commute distance, and it allows us rewrite the problem using our proposed metric as

$$\min_{\hat{\pi}\in\Pi_{u,v}}\; 2\sum_{i=1}^{n-1} w_{(\hat{\pi}_i,\hat{\pi}_{i+1})}.$$

This is equivalent to our proposed approach up to a constant factor. We can thus interpret our approach as (approximately) finding an optimal lower bound to the commute distance, with the advantage that it can be computed very efficiently and with simple standard graph algorithms.

This derivation has involved a number of approximation steps that are not necessarily the tightest ones possible, see the review of Lovász (1993) for other approximations. Yet, this argument still gives some intuition why minimizing our proposed objective might be sensible and it allows to derive a computationally very advantageous algorithm.

4 Experiments

To demonstrate our methods we use two large, real world semantic ER graphs, namely DBpedia and OpenCyc.

For the DBpedia dataset, we combine the category (skos) and the article-category data files. From this we create an unweighted, undirected graph neglecting the different relationship types and directions. We ignore literals since they do not add information to the graph structure. Furthermore, we discard the “Concept” node to which each category is connected. Although this is exactly the type of high-degree node that disadvantages the naïve step distance against our method, we leave it out in the interest of a fair comparison. For the step distance one could assume this master node to be removed, but as we will show below, there are still many other high-level nodes that cause similar problems. Removing these is not as trivial and could lead to unwanted effects on the search results.

We similarly define the graph for the OpenCyc dataset. An overview of the properties of both graphs is given in Table 1.

	DBPedia	OpenCyc
Vertices	3,660,898	150,088
Edges	8,947,631	554,762
Average degree	4.88	7.39

Table 1. Basic statistics of the used datasets.

As baseline methods for our comparisons we use the following two approaches: First, we compare our method to using shortest paths with the step distance. Second, we compute a simple approximation of the commute distance.

The exact computation of the commute distance on the full graph is intractable, since it requires the graph Laplacian’s pseudoinverse, a matrix that for most graphs is too big to even be stored. Instead, we assume here that the commute distance is moderately local. For each query, we extract the 1000 closest nodes to the query node – in step distance – and only use the subgraph spanned by these nodes and the edges between them to compute the commute distance using the analytic formula of Klein and Randić (1993). If the subgraph has only few edges leaving it, the approximation is fairly reasonable. However, if a very unspecific node with many neighbors is among the closest nodes to the query vertex, then it will connect almost any node in the graph to the query by a path of, say, length 2. In this case the selection of the 1000 closest neighbor nodes is arbitrary and not much can be expected from our approximation of the commute distance. The baseline should thus not be regarded as a completely accurate representative of the true commute distance.

Step		Our approach		Approx. Commute	
Espresso	0	Espresso	0	Espresso	0
(C)Italian beverages	1	(C)Italian beverages	4.5	(C)Italian loanwords	1295.75
(C)Italian loanwords	1	(C)Coffee beverages	5.05	(C)Coffee beverages	1296.86
(C)Coffee beverages	1	(C)Italian loanwords	6.09	(C)Italian beverages	1297.35
(C)Italian cuisine	2	Bombardino	7.9	(C)Italian cuisine	1339.5
(C)Italian words and phrases	2	Caffè corretto	8.59	(C)Opera terminology	1401.79
(C)Italian language	2	Grappa	8.59	(C)Italian words and phrases	1452.94
(C)English words foreign origin	2	Torani	8.59	(C)Pasta	1467.75
(C)Romance loanwords	2	Lemonsoda	8.59	(C)Mediterranean cuisine	1529.31
(C)Beverages by region	2	Oransoda	8.59	(C)Cuisine by nationality	1544.99
(C)Italian alcoholic beverages	2	Pelmosoda	8.59	(C)Opera genres	1582.18
(C)Coffee preparation	2	Beverly (drink)	8.59	(C)Opera	1584.02
Castrato	2	Doppio	8.59	(C)Performing arts	1599.79
Da capo	2	Caffè	9	(C)Musical notation	1601.59
Graffiti	2	Chinotto	9	(C)European cuisine	1664.97
Glissando	2	Ammazzacaffè	9	(C)Italian language	1685.92
Macaroni	2	Stappj	9	Turkish coffee	1691.75
Mozzarella	2	Galvanina	9	(C)Beverages by region	1721.4
Opera	2	Irish coffee	9	(C)Dried meat	1737.1
Pasta	2	Cortado	9	(C)Musical theatre	1740.42
Pizza	2	Iced coffee	9	(C)Music	1743.96
Spaghetti	2	Pepsi Kona	9	(C)Articulations	1756.06
Tempo	2	Flat white	9	(C)English words foreign origin	1756.7
Cappuccino	2	Mochasippi	9	(C)Singing	1760.76
Legato	2	Red eye (drink)	9	(C)Salumi	1764.59
Staccato	2	Liqueur coffee	9	(C)Croatian cuisine	1769.7
Operetta	2	Lungo	9	(C)Entertainment	1773.37
Cadenza	2	Caffè Americano	9	(C)Theatrical genres	1788.05
Concerto	2	Espresso con panna	9	(C)Italian culture	1795.82
Cantata	2	Caffè breve	9	(C)Italian prod. protected origin	1799.13

