

Similarity Search in Knowledge Graphs: Vector Space Model and Graph Embedding

Svetla Boytcheva^{1,2}, Atanas Kiryakov¹, and Pavlin Gyurov¹

¹ Ontotext, Sirma AI, Sofia, Bulgaria

{svetla.boytcheva, atanas.kiryakov, pavlin.gyurov}@ontotext.com

² Institute of Information and Communication Technologies,

Bulgarian Academy of Sciences, Bulgaria

svetla.boytcheva@gmail.com

Abstract. Exploring diverse knowledge graphs with SPARQL queries requires a laborious process of determining the appropriate predicates, classes and graph patterns. Another drawback is that such structured queries represent Boolean search without relevance ranking, which is impractical for flexible querying of big volumes of data. We present an experiment of adaptation of the Vector Space Model (VSM) document retrieval technique for knowledge graphs. As a demonstration we implemented SPARQL queries, which retrieve similar companies in FactForge - a graph of more than 2 billion statements, combining DBpedia, Geonames and other data. Initial evaluation shows that graph analytic technique PageRank can improve the F-Score of the VSM baseline from 66% to 73%. The use of industry classifications and news mentions bring together another 5% improvement. At the end we present early experiments on the usage of graph embedding techniques to solve the same problem: the best performing methods are HolE, TransD and ComplEx.

Keywords: Knowledge Graphs · Similarity · Graph Embedding .

1 Motivation

In a big data era, characterized by 5V³ (volume, velocity, variety, veracity, and value) knowledge management is a quite challenging task and requires the design and development of new smart and efficient solutions. One prominent new paradigm are the so-called Knowledge Graphs (KG), which put data in context via linking and semantic metadata and this way provide a framework for data integration, unification, analytics and sharing. Given a critical mass of domain knowledge and good level of connectivity, KGs can serve as context that helps computers comprehend and manipulate data.

Data in KG is accessed via structured query languages such as SPARQL⁴. Defining SPARQL queries involves determining the right classes, predicates, graph patterns and filters. This is not a problem for graphs which represent

³ <https://www.bbva.com/en/five-vs-big-data/>

⁴ <http://www.w3.org/TR/sparql11-overview/>

uniform information for single application. However the most interesting applications of KGs involve putting together data from multiple proprietary databases, encyclopedic knowledge (e.g. Wikipedia) and domain specific data and models (e.g. Geonames or SNOMED). Such graphs typically contain billions of facts, about millions of concepts and entities from thousands of classes, connected with thousands of different relationship types. Crafting useful SPARQL queries for such graphs can be a very laborious process. Another drawback of SPARQL, as a mechanism to access such graphs, is that structured queries represent Boolean search without relevance ranking, which is impractical for big volumes of diverse data. Often there are thousands of results that match the formal criteria, but what is really needed are the top 10 of those, ranked by relevance or importance (whatever the criteria).

There is a need for more advanced information retrieval models, resembling the web-scale full-text search techniques, which allow obtaining relevant information without the need of massive efforts of highly qualified librarian. Such techniques can help not only data exploration, but they can also be used for data management tasks such as reconciliation (linking and fusing information about one and the same object across different sources) and data cleaning (e.g. detecting duplicates).

For all those functionalities we need to be able to compare entities in the KG and to do judgment about their proximity. There are several proximity metrics, but we are looking for one that is human like and matches human expectations. Such proximity measure should not be dominated by the "popularity" of the entities, for instance, 9 of the top-10 most popular companies in the sector to be considered most similar to the 10th one. They should not be dominated by a single characteristic of the entity, e.g. revenue. There is not a necessity of any direct relations to exist between the entities, e.g. to judge Google similar to Alphabet because those are related with subsidiary relationship. We can already retrieve related objects using without using advanced information retrieval techniques.

Objective: To find similarity measure for nodes in a KG.

Hypothesis: Similar nodes share features and have similar ranks.

This paper extends the experiments with adaptation of VSM model which are reported in [6].

2 The Data

The experiments in this paper are performed on FactForge⁵ - a KG of Linked Open Data (LOD) and news articles about people, organizations and locations. It includes:

- DBpedia⁶ (the English version) - an RDF-ized version of Wikipedia ,
- GeoNames⁷ - exhaustive geographic information about populated places, countries and other features on Earth,

⁵ <http://factforge.net>

⁶ <http://wiki.dbpedia.org/>

⁷ <http://www.geonames.org/>

- The Financial Industry Business Ontology (FIBO),
- News metadata from NOW⁸ - a Semantic News Portal loaded with more than 1 million international news articles in English, continuously collected since year 2015. Articles are semantically annotated, [7], with references to DBPedia concepts and entities mentioned in the text – on average 50 references per article.

The entire graph contains more than 2 billion explicit statements loaded in Ontotext GraphDB – a semantic graph database engine, formerly known as OWLIM, [2]. GraphDB performs forward-chaining reasoning to infer another 300 million statements. Thus more than 2.5 billion facts are indexed in FactForge and available for public querying and exploration.

3 Adapting the Vector Space Model for Graphs

In FactForge data is represented in RDF, [8], where each edge in the graph represents a triple `<subject, predicate, object>`. The subject is the source edge, represented by a unique identifier of the resource (concept, entity, document, etc.) being described. The predicate represents the type of the relationship. The object is the target node, which could be either identifier of another resource or an XML literal.

To adapt VSM, [10], for searching for similar nodes in a KG, we will consider nodes as documents and the outgoing edges (PO-pairs) – as terms. Node $node_i$ will be represented by the following vector of the weighted PO-pairs in the graph, where n is the total number of all distinct PO-pairs in the graph.

$$node_i = \langle po_{i1}, po_{i2}, \dots, po_{in} \rangle \quad (1)$$

One can find description of the VSM and its adaptation to KG in [6].

The main concern is that there could be too many different PO-pairs in the KG. For instance, for FactForge it would be a vector space with more than 1 million dimensions. Thus, for reduction of the dimensions some restrictions for the search space are needed.

4 Experiments on node similarity crafted in SPARQL

We present a series of experiments of implementing the node similarity in FactForge based on the above VSM adaptation using SPARQL queries.

