Similarity Search in Knowledge Graphs: Adapting the Vector Space Model

Atanas Kiryakov[1] and Svetla Boytcheva[1,2]

[1] Ontotext, Sirma AI, Sofia, Bulgaria
{atanas.kiryakov,svetla.boytcheva}@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 first demonstration we implemented SPARQL queries, which retrieve similar cities in FactForge - a graph of more than 2 billion statements, combining DBPedia, GeoNames and other data. The basic methods from this paper are augmented with graph analytics and embedding techniques and formally evaluated in [2]

Keywords: Knowledge Graphs · Similarity · Graph Embedding .

1 Motivation

In a big data era, characterized by 5V[3] (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 SPARQL[4]. Defining SPARQL queries involves determining the right classes, predicates, graph patterns and filters. This is not a problem for graphs which represent uniform information for single application. However the most interesting applications of KGs involve putting together data from multiple proprietary databases,

[3] https://www.bbva.com/en/five-vs-big-data/
[4] http://www.w3.org/TR/sparql11-overview/

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 s 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 fussing 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. Thus, allow even for experts that have some awareness about the domain, but are not a top-level analyst who specialized deeply in the domain for years, to derive predictable results without defining a strict criteria.

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.

2 Knowledge Graphs and RDF

The knowledge graph represents a collection of interlinked descriptions of entities – real-world objects and events or abstract concepts (e.g., documents) – where:

- Descriptions have formal semantics that allow both people and computers to process them in an efficient and unambiguous manner;
- Entity descriptions contribute to one another, forming a network, where each entity represents part of the description of the entities, related to it, and provides context for their interpretation.

Knowledge graphs, represented using the Semantic Web standards RDF(S) and OWL, provide the best framework for data integration, unification, linking and reuse, because they combine: expressivity, efficiency and interoperability.

Resource Description Format (RDF, [4]) is the basic data-model for the Semantic Web. It allows resources to be described through relationships to other resources and literals. The resources are defined through unified resource identifiers (URI, e.g. URL). As in XML, literals are any concrete data values e.g. strings, dates, numbers, etc. The main modelling block in RDF is the statement – a triple $< subject, predicate, object >$, where:

- *subject* is the resource, which is being described;
- *predicate* is a resource, which determines the type of the relationship;
- *object* is a resource or a literal, which represents the "value" of the attribute.

A set of RDF triples represents a graph, where resources and literals are nodes and each statement is represented by an edge directed from the *subject* to the *object* and labeled with the *predicate*. So-called blank nodes can also appear in the graph, representing unique anonymous resources, used as auxiliary nodes. A sample graph, which describes a web page, created by a person called Adam, can be seen in Figure 1.

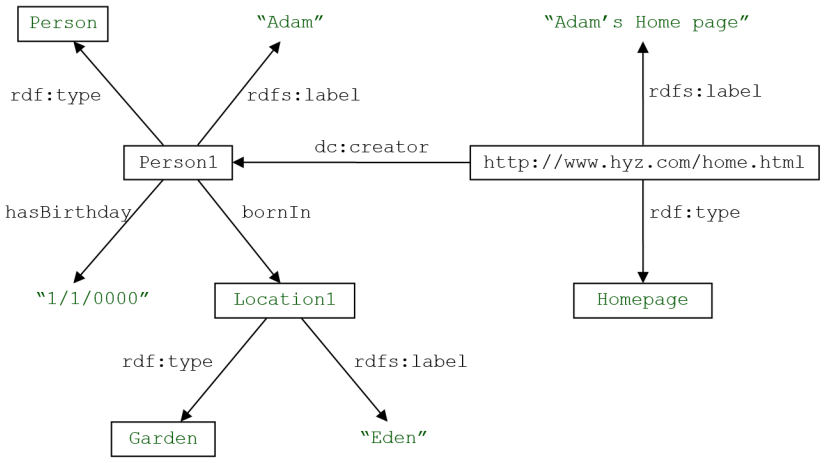

Fig. 1. RDF Graph Describing Adam and His Home Page

3 Proximity metrics

The problem of proximity measure between two objects is important in many tasks for knowledge management. The term **proximity** is used to refer, either to **similarity** or **dissimilarity**. Usually, the values are in the range $[0, \infty]$, but more often for simplicity are scaled in the range [0,1].

The term distance (metric) is used mainly to measure dissimilarity between two data objects.