Table 2. Top 30 results of neighborhood search for query node "Espresso" in DBpedia, along with the distances from the query node. First, Step means shortest path finding with the step distance; then follows our proposed approach; the last column shows the results of our simple approximation of the commute distance. Entities marked with (C) represent skos categories, other items are regular DBpedia resources.

4.1 First Task: Neighborhood finding

In the following we discuss a number of example results from the two datasets. In Table 2 we list the results of a search for the query node "Espresso." In this case, the step distance gets easily distracted by the high-degree neighbor "Italian loanwords." As a result, the majority of the results listed are unrelated Italian terms which refer mostly to music and food. The commute distance approximation returns highly irrelevant words that are also related mostly to food and music, but this is probably due to the nature of the approximation we use – most of the 1000 nodes nearest to the espresso node are probably also due to the Italian loanwords node. Our method, on the other hand, returns a list of about one third Italian sodas and non-coffee beverages and about two thirds drinks made with espresso or at least coffee, as well as a few other types of terms.

In Table 3 we performed another search for the term "iPod." The step distance mostly gives us various categories relating to hardware or software, and the commute distance mixes these results with a few more specific terms relating to the iPod's function and to the related iPhone. Our method, on the other hand, yields mostly articles relating specifically to variations and functions of the iPod and the iTunes software that is integral to the use of the iPod.

We also provide results for the OpenCyc dataset, which is of a slightly different nature. It contains many rather unspecific nodes like "temporally stuff like thing" which are nice examples of how such high-degree nodes are avoided by our algorithm. In Table 4 we show the results of a search for "machine learning." While the results of the other methods become wildly irrelevant after only the first few matches, nearly the first half of our results are still relevant to the topic at hand.

Step		Our approach		Approx. Commute	
IPod	0	IPod	0	IPod	0
(C)2001 introductions	1	(C)IPod	4.97	(C)ITunes	695.93
(C)IPod	1	(C)Industrial designs	5.78	(C)Portable media players	698.52
(C)Portable media players	1	(C)ITunes	5.98	(C)Digital audio players	750.23
(C)ITunes	1	(C)2001 introductions	6.29	(C)IPhone OS software	757.78
(C)IPhone OS software	1	(C)Portable media players	6.49	(C)IPod	784.31
(C)Industrial designs	1	(C)IPhone OS software	6.52	(C)Industrial designs	857.01
(C)2001	2	IPod click wheel	8.15	(C)Smartphones	889.69
(C)Apple Inc. software	2	IPod Photo	8.84	(C)2001 introductions	907.63
(C)Industrial design	2	List of iPod models	8.84	(C)Mac OS X software	929.97
(C)Windows software	2	Dock Connector	8.84	(C)Touchscreen portable media players	955.66
(C)Software by operating system	2	IPod Mini	9.25	(C)Consumer electronics brands	959.29
(C)Apple Inc. hardware	2	IPod advertising	9.25	(C)Apple Inc. software	973.22
(C)Windows media players	2	IPhone Touch	9.25	(C)IPhone	974.07
(C)Mac OS X software	2	IPod Nano	9.53	(C)2007 introductions	1010.71
(C)Digital audio players	2	Neistat Brothers	9.53	IPhone	1025.22
(C)USA PATRIOT Act	2	IPod Classic	9.53	(C)IPhone OS	1031.79
(C)MPEG	2	Ipod+HP	9.53	(C)Web 2.0	1035.86
(C)IPod accessories	2	List of iPhone OS devices	9.53	(C)Windows software	1047.63
(C)IPod software	2	IPod Shuffle	9.76	ITunes	1049.63
(C)21st-century introductions	2	Juicy Salif	9.77	(C)Apple Inc. hardware	1057.46
(C)ITunes-exclusive releases	2	DADVSI	10.09	(C)Software by operating system	1075.57
(C)IPhone OS games	2	NextWorth Solutions	10.09	(C)Online social networking	1096.2
(C)Mac OS X media players	2	IMix	10.17	(C)Mac OS software	1096.22
(C)Apple Inc. peripherals	2	Genius (iTunes)	10.17	(C)Personal digital assistants	1112.79
(C)Apple Inc. services	2	AirTunes	10.17	(C)Brands	1126.36
(C)Vehicles introduced in 2001	2	ITunes law	10.17	(C)Media players	1126.94
(C)IPhone	2	ITunes Music Store	10.17	(C)Creative Technology products	1129.28
(C)2001 comic debuts	2	ITunes U	10.17	(C)IPod software	1151.43
(C)IPhone OS	2	ITunes Applications	10.17	Nimbuzz	1156.45

Table 3. Top 30 results of neighborhood search for query node "iPod" in DBpedia. Labels as in Table 2.