The experimental setup contains the following sub-tasks:

- SH1 and SH2: Shared PO-pairs identification and filtering by predicate
- SH3 and SH4: Similar nodes by count of shared edges
- SH5: Shared edges, ranked by popularity
- SH6: Similar nodes by weighted sum of shared edges and PageRank

⁸ <http://now.ontotext.com>

The first two experiments (queries SH1 to SH4) are presented in [6], where we got encouraging results when retrieving cities similar to a given one. The SPARQL queries needed to do that were quite simple also. However, when we tried the same rudimentary techniques for people and companies it becomes obvious that PO-pairs has to be weighted with regard to their "popularity" and information value.

4.1 Edge "popularity"

Obviously not only the number of shared PO-pairs plays important role in the similarity of two nodes in the KG. Like in the VSM for documents the term frequency is important, the "popularity" of PO-pair is also important. This experiment (Figure 4) aims to find the top 300 PO-pairs of the pairs describing `?node=Sofia`, ranked frequency of appearance in the graph. The results show (Figure 5) that these PO-pairs refer to very general features – like being located on Earth or a specific continent and etc. Like stop words in document retrieval, these PO-pairs do not contribute to the similarity measure and needs to be skipped from further consideration.

An improved version of SH5A (Figure 6) filters the most general PO-pairs, like predicate `gn:parentFeature` combined with Earth, the continents and United States. The results (Figure 7) show some improvement. Alongside with geographical locations there were also other features, e.g. `timezone +2` and relations to politics and history like `communism`, `Ancient_Rome`, `Austria-Hungary`, `Byzantine_Empire`, `Eastern_Orthodox_Church`, etc.

4.2 Node similarity as weighted sum of shared edges

Using the PO-pair popularity from the previous experiment we want to try node similarity, which goes beyond counting (CLM). We use the global PO-pair popularity as weight, analogous to using TF as weight for terms in document retrieval.

Let k be the number of all shared PO-pairs of a node n with some other nodes and $\langle po_1, po_2, \dots, po_k \rangle$ be their corresponding popularity values in the graph. We reduce the vector space V dimensionality to only those k PO-pairs. Then the nodes that share PO-pairs with the node n can be represented as vectors in this vectors space:

$$node_i = \langle w_{i1}, w_{i2}, \dots, w_{ik} \rangle \quad (2)$$

Where

$$w_{ij} = \begin{cases} \frac{1}{po_i}, & \text{if } node_i \text{ is linked to the } j^{th} \text{ PO-pair} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Let define the set of PO-pairs shared by the node n and $node_i$ in the vector space V as:

$$p_i = \{j \mid 1 \leq j \leq k \text{ and } node_i \text{ is linked to the } j^{th} \text{ PO-pair}\} \quad (4)$$

The similarity measure between the given node n and some other node $node_i$ is calculated by:

$$s(n, node_i) = \sum_{j=1}^k \sqrt{w_i w_{ij}} = \sum_{j \in p_i} \sqrt{\left(\frac{1}{po_i}\right)^2} = \sum_{j \in p_i} \frac{1}{po_i} \quad (5)$$

The example attempts to find the organizations most similar to `?node=dbr:Microsoft`. We use this switch from cities to organizations for two reasons. First, it turns out to be harder to derive similarity for companies, as compared to cities - the counting edges experiment does not bring useful results for companies. Second, finding similar companies has numerous business applications, e.g. identifying potential clients, suppliers or companies to invest in.

The query on Figure 8 in Appendix 1 implements this measure; its only bulky part is the sub-query that calculates the PO-pairs popularity like in SH5B. We filter the similar nodes we are searching for to be of type `dbo:Organisation` only. The results contain both the total number of shared PO-pairs (for information only) and the similarity score. The top N=20 ranked nodes in descending order by the similarity score are selected as a result (Figure 9). In this case similarity score is not normalized in the range $[0, 1]$, and greater values of $s(x, y)$ refer to more similar objects. We can observe that as more related to Microsoft are recommended companies like Microsoft Studios, Microsoft Mobile and Microsoft Store, that are highly related to it as subsidiary, branch, etc.

We are still missing in this list all top software companies like IBM, Alphabet, Facebook, Oracle, SAP, Symantec, VMware, Baidu, etc. which are similar in size and have similar offerings to Microsoft across their various product lines. The only result that is expected and shown up among the top 10 similar companies is Intel.

4.3 Feature filtering and logarithmic weights

The next modifications of this experiment SH6B (Figure 10) try to overcome these problems:

1. Additional filters were included to avoid some unintended relations and PO-pairs:
 - (a) Filtering of the related entities, particularly parents and subsidiaries, because we are looking for similar nodes, not for related ones.
 - (b) Filtering out PO-pairs that generalize another pairs, i.e. not adding to the score a pair if there is another relation between the nodes that is `rdfs:subPropertyOf` of the predicate of the current PO-pair.
 - (c) Filtering PO-pairs related to locations, in order to reduce weight of redundand PO-pairs: `ff-map:primaryCity`, `ff-map:primaryCountry`, `dbo:location`.

- Improved version of the similarity measure is used – PO-pair weight in SH6A was calculated as popularity (total count of occurrence) value. In SH6B the weight value is inspired by classical IDF, i.e. $-\log(P(j))$, where $P(j)$ is the probability PO-pair j to occur for the node n . According to the definition of the probability: $P(j) = \frac{\text{popularity of } j}{k}$, where k is the total number of all PO-pairs related to n . Because k is the same for all PO-pairs and it does not play role in the comparison for similarity measure and ranking, thus the value of k is omitted. In this version is used the following definition for weights

$$\text{node}_i = \langle w_{i1}, w_{i2}, \dots, w_{ik} \rangle \quad (6)$$

$$w_{ij} = \begin{cases} \frac{1}{\log(po_i + 1)}, & \text{if } \text{node}_i \text{ is linked to the } j^{\text{th}} \text{ PO-pair} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

Where a correction by +1 is made in the logarithm in order to avoid division by zero and log-transformation is used to reduce the variability of data.

$$s(n, \text{node}_i) = \sum_{j=1}^k \sqrt{(w_i w_{ij})} = \sum_{j \in p_i} \sqrt{\left(\frac{1}{\log(po_i + 1)}\right)^2} = \sum_{j \in p_i} \frac{1}{\log(po_i + 1)} \quad (8)$$

where p_i is defined as in equation (5).