Definition: A function $d : X \times X \longrightarrow R^+ \cup \{0\}$ is called **metric** on a set X iff it satisfies the following conditions:

1. $d(x, y) \geq 0, \quad \forall x, y \in X$
2. $d(x, y) = 0 \Leftrightarrow x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \geq d(x, y) + d(y, z)$

Similarity is the degree of likeness of two objects using specified metrics for the domain. For similarity the value of $s(x, y) = 1$, iff $x = y$. Where the value 1 refers to the logical value true (yes, the values are same), and 0 means false (no, the values are different). And for dissimilarity the values are just the opposite: $d(x, y) = 0$, iff $x = y$. Where value 0 means that both object match (are identical with distance 0), and value 1 means that there is significant difference between both values.

The most famous distance metric for continuous values is the Minkowski family L_p defined as:

$$L_p(x, y) = d(x, y) = (\sum_{k=1}^{n} |x_k - y_k|^p)^{\frac{1}{p}} \tag{1}$$

Frequently are used some specifications of L_p norm for $p = 1$ – Manhattan distance (or City block), for $p = 2$ – Euclidean distance and for $p = \infty$ – Chebyshev (supremum) distance also known as uniform norm:

$$L_\infty(x, y) = L_{max(x,y)} d(x, y) = \lim_{p \to \infty} (\sum_{k=1}^{n} |x_k - y_k|^p)^{\frac{1}{p}} = |x_k - y_k| \tag{2}$$

The most common relation between similarity and dissimilarity measures is:

$$s(x, y) = 1 - d(x, y) \tag{3}$$

The proximity measure techniques depend on the attributes (features) values nature, whether they are numerical, binary, or categorical. In addition, for numerical values need to be considered whether they are discrete or continuous well-ordered sets in some range, interval-scaled or ratio-scaled. Also, numerical values can be single or multiple. For instance, for geographic locations, we can have an attribute with values for latitude-longitude. On the other hand, categorical attributes can be nominal or ordinal. Some examples of proximity measures are shown in the table below (Table 1).

Typically, similarity measures have the following properties:

1. $s(x, y) = 1 \quad iff \quad x = y \quad (0 \leq s \leq 1)$
2. $s(x, y) = s(y, x) \quad \forall x, y$ (symmetry)

For non-symmetric similarity measures is used confusion matrix. In this case are used some corrections:

Attribute type	**Example**	**Dissimilarity**	**Similarity**
Continuous, Interval-scaled or Ratio-scaled	Age, Weight	$d = \lvert x - y \rvert$	$s = -d$ $s = \frac{1}{1+d}$ $s = e^{-d}$ $s = 1 - \frac{(d - d_{min})}{(d_{max} - d_{min})}$
Nominal / Binary	Gender: male/female	$d = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$ (4)	$s = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases}$ (5)
Ordinal / Categorical	Education – primary school, secondary school, high-school, undergraduate, graduate,... (Please, note that there is some ordering) Operating system: Linux, Unix, Windows, ...	$d = \frac{\lvert x - y \rvert}{n - 1}$ Where n is the number of values and values are mapped to integers in the range $[0, n-1]$	$s = 1 - d$

Table 1. Proximity metrics for different attribute types

$$s(x, y) \neq s(y, x)$$

$$s'(x, y) = s'(y, x) = \frac{s(x, y) + s(y, x)}{2}$$

Similarity measures between objects that contain only binary attributes are called similarity coefficients. Let x and y be two objects that consist of n binary attributes. Then x and y can be represented by binary vectors: $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$. The confusion matrix (Table 2) for x and y is defined as follows:

The most general definition of proximity measure is:

$$d(x, y) = \frac{\delta f_{00} + f_{11}}{\delta f_{00} + \lambda(f_{01} + f_{10}) + f_{11}} \tag{6}$$

		x	
		0	1
y	0	$f_{00} = \sum_{k=1}^{n} \overline{x_k} \overline{y_k}$	$f_{10} = \sum_{k=1}^{n} x_k \overline{y_k}$
	1	$f_{01} = \sum_{k=1}^{n} \overline{x_k} y_k$	$f_{11} = \sum_{k=1}^{n} x_k y_k$

Table 2. Confusion matrix

4 The Data

Most of the experiments in this paper are performed on FactForge[5] - a knowledge graph of Linked Open Data (LOD) and news articles about people, organizations and locations. It includes:

- DBpedia[6] (the English version) - an RDF-ized version of Wikipedia ,
- GeoNames[7] - exhaustive geographic information about populated places, countries and other features on Earth,
- The Financial Industry Business Ontology (FIBO),
- News metadata from NOW[8] - a Semantic News Portal loaded with more than 1 million international news articles in English, continuously collected since year 2015.