To demonstrate the dramatic computational advantage of our method against even the described approximation of the commute distance, we picked 1000 query nodes at random and performed a query with it using all three methods. The mean run-times on a standard desktop PC as well as the standard deviation for each method is given below

Step	Our Approach	Approx. Commute
0.13s (0.07s)	0.11s (0.04s)	10.43s (9.51s)

The average run-time for our method was 0.11 seconds, compared to an average of 10.43 seconds for our approximation of the commute time – a difference of two orders of magnitude. As would be expected, our method runs approximately as fast as the step distance method.

It is worth noting that the time taken to approximate the commute time was extremely high in some cases. The longest time taken with the commute distance was over one minute, whereas the longest time taken with our method was only 0.66 seconds. Furthermore, one should also consider that the method we have used to calculate the commute distance is only an approximation using a graph of 1000 nodes. The most computationally intensive step required of the commute distance is the calculation of the pseudoinverse. Since this step requires cubic time to calculate, an attempt to improve the accuracy of the estimate by adding more nodes to the approximation would drastically increase the time required for computation, while an exact computation would be intractable for most practical problems.

4.2 Task 2: Path Finding

In this section, we present an example of our method as applied to the path finding task, showing the novel connections that our method is able to find

Step		Our approach		Approx. Commute	
machine learning	0	machine learning	0	machine learning	0
temporal stuff also a durative event	1	machine rule induction	2.48	first-order collection	875.61
computer activity	1	discriminative weight learning	2.89	temp stuff also a durative event	887.91
discriminative weight learning	1	generative weight learning	2.89	computer activity	897.63
generative weight learning	1	MLN Generated Using Learning Type	3.18	temporal stuff	921.03
machine rule induction	1	computer activity	6.27	employee computer activity type	1061.05
MLN Generated Using Learning Type	1	markov logic network	6.87	computer activity type	1090.34
alcoholism	2	temporal stuff also a durative event	7.75	athletic activity	1104.59
burning	2	MLN Data File Pathname	9.86	physical information transfer	1115.24
flowing	2	MLN File Pathname	9.86	biological transportation	1138.37
anthem	2	MLN Generated Using Cmd String	9.86	body movement	1152.19
the union of ensemble showman	2	MLN Rule File Pathname	9.86	recreational activity	1169.12
playing	2	MLN Type Const Dec File Pathname	9.86	using a computer	1181.74
halt	2	MLN Represented By Microtheory	10.27	information-accessing event	1195.75
rock climbing	2	Content Of MLN Fn	10.56	physical event	1196.47
snow-skiing	2	computer activity that computer did	11.85	structured information source	1213.39
Iter. Event Scene Fn id veg. 1-3 km	2	computer activity that person did	11.85	type of accomplishment	1236.74
rafting	2	hack	11.85	individual	1239.97
candy making	2	computer thread	11.85	computer editing	1256.46
composting	2	help desk session	11.85	internet activity	1266.56
woodworking	2	network packet filtering	11.85	running computer process	1280.32
diagnosis of Wegeners granulomatosis	2	network packet routing	11.85	locomotion event	1280.92
breast cancer treatment	2	opening presents	11.85	ride	1303.08
AIDS treatment	2	packet sniffing	11.85	CW instantiating	1313.32
acne care	2	partitioning a disk	11.85	unnatural thing	1315.36
affliction procedure	2	placing a residual malicious program	11.85	biological process	1321.43
allergic reaction treatment	2	browser requests a secure connection	12.13	QA clarifying collection type	1338.81
atrial septal aneurysm med treatment	2	locking computer display	12.13	internet communication	1355.33
most autistic procedure	2	website maintenance	12.13	network propagation	1357.71
vision impairment treatment	2	network prop. malicious program	12.13	candidate KB completeness node	1360.95

Table 4. Top 30 results of neighborhood search for query node "machine learning" in OpenCyc. Labels as in Table 2.

between two nodes in the graph – i.e., between two concepts in our semantic network. We also compare our method to path finding using the naïve step distance to show the advantage that our method has in discovering truly distinct and specific connections between concepts.