In SPARQL query for SH6B (Figure 10) we initially selected top N=500 ranked PO-pairs linked to the node n , with the highest weight ($w_{ij} = \text{?pair_weight}$). Then we filter unintended relations applying filtering described in 1a), 1b), and 1c). Finally we select the top N=100 ranked similar nodes according to the similarity measure $s(n, \text{node}_i)$.

The results (Figure 11) show significant improvement and the majority of expected results are present. However, still some odd companies appear, like 21Vianet Group – the exclusive operator of Microsoft Azure and Office 365 services in China. Although this company has less shared PO-pairs with Microsoft than the other companies, the total weight is comparable with the top candidates. One of the main disadvantages of TF is that it favors rare PO-pairs.

In order to speed up the similarity evaluations, the next optimization of the query SH6C (Figure 12) counts on pre-calculation of the PO-pairs weights. It applies the same filtering like in SH6B, but it executes much faster as the global information about the PO-pair popularity is not calculated each time. To solve the problem with too rare PO-pairs, additional filtering is included, cutting those of them with weight below the threshold of 10%, i.e. $\text{?pair_weight} \leq 0.10$.

4.4 Difference in PageRanks as penalty

In SH6C we also extend the VSM proximity with a new consideration: the importance of the nodes in the graph, using PageRank, [3] as a measure of centrality and popularity. GrapnDB comes with RDFRank plug-in, which calculates the

PageRank-s of all nodes in a given RDF graph. Those can be accessed with system predicates, e.g. `rank:hasRDFRank3` gets the rank of the node with precision of three digits. The rank is used for dissimilarity measure, where distance between two nodes is measured as the difference between their ranks. For similarity is used the idea of TF-IDF.

$$d(n, node_i) = |rank(n) - rank(node_i)| \quad (9)$$

$$s(n, node_i) = \left(\sum_{j \in p_i} \frac{1}{\log(po_j + 1)} \right) (1 - d(n, node_i))^q \quad (10)$$

Some experiments are performed for values of $q = 1, 2, 3, 4$. Different values of q allow the dilution of the function values around the two critical points 0 and 1 for $s(x, y) = (1 - d(x, y))^q$.

The results of SH6C for $q = 4$ (Figure 13) show that the problem with outlier is solved for the case of Microsoft. The major problems with false positive examples are:

- Big number of shared specific features take dominance;
- Joint-stock company;
- Participation in competitions;
- Shared locations;
- Links to countries and cities.

5 Evaluation of the VSM adaptation

For evaluation of the proposed similarity measure we developed a small golden standard (Figure 1) for some famous companies, for which the human expectations and the proposed similarity rank can be assessed easily. The golden standard contains 102 pairs of companies.

The golden standard contains manually evaluated pairs of companies with respect of their appearance in top-10 similar companies by (True/False). The automatic marks are classified as: "correct match T_p " (This guess is correct); "missing match F_n " (There is a missing guess-reduces Recall); "incorrect match F_p " (Incorrect similarity guess); "expected miss T_n " (Not scored similar, as expected).

Evaluation metrics:

$$Precision = \frac{T_p}{T_p + F_p} \quad (11)$$

$$Recall = \frac{T_p}{T_p + F_n} \quad (12)$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (13)$$

$$Accuracy = \frac{T_p + T_n}{T_p + F_p + T_n + F_n} \quad (14)$$

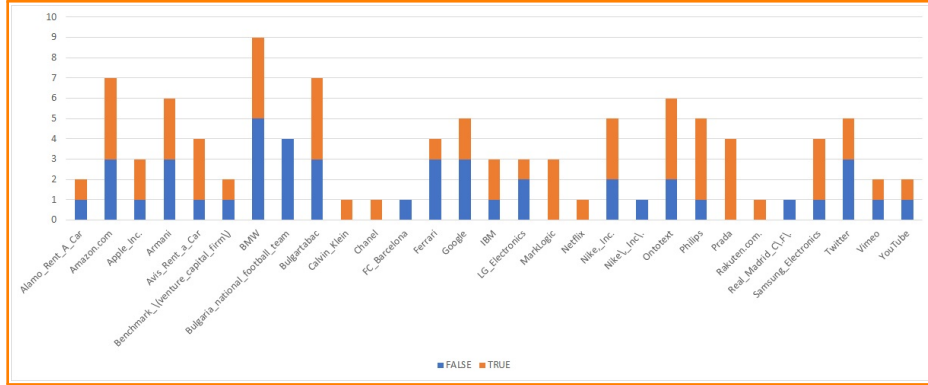


Fig. 1. Company similarity golden standard

$$Accuracy \ w/o \ F_n(AF) = \frac{T_p + T_n}{T_p + F_p + T_n} \quad (15)$$

	Baseline	$(1 - d)^2$	$(1 - d)^3$	$(1 - d)^4$	$(1 - d)^4 - Industry$	$(1 - d)^4 - Industry - Now$
Recall	85.2%	75.9%	79.6%	79.6%	83.3%	79.6%
Precision	54.1%	57.7%	63.2%	66.2%	71.4%	75.4%
F1	66.2%	65.6%	70.5%	72.3%	76.9%	77.5%
Accuracy	48.04%	51.96%	58.82%	61.76%	70.59%	72.55%
Accuracy w/o F_n	52.69%	60.92%	67.42%	70.79%	77.42%	81.32%

Table 1. Evaluation of similarity metrics for Company Graph golden standard

The evaluation results (Table 1) show that the highest Recall is obtained for the baseline (SH6A) and the highest Precision and F1-measure for modification of the SH6C experiment with adding information about Industries and company popularity and co-occurrence in the news by Now⁹. We experimented also with some other similarity measures, which also demonstrated good performance in terms of "Accuracy w/o F_n (AF)". This is the case of $s(x, y) = e^{-d(x, y)}$ with AF=81.82%. AF is an important evaluation, because when there are too many similar objects in the vector space, it is not possible for all of them to appear in the first top 5 and top 10 results, and many of them remain outside the evaluation subsets and are mistaken for F_n , only due to the limitations of the subset tested.