The entire graph contains more than 2 billion explicit statements loaded in Ontotext GraphDB – a semantic graph database engine, formerly known as OWLIM, [1]. 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.

FactForge contains about 500 thousand `owl:sameAs` mappings, most of them from GeoNames to DBPedia. For instance, `<dbr:Sofia,owl:sameAs, geodata:727011/>` states that these identifiers are equivalent, i.e. that they refer to one and the same real world entity. Effectively `owl:sameAs` binds nodes from different datasets, so, that their identifiers can be used interchangeably and all the statements made using one of the identifiers should also be valid with the other one. For instance, GeoNames states that `<geodata:727011/,gn:parentCountry,geodata:732800/>`, which is that Sofia is part of Bulgaria. Applying `owl:sameAs` reasoning GraphDB infers that `<dbr:Sofia,gn:parentCountry,dbr:Bulgaria>`. This reasoning brings great advantages when multiple datasets are combined, but as a side effect it can massively expand the results for queries, expanding each single results with all possible variations of the different `owl:sameAs`-equivalent identifiers. To avoid this expansion, one should add `FROM onto:disable-sameAs` to the SPARQL queries.

[5] http://factforge.net
[6] http://wiki.dbpedia.org/
[7] http://www.geonames.org/
[8] http://now.ontotext.com

Similarity Coefficient	δ	λ	Definition
Simple matching coefficient (SMC) (Coordination Level Matching)	1	1	$SMC = \frac{number\,of\,matching\,attribute\,values}{number\,of\,attributes}$ $SMC(x,y) = \frac{f_{00}+f_{11}}{f_{00}+f_{01}+f_{10}+f_{11}}$ For sets X and Y: $SMC(X,Y) = \lvert X \cap Y \rvert$
Sørensen–Dice Coefficient	0	0.5	$DSC(x,y) = \frac{2f_{11}}{f_{01}+f_{10}+2f_{11}}$ For sets X and Y: $DSC(X,Y) = 2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$
Jaccard's Coefficient	0	1	$J = \frac{number\,of\,matching\,presences}{number\,of\,attributes\,not\,involved\,in\,00\,matches}$ $J(x,y) = \frac{f_{11}}{f_{01}+f_{10}+f_{11}}$ For sets X and Y: $J(X,Y) = \frac{\lvert X \cap Y \rvert}{\lvert X \cup Y \rvert}$
Cosine Coefficient	-	-	$\cos(x,y) = \frac{x \cdot y}{\lVert x \rVert \cdot \lVert y \rVert}$ $\cos(x,y) = \frac{\sum_{k=1}^{n} x_k y_k}{\sum_{k=1}^{n} x_k^2 + \sum_{k=1}^{n} y_k^2}$ $\cos(x,y) = \frac{f_{11}}{\sqrt{f_{01}+f_{11}}\sqrt{f_{10}+f_{11}}}$ For sets X and Y: $\cos(X,Y) = \frac{\lvert X \cap Y \rvert}{\sqrt{\lvert X \rvert}\sqrt{\lvert Y \rvert}}$
Overlap Coefficient (Szymkiewicz–Simpson coefficient)	-	-	$overlap(X,Y) = \frac{f_{11}}{\min(f_{01}, f_{10})}$ For sets X and Y: $overlap(X,Y) = \frac{\lvert X \cap Y \rvert}{min(\lvert X \rvert, \lvert Y \rvert)}$

Table 3. Some of the most popular similarity coefficients [7]

5 Models

In RDF knowledge graphs the resources (concepts, entities, documents, etc.) are represented as nodes. Each outgoing edge contributes to the description of the resource. In other words the features, which describe the nodes, are these outgoing edges — each of them being defined by a `<predicate,object>` pair or shortly PO-pair. Naturally comes the assumption that similar nodes have similar features, i.e. common PO-pairs.

The next example (Figure 2) presents two companies, which share two PO-pairs — they are both `<located in,San Francisco>` and operate in `<industry,Aerospace>`.