We compare the results of our path-finding method to those of the step distance path finding method in the case of finding connections between "computer vision" and "machine learning", again with data from DBpedia. The resulting paths are listed in Table 5. Many of the results of our method provide insight into exactly how machine learning is used to solve specific tasks in the computer vision domain. Although insightful, some of the paths returned here by our method have significant intersection with each other. This could, however, be remedied by, for example, modifying the k-shortest paths algorithm to add extra weight to the edges equivalent to the ones traversed in previously discovered paths. Such a modification would lead to increased diversity in the results.

The step distance, on the other hand, gives us only very vague, general connections between the two subjects. The most that we learn from these results is that computer vision and machine learning are both within the subject of artificial intelligence.

Note that our method is actually able to find informative paths of significant length. While for the step distance the exponential number of possibilities for such paths quickly renders the retrieval infeasible, our method is still able to discriminate between the many choices. This might be an important advantage when applying this framework to biomedical databases, such as for example Linked Life Data. Here, one often tries to find non-obvious rather long distance interactions between different genes and diseases to discover novel pathways. Focusing on the most discriminative ones might save significant research time in this domain.

Our Approach:

- Path 1 (length 15.2407): Computer vision - (C)Computer vision - (C)Learning in computer vision - Machine learning
- Path 2 (length 22.1722): Computer vision - (C)Computer vision - (C)Object recognition and categorization - Boosting methods for object categorization - (C)Learning in computer vision - Machine learning
- Path 3 (length 22.4706): Computer vision - (C)Artificial intelligence - (C)Cybernetics - Machine learning
- Path 4 (length 23.5585): Computer vision - (C)Computer vision - Segmentation based object categorization - (C)Object recognition and categorization - Boosting methods for object categorization - (C)Learning in computer vision - Machine learning
- Path 5 (length 23.5585): Computer vision - (C)Computer vision - Object recognition (computer vision) - (C)Object recognition and categorization - Boosting methods for object categorization - (C)Learning in computer vision - Machine learning

Step Distance:

- Path 1 (length 3): Computer vision - (C)Artificial intelligence - (C)Machine learning - Machine learning
- Path 2 (length 3): Computer vision - (C)Computer vision - (C)Learning in computer vision - Machine learning
- Path 3 (length 3): Computer vision - (C)Artificial intelligence - (C)Cybernetics - Machine learning
- Path 4 (length 4): Computer vision - (C)Artificial intelligence - (C)Machine learning - (C)Learning - Machine learning
- Path 5 (length 4): Computer vision - (C)Computer vision - (C)Artificial intelligence - (C)Machine learning - Machine learning

Table 5. Path finding between the terms “Computer vision” and “machine learning” in DBpedia.

5 Conclusion

We have presented a novel metric that allows us to solve two important information retrieval task in semantic networks efficiently. The metric just depends on the degrees of adjacent nodes, and shortest path search with such a metric will thus avoid unspecific, high-degree nodes. This allows us to find interesting neighbors of a query node and novel, specific links between entities, while only using standard graph algorithms.

Often the authors themselves discovered novel, interesting information when querying the test datasets DBpedia and OpenCyc with different entities. This makes us strongly believe that the proposed approach could also be helpful to others.

A detailed user study is currently under way. From a technical point one could imagine mixing the step metric and the proposed one to obtain a tunable trade-off between the length and the distinctiveness of a path. It would also be interesting to explore ways to learn additional parameters in the metric, e.g. assigning different weights to specific edge types. Such parametric learning approaches, however, would require a benchmarking dataset which is currently not available to us. In contrast, the proposed approach is parameter free and solely dependent on intuitive arguments.

Bibliography

Antezana, E., Kuiper, M., and Mironov, V. (2009). Biological knowledge management: the emerging role of the Semantic Web technologies. *Briefings in bioinformatics*.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2008). Dbpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735.

Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., and Aly, M. (2008). Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceeding of the 17th international conference on World Wide Web*, pages 895–904. ACM.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

Bundschus, M., Dejori, M., Stetter, M., Tresp, V., and Kriegel, H. (2008). Extraction of semantic biomedical relations from text using conditional random fields. *BMC bioinformatics*, 9(1):207.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Kasneci, G., Suchanek, F., Ifrim, G., Ramanath, M., and Weikum, G. (2008). Naga: Searching and ranking knowledge. In *Proc. of ICDE*, pages 1285–1288.

Klein, D. and Randić, M. (1993). Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95.

Lovász, L. (1993). Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46.

Momtchev, V., Peychev, D., Primov, T., and Georgiev, G. (2009). Expanding the pathway and interaction knowledge in linked life data. In *Proc. of International Semantic Web Challenge*.

Sarkar, P., Moore, A., and Prakash, A. (2008). Fast incremental proximity search in large graphs. In *Proceedings of the 25th international conference on Machine learning*, pages 896–903. ACM.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA. ACM Press.

Yen, J. (1971). Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.