⁹ <https://now.ontotext.com/#channel>

6 Discussion on the VSM adaptation

The experiments with VSM model, conducted so far, show that:

- The selection of the number of features is important (set to 500). Smaller number does not allow for good differentiation of entities like IBM. Bigger number makes computation for Google and alike too complicated;
- Filtering features by maximum popularity (TF) is beneficial. Currently the threshold is set to 0.1 – lower values have negative impact on entities with small descriptions. For instance, without such filter, for Ontotext that is described with roughly 100 pairs, the pair `<industry,software>` expands the list of candidates too much;
- Using feature popularity (information value) for weighting is needed and beneficial. This confirms what we already know from the use of term frequency (TF) for weighting in VSM for document retrieval. As a local information is used the count of distinct shared pairs and for global information is used sum of reciprocal popularity values for some normalization of the popularity values in the range $[0, 1]$, a kind of inverse document frequency.
- Using node importance (an adapted version of PageRank) helps. Note, that we do not use this rank for weighting, we penalize big differences in the importance between two nodes.

The setup used to experiment with VSM has its limitations. The main bottleneck is the golden standard. It is non clear how the decision that some company should be in top 10 most similar companies was made. There are many cases when the company is ranked as 11th or 12th which does not make similarity measure less accurate. The GS probably will be better to contain not only positive and negative example but to allow fuzzy membership to a certain extend.

A principal limitation in the VSM experiments presented above is that PO-pairs are consider as independent features. Ideally, we should be able to take into account similarities between predicates (relation types) and objects (target nodes). For instance, the features `<hasOfficeIn,SofiaOblast>` and `<locatedIn,Sofia>` are not 100% unrelated. In the following section we report initial results from experiments, which addresses this problem.

7 Knowledge Graph Embedding

Our second series of experiments towards nodes similarity makes the following extensions:

- We introduce 4 grades of similarity: Very similar, Plenty of important similarities, Mostly different with some similarity, Dissimilar. It practice human experts find it hard to judge only in two grades (similar-dissimilar);
- A bigger and more systematically developed golden standard;
- Using graph embedding techniques, instead of vanilla VSM.

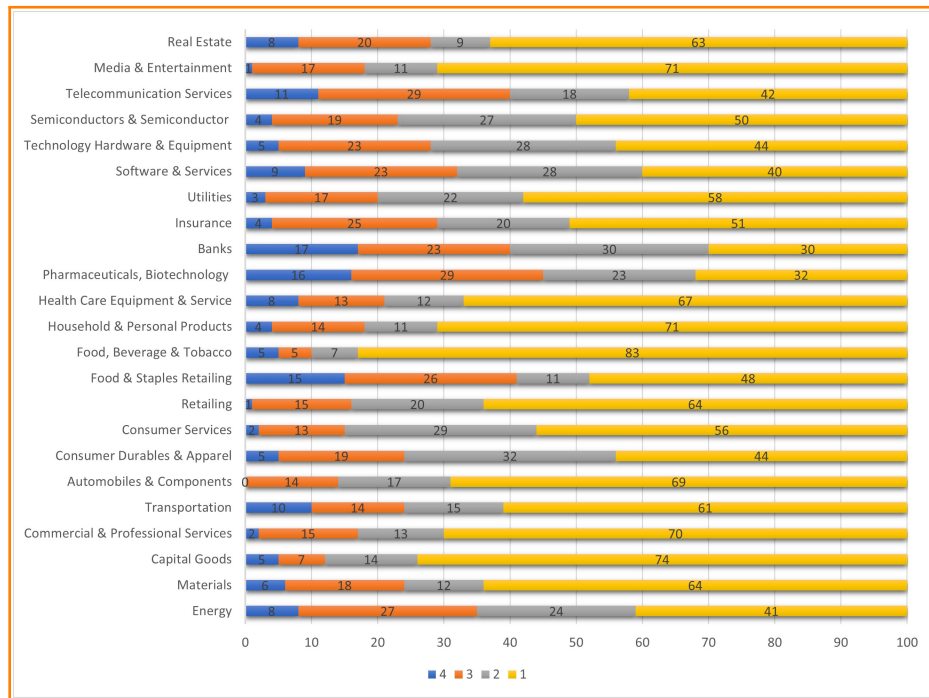


Fig. 2. Ratio of pairs of companies annotated in the four categories: 1- "Dissimilar", 2- "Mostly different with some similarity", 3 - "Plenty of important similarities", 4 - "Very similar"

7.1 Golden Standard for Similarity of Companies in DBpedia

Taking into account the problem with the size and representativeness of the golden standard (GS), we developed a new one that covers all 24 industrial sectors at level two of GICS¹⁰. Some 2300 pairs of companies for all sectors (approx. 100 pairs per sector) were annotated manually with four grades of similarity (see Fig. 1). The average inter-annotator agreement was about 71%, which shows that even for a person the assessment of similarity between two companies is a very difficult and ambiguous task, despite the clear rules and criteria in the methodology for building the GS. We selected 4 companies from each sector from DBpedia and asked experts to judge their similarity with top 25 similarity candidates generated for each of them by SH6C SPARQL query from the VSM experiments. In total 1898 different companies were included in the GS. Although a large GS was created, the main problem of KG scarcity is still not overcome, as for many of the features there is no value, there is a huge variability of URIs that are associated with the same value and not

¹⁰ https://www.spglobal.com/marketintelligence/en/documents/112727-gics-mapbook_2018_v3_letter_digitalspreads.pdf

mapped in the KG. For example, for locations, products, etc. The sparseness of KG is one of the main reasons for low accuracy in the search for similarity, as important relationships between objects are omitted. DBPedia borrows both the advantages and the disadvantages of Wikipedia - it has great coverage and high level of connectivity, but descriptions are sometimes quite irregular.

The evaluation of the proposed vector spaces models over the new GS show that the accuracy vary for different sectors from lowest about 10% (for "Food, Beverage & Tobacco Food, Beverage & Tobacco", "Capital Goods", "Automobiles & Components") up to about 45% (for "Pharmaceuticals, Biotechnology", "Banks", "Telecommunication Services", "Food & Staples Retailing"), and in average 24.95% for all sectors.

7.2 KG Embedding Techniques

The results achieved with a naive implementation of VSM motivated us to search for other methods that can be applied for the similarity task. Currently the state of the art approaches are based on the so called graph embedding [4]. They rely on an initial representation in low-dimensional VSM of the entities and relations, as well as evaluation of the plausibility of each edge in the graph. Then vectors for each node and edge are adjusted in the course of several iterations of updating the embedding vectors by using different optimization techniques in order to maximize the global plausibility of the facts expressed as edges in the graph. The main disadvantage of such KG embedding models is that they can be trained only on the observed facts, i.e. under closed world assumption. The main advantage is that different additional properties of the entities are taken in consideration, e.g. type of the entities, textual description, complex paths and relations. The essence of this methods is that they derive "latent semantics" by means of analyzing co-occurrence. The embedding vectors for edges `<locatedIn,Sofia>` and `<hasOfficeIn,SofiaOblast>` are expected to have high proximity and on their turn to contribute to the proximity of two nodes, e.g. companies, characterized by one of them and by the other.

The training dataset of RDF triples was generated on the top of the GS (pairs of company identifiers with similarity judgement) enriched with additional information about the companies like: name, text description, location, industry, main products, number of employees, revenue, etc. The resulting data set was augmented with additionally generated similarity/dissimilarity triples and other reflexive relations so that a high degree of connectivity was achieved with total 355711 RDF triples, 4695 entities and 21 relation types. Note that the graph developed this way is much smaller and poorer compared to the original knowledge graph used for the VSM experiments so far - the FactForge graph with more than 2 billion statements.

7.3 Evaluation of graph embedding techniques

Experiments with this training data set was performed with several state-of-the-art KG embedding methods. The best results were achieved for Semantic Vectors

[13] (Predication-Based Semantic Indexing, integrated in GraphDB as Semantic Search plug-in), translation-based model TransD [5], tensor factorization-based models HolE [9] [14] and ComplEx [11] [12]. So further experiments were limited to them using the OpenKE toolkit¹¹.

The head prediction scores were calculated for estimating similarity among the companies of the GS. The Top 10 and Top 5 most similar companies were calculated for each algorithm and was performed evaluation based no the GS (see Table 3 and Table 2). The KG embedding models were trained with the default hyper parameters. We also used the Graphlet approach (described below) – the rows for these experiments are marked with "Graph".

Method	T_p	F_n	F_p	T_n	precision	recall	F1	Accuracy	AF
<i>ComplEx</i>	135	407	55	1576	71.05%	24.91%	36.89%	78.7%	96.9%
<i>ComplEx Graph</i>	65	477	31	1599	67.71%	11.99%	20.38%	76.6%	98.2%
<i>HolE</i>	104	437	85	1546	55.03%	19.22%	28.49%	76.0%	95.1%
<i>HolE Graph</i>	150	391	134	1496	52.82%	27.73%	36.36%	75.8%	92.5%
<i>TransD</i>	270	272	579	1052	31.80%	49.82%	38.82%	60.8%	69.5%
<i>Semantic Vectors</i>	58	484	80	1550	42.03%	10.70%	17.09%	74.0%	95.3%

Table 2. Evaluation for Top 5 results of the KG embedding models

Unsurprisingly, Top 5 comparisons have lower accuracy and F1 metrics than top 10. Some examples for searching Top 10 similar companies to Prada by the trained KG embedding model of HolE is shown on Figure 3.

Method	T_p	F_n	F_p	T_n	precision	recall	F1	Accuracy	AF
<i>ComplEx</i>	169	372	123	1508	57.88%	31.24%	40.58%	77.2%	93.2%
<i>ComplEx Graph</i>	85	456	55	1576	60.71%	15.71%	24.96%	76.5%	96.8%
<i>HolE</i>	160	382	166	1465	49.08%	29.52%	36.87%	74.8%	90.7%
<i>HolE Graph</i>	260	281	257	1373	50.29%	48.06%	49.15%	75.2%	86.4%
<i>TransD</i>	346	196	766	864	31.12%	63.84%	41.84%	55.7%	61.2%
<i>Semantic Vectors</i>	91	451	145	1485	38.56%	16.79%	23.39%	72.6%	91.6%

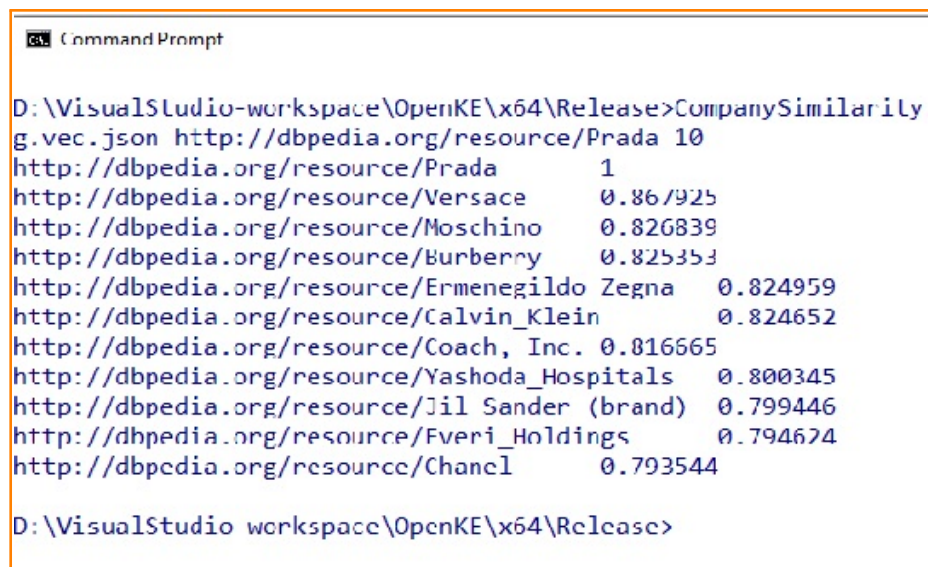
Table 3. Evaluation for Top 10 results of the KG embedding models

When the search space is too large one can apply recursive factorization of the complex graph into sub-graph patterns of size k nodes (called *Graphlets*)[1]. Graphlets contain statistically significant information about the KG embedding and provide fast, scalable and effective solution for large vector spaces.

¹¹ <http://139.129.163.161/index/toolkits#openke>

In our Graphlet approach, using the existing trained KG-embedding model for evaluating the similarity of the edges, the following heuristics were made: If two companies have similar properties, they are likely to be similar.

The experiments and detailed result evaluation (not only the numbers, but also the reasons behind them) showed much more relevant similarity scoring compared to the original head prediction (see Table 3 and Table 2).



```
Command Prompt
D:\VisualStudio-workspace\OpenKE\x64\Release>CompanySimilarity.g.vec.json http://dbpedia.org/resource/Prada 10
http://dbpedia.org/resource/Prada 1
http://dbpedia.org/resource/Versace 0.867925
http://dbpedia.org/resource/Moschino 0.826839
http://dbpedia.org/resource/Burberry 0.825353
http://dbpedia.org/resource/Ermenegildo Zegna 0.824959
http://dbpedia.org/resource/Calvin_Klein 0.824652
http://dbpedia.org/resource/Coach, Inc. 0.816665
http://dbpedia.org/resource/Yashoda_Hospitals 0.800345
http://dbpedia.org/resource/Jil Sander (brand) 0.799446
http://dbpedia.org/resource/Everi_Holdings 0.794674
http://dbpedia.org/resource/Chanel 0.793544
D:\VisualStudio workspace\OpenKE\x64\Release>
```

Fig. 3. Prada Graphlet results for top 10 with HoE

Moreover, the Graphlet approach also addresses the following drawback of all the KG embedding algorithms - they work on observed facts in the training dataset. Thus companies outside of the training dataset can not be evaluated. However, if a "new" company is "defined" using the properties, literals, etc. from the existing KG embedding model, the Graphlet approach provides functionality to calculate the similarity of such "new" company to the companies already available in the KG embedding model.

There is a lot of further work for fine tuning of this approach, e.g. different properties could have different weight when the similarity is calculated, when some numeric literals are used, it might be more accurate if they are grouped in intervals and these intervals to be used as property values and so on.

8 Conclusion and Further Work

We demonstrated that IR techniques can be adapted to perform search for similar nodes in a KG. We implemented the VSM, treating each node in the graph

as document and its outgoing edges (predicate-object pairs) as terms, which describe it. The PO-pairs were weighed to reflect that the more popular pairs bring less information value. We have proven that graph analytics (namely PageRank) can improve the performance of the baseline VSM.

This model has been implemented via SPARQL queries against a graph of more than 2 billion statements, central to which is DBPedia. It should be recognized that DBPedia is very special graph, based on the special properties of Wikipedia: a very large set of highly interlinked descriptions of the most the popular entities and concepts. What is most important, it is developed under well determined editorial guidelines and processes. We should admit that such IR techniques may not be able to bring decent results on graphs which are less interconnected and concise.

The VSM experiment presented in the main body of the article served as proof of concept that such techniques can be used for node similarity. While it delivers useful results, when tuned for a specific case (similarity of companies, which are popular enough to the appear in Wikipedia), we were quite aware of its principled limitation - it deals with discrete features, i.e. `<locatedIn,Manhattan>` and `<headquarteredIn,NewYorkCity>` are treated as completely different features. To address this we extended the experiment developing a bigger corpus and employing few of the most popular graph embedding techniques, which bring some flavour of latent semantics. Some additional improvement of the KG are needed to overcome the problem with KG sparsity, like data cleaning, normalization to standard classifications and thesauri, entity matching, literals transformation, etc. The conclusion from the experiments so far is that the best performing method is HolE (F-Score 49%), followed by TransD and ComplEx. Those easily outperform that VSM implementation (F-Score 23%).

References

1. Ahmed, N.K., Neville, J., Rossi, R.A., Duffield, N.G., Willke, T.L.: Graphlet decomposition: Framework, algorithms, and applications. *Knowledge and Information Systems* **50**(3), 689–722 (2017)
2. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: Owlrim: A family of scalable semantic repositories. *Semantic Web* **2**(1), 33–42 (2011)
3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Computer Networks* **30**(1), 107–117 (1998)
4. Dai, Y., Wang, S., Xiong, N.N., Guo, W.: A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics* **9**(5), 750 (2020)
5. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. pp. 687–696 (2015)
6. Kiryakov, A., Boytcheva, S.: Similarity search in knowledge graphs: Adapting the vector space model. In: *Knowledge, Language Models*, (eds. Milena Slavcheva, Kiril Simov, Petya Osenova, Svetla Boytcheva). Incoma (2020)
7. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. *Journal of Web Semantics* **2**(1), 49–79 (2004)

8. Klyne, G., Carroll, J.J., McBride, B.: Resource description framework (rdf): concepts and abstract syntax, 2004. February. URL: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210> (2009)
9. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. arXiv preprint arXiv:1510.04935 (2015)
10. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. *Communications of the ACM* **18**(11), 613–620 (1975)
11. Scharffe, F., Liu, Y., Zhou, C.: Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In: *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR)*, Pasadena (CA US) (2009)
12. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. *International Conference on Machine Learning (ICML)* (2016)
13. Widdows, D., Cohen, T.: Reasoning with vectors: A continuous model for fast robust inference. *Logic Journal of the IGPL* **23**(2), 141–173 (2015)
14. Xue, Y., Yuan, Y., Xu, Z., Sabharwal, A.: Expanding holographic embeddings for knowledge completion. In: *Advances in Neural Information Processing Systems*. pp. 4491–4501 (2018)

A Appendix 1: SPARQL Queries

Here we provide queries that implement VSM-like node similarity in FactForge along with snapshots from query results. FactForge is a graph of over 2 billions statements (see section 2) loaded in GraphDB and publicly available for exploration at <http://factfroge.net> These queries are also available in FactForge as Saved queries. We start with list of prefixes used followed by queries and results.

```

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX gn: <http://www.geonames.org/ontology#>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX ff-map: <http://factforge.net/ff2016-mapping/>
PREFIX wd: <http://www.wikidata.org/entity/>

```

```

10 SELECT ?p ?o (COUNT(DISTINCT ?different_node) AS ?pair_popularity) {
11     BIND( dbr:Sofia AS ?node)
12     ?node ?p ?o .
13     FILTER(?p NOT IN ( rdf:type, owl:sameAs, dul:sameSettingAs,
14                       dbo:wikiPageWikiLinkText, dbp:wikiPageUsesTemplate))
15     ?different_node ?p ?o . FILTER(?node != ?different_node)
16 } GROUP BY ?node ?p ?o ORDER BY DESC(?pair_popularity) LIMIT 300

```

Fig. 4. Query SH5A: Shared edges by popularity

	p	o	pair_popularity
1	gn:parentFeature	dbr:Earth	"11315596"^^xsd:integer
2	gn:featureClass	gn:P	"4447649"^^xsd:integer
3	gn:parentFeature	dbr:Europe	"2448280"^^xsd:integer
4	gn:parentFeature	dbr:Eastern_Europe	"586226"^^xsd:integer
5	dbo:wikiPageWikiLink	dbr:List_of_sovereign_states	"222049"^^xsd:integer
6	dbp:subdivisionType	dbr:List_of_sovereign_states	"221549"^^xsd:integer
7	dbo:wikiPageWikiLink	dbr:Poland	"73843"^^xsd:integer
8	dbo:wikiPageWikiLink	dbr:Russia	"70214"^^xsd:integer
9	dbo:wikiPageWikiLink	dbr:Catholic_Church	"64478"^^xsd:integer
10	dbo:wikiPageWikiLink	dbr:Paris	"54645"^^xsd:integer

Fig. 5. Results for SH5A: Shared edges by popularity


```

15 SELECT ?p ?o (COUNT(DISTINCT ?different_node) AS ?pair_popularity) {
16   BIND( <http://dbpedia.org/resource/Sofia> AS ?node)
17   ?node ?p ?o .
18   FILTER(?p NOT IN ( rdf:type, owl:sameAs, dul:sameSettingAs,
19                     dbo:wikiPageWikiLinkText, dbp:wikiPageUsesTemplate))
20   FILTER(?p != gn:parentFeature || ?o NOT IN (dbr:Earth,
21                                               dbr:Europe, dbr:Asia, dbr:Americas, dbr:North_America,
22                                               dbr:South_America, dbr:Africa, dbr:United_States ))
23   FILTER(?p != gn:featureClass || ?o != gn:P)
24   FILTER(?p != gn:countryCode || ?o NOT IN ("US"))
25   FILTER(?p NOT IN (gn:parentCountry, ff-map:location,
26                   ff-map:locationCountry, dul:hasLocation,
27                   dbo:country,wd:P17 ) || ?o != dbr:United_States)
28   FILTER(?p NOT IN (dbo:wikiPageWikiLink, dct:subject) ||
29             ?o != dbc:Living_people)
30
31   ?different_node ?p ?o . FILTER(?node != ?different_node)
32 } GROUP BY ?node ?p ?o ORDER BY DESC(?pair_popularity) LIMIT 200

```

Fig. 6. Query SH5B: Shared edges by popularity, filtered most popular

	p	o	pair_popularity
1	gn:parentFeature	dbr:Eastern_Europe	"586226"^^xsd:integer
2	dbo:wikiPageWikiLink	dbr:List_of_sovereign_states	"222049"^^xsd:integer
3	dbp:subdivisionType	dbr:List_of_sovereign_states	"221549"^^xsd:integer
4	dbo:wikiPageWikiLink	dbr:Poland	"73843"^^xsd:integer
5	dbo:wikiPageWikiLink	dbr:Russia	"70214"^^xsd:integer
6	dbo:wikiPageWikiLink	dbr:Catholic_Church	"64478"^^xsd:integer
7	dbo:wikiPageWikiLink	dbr:Paris	"54645"^^xsd:integer
8	dbo:utcOffset	"+2"	"51181"^^xsd:integer
9	dbo:wikiPageWikiLink	dbr:Soviet_Union	"50885"^^xsd:integer
10	dbo:wikiPageWikiLink	dbr:Europe	"41893"^^xsd:integer

Fig. 7. Results for SH5B: Shared edges by popularity, filtered most popular

```

12 SELECT ?similar_node
13     (COUNT(DISTINCT ?pair) AS ?shared_pairs)
14     (SUM(1/?pair_popularity) AS ?similarity_score)
15 {
16     { SELECT ?node ?p ?o
17         (COUNT(DISTINCT ?different_node) AS ?pair_popularity)
18         {
19             BIND( dbr:Microsoft AS ?node)
20             ?node ?p ?o .
21             FILTER(?p NOT IN ( rdf:type, owl:sameAs, dul:sameSettingAs,
22 dbr:wikiPageWikiLinkText, dbp:wikiPageUsesTemplate))
23             FILTER(?p != gn:parentFeature || ?o NOT IN (dbr:Earth, dbr:Europe, dbr:Asia,
24 dbr:Americas, dbr:North_America,
25 dbr:South_America, dbr:Africa, dbr:United_States ))
26             FILTER(?p != gn:featureClass || ?o != gn:P)
27             FILTER(?p != gn:countryCode || ?o NOT IN ("US"))
28             FILTER(?p NOT IN (gn:parentCountry, ff-map:location, wd:P17, ff-
29 map:locationCountry, dul:hasLocation, dbo:country )
30             || ?o != dbr:United_States)
31             FILTER(?p NOT IN (dbo:wikiPageWikiLink, dct:subject) || ?o != dbc:Living_people)
32             ?different_node ?p ?o . FILTER(?node != ?different_node)
33         } GROUP BY ?node ?p ?o ORDER BY ?pair_popularity }
34     ?similar_node ?p ?o . FILTER(?similar_node != ?node)
35     ?similar_node a dbo:Organisation .
36     BIND(CONCAT(STR(?p), STR(?o)) AS ?pair)
37 } GROUP BY ?similar_node ORDER BY DESC(?similarity_score) LIMIT 20

```

Fig. 8. Query SH6A: Similar nodes by weighted sum of shared edges

	similar_node	shared_pairs	similarity_score
1	dbr:Microsoft_Studios	"64"^^xsd:integer	"34.810326149127563932672567"^^xsd:decimal
2	dbr:21Vianet	"18"^^xsd:integer	"6.045021299286407389912937"^^xsd:decimal
3	dbr:Secure_Islands	"29"^^xsd:integer	"4.135484616325125500188939"^^xsd:decimal
4	dbr:Microsoft_Store	"49"^^xsd:integer	"4.106176575217512882112406"^^xsd:decimal
5	dbr:SinaSoft_Corporation	"20"^^xsd:integer	"4.029139265835503943236249"^^xsd:decimal
6	dbr:intel	"75"^^xsd:integer	"3.260266724702320291104799"^^xsd:decimal
7	dbr:Microsoft_Mobile	"39"^^xsd:integer	"3.21931437018850762262320"^^xsd:decimal
8	dbr:Godrej_Infotech	"17"^^xsd:integer	"3.022268858493237585237043"^^xsd:decimal
9	dbr:Microsoft_Digital_Crimes_Unit	"8"^^xsd:integer	"2.240951390837709100601130"^^xsd:decimal
10	dbr:Quest_Software	"21"^^xsd:integer	"2.036929147030291759331856"^^xsd:decimal

Fig. 9. Results for SH6A: Similar nodes by weighted sum of shared edges

```

17 SELECT ?similar_node (COUNT(DISTINCT ?pair) AS ?shared_pairs) (SUM(?
pair_weight) AS ?similarity_score)
18 {
19   { SELECT ?node ?p ?o
20     (1/(f:log(COUNT(DISTINCT ?different_node))+1) AS ?pair_weight)
21     {↔} GROUP BY ?node ?p ?o ORDER BY DESC(?pair_weight) LIMIT 500
22   } # as query SH5B, but 1/f:log+1 used instead of 1/?pair_weight
35
36   ?similar_node ?p ?o .
37   FILTER(?similar_node != ?node && ?similar_node != ?o)
38   ?similar_node a dbo:Company .
39   FILTER NOT EXISTS {?node dbo:subsidiary|dbo:parent ?similar_node.}
40   FILTER NOT EXISTS {?similar_node ?p2 ?o . FILTER(?p2 != ?p)
41     ?p2 rdfs:subPropertyOf ?p.
42   }
43   FILTER NOT EXISTS { FILTER(?p = dbo:location)
44     ?similar_node ff-map:primaryCountry | ff-map:primaryCity ?o .
45     ?node ff-map:primaryCountry | ff-map:primaryCity ?o .
46   }
47   BIND(CONCAT(STR(?p), STR(?o)) AS ?pair)
48 } GROUP BY ?similar_node ORDER BY DESC(?similarity_score) LIMIT 100

```

Fig. 10. Query SH6B: Similar nodes by weighted sum of shared edges, log-weight

	similar_node	shared_pairs	similarity_score
1	dbr:Apple_Inc.	"64" ^{xsd:integer}	"12.401238552081056" ^{xsd:double}
2	dbr:Intel	"42" ^{xsd:integer}	"8.741598937775576" ^{xsd:double}
3	dbr:Facebook	"42" ^{xsd:integer}	"8.219393119756319" ^{xsd:double}
4	dbr:21Vianet	"14" ^{xsd:integer}	"7.715773748173311" ^{xsd:double}
5	dbr:Google	"34" ^{xsd:integer}	"6.55251742214344" ^{xsd:double}
6	dbr:Dell	"34" ^{xsd:integer}	"6.0616087794240245" ^{xsd:double}
7	dbr:Oracle_Corporation	"30" ^{xsd:integer}	"5.815410265869873" ^{xsd:double}
8	dbr:IBM	"31" ^{xsd:integer}	"5.6667386836466465" ^{xsd:double}
9	dbr:AOL	"26" ^{xsd:integer}	"5.095279439402964" ^{xsd:double}
10	dbr:Hewlett-Packard	"25" ^{xsd:integer}	"4.7100522334192565" ^{xsd:double}

Fig. 11. Results for SH6B: Similar nodes by weighted sum of shared edges, log-weight

```

13 SELECT ?similar_node
14 (SUM(?pair_weight) AS ?similarity_score1)
15 (?similarity_score1 * ((1-abs(?node_rank-?sim_node_rank))*(1-abs(?node_rank-?
sim_node_rank))*(1-abs(?node_rank-?sim_node_rank))*(1-abs(?node_rank-?sim_node_rank)))
AS ?similarity_score)
16 from onto:disable-sameAs
17 { { SELECT ?node ?node_rank ?p ?o ?pair_weight
18 { BIND( dbr:Microsoft | AS ?node)
19 ?node rank:hasRDFRank3 ?node_rank.
20 ?node ?p ?o . FILTER(?o != ?node)
21 [] ff-map:po-weight-predicate ?p ; ff-map:po-weight-object ?o ; ff-map:po-
weight ?pair_weight.
22 FILTER(?pair_weight > 0.10)
23 } ORDER BY DESC(?pair_weight) LIMIT 500 }
24 ?similar_node ?p ?o . FILTER(?similar_node != ?node && ?similar_node != ?o)
25 ?similar_node a dbo:Company .
26 FILTER NOT EXISTS { ?node dbo:subsidiary | dbo:parent ?similar_node . }
27 FILTER NOT EXISTS { ?similar_node ?p2 ?o . FILTER(?p2 != ?p) ?p2
rdfs:subPropertyOf ?p. }
28 FILTER NOT EXISTS { FILTER(?p = dbo:location)
29 ?similar_node ff-map:primaryCountry | ff-map:primaryCity ?o .
30 ?node ff-map:primaryCountry | ff-map:primaryCity ?o . }
31 ?similar_node rank:hasRDFRank3 ?sim_node_rank.
32 } GROUP BY ?similar_node ?node_rank ?sim_node_rank ORDER BY DESC(?similarity_score)
LIMIT 20

```

Fig. 12. SPARQL query SH6C: Similar nodes by weighted sum of shared edges, log-weight, optimized

	similar_node	similarity_score1	similarity_score
1	dbr:Apple_Inc.	"12.711680937478992"^^xsd:double	"12.409335941960402"^^xsd:double
2	dbr:Facebook	"8.963025370985656"^^xsd:double	"8.855951799665537"^^xsd:double
3	dbr:Intel	"9.196486963959867"^^xsd:double	"8.44797590586151"^^xsd:double
4	dbr:Google	"7.392273411204255"^^xsd:double	"7.129722644801692"^^xsd:double
5	dbr:Dell	"6.718268970574828"^^xsd:double	"6.046348210373025"^^xsd:double
6	dbr:IBM	"6.332369212276205"^^xsd:double	"5.960891105439175"^^xsd:double
7	dbr:Oracle_Corporation	"6.39550630318854"^^xsd:double	"5.779542005596222"^^xsd:double
8	dbr:AOL	"5.51100957112886"^^xsd:double	"5.103958249238079"^^xsd:double
9	dbr:Hewlett-Packard	"5.410628763098229"^^xsd:double	"4.909609119631528"^^xsd:double
10	dbr:Amazon.com	"4.806223971605846"^^xsd:double	"4.254915898696704"^^xsd:double

Fig. 13. The top 10 results for SH6C with similarity for value q=4