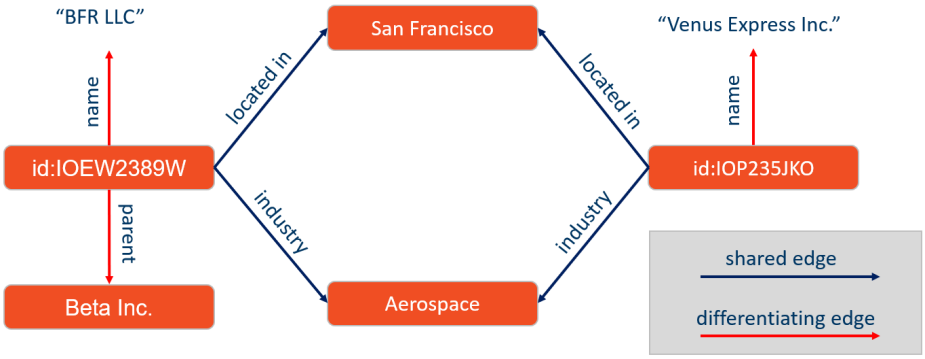

Fig. 2. Shared And Differentiating Edges

5.1 Vector Space Model

Traditionally similarity search is applied in information retrieval [3], and indexing [6] is applied for documents. The classical vector space model (VSM) was proposed by Salton et al. Lets $D = \{d_1, d_2, \cdots, d_k\}$ is a collection of documents that contain n different terms in total $T = \{t_1, t_2, \cdots, t_n\}$. There are assigned weights to terms by considering two types of information – local and global. The local information is based on the individual documents, and the global information is based on the entire collection of documents, under consideration. In the VSM [9] [10][11], documents are viewed as bag of words and are represented as n-dimensional vectors of the terms weights that they consist.

$$d_i = \langle w_{i1}, w_{i2}, \cdots, w_{in} \rangle \tag{7}$$

[9] http://www.minerazzi.com/tutorials/term-vector-1.pdf
[10] http://www.minerazzi.com/tutorials/term-vector-2.pdf
[11] http://www.minerazzi.com/tutorials/term-vector-3.pdf

Where w_ij denotes the weight of the j^{th} term in the document d_i. In the binary vector space, the weight just denotes the presence of a term in the document:

$$w_{ij} = \begin{cases} 1, \text{ if j-th term occurs in the document } d_i \\ 0, \text{ otherwise} \end{cases} \tag{8}$$

In the more sophisticated VSP the weight is defined as:

$$w_{ij} = L_{ij} G_j \tag{9}$$

where L_{ij} is the local information about the j^{th} term in the document d_i; G_j is the global information about the j^{th} term. The local information about the j^{th} term is measured as its frequency in the in the document d_i, denoted by f_{ij}. The local term weight is also called term frequency (TF). It needs a correction by global term weight, because some of the most frequent terms make the VSM vulnerable also known as spamming. Hence, a correction for the specificity is needed that is known as inverse documents frequency (IDF), i.e. discriminatory power of a term. This presentation is known as Salton's classical TF-IDF VSM. Let m is the number of the documents that contain the j^{th} term, P_j is the probability that a document from the collection D will contain the j^{th} term.

$$P(j) = \frac{m}{|D|} \tag{10}$$

$$G_j = IDF_j = -\log(P(j)) = -\log(\frac{m}{|D|}) \tag{11}$$

$$w_{ij} = -f_{ij} \log(P(j)) = f_{ij} \cdot IDF_j = TF \cdot IDF \tag{12}$$

However, in practice rare terms are not frequently queried. Thus, TF-IDF is useful for initial filtering of the document collection, but have some disadvantages, because it favorites rare terms. Although some improvement of TF-IDF were made by introducing keyword density values (KDV) (i.e. the term frequency divided by the length of the document), in general none of them is measuring the semantic weight of the terms to the documents. The idea of using KDV as term weight was abandoned[12] . Measuring the importance and relevance of a term to a document is still challenging task.

In the vector space the most common metric for similarity between documents is cosine metric – measuring the cosine of the angle between two documents vectors. An example for 3D space is presented on Figure 3.

The similarity metric between two documents $d_a = \langle w_{a1}, w_{a2}, \cdots, w_{an} \rangle$ and $d_b = \langle w_{b1}, w_{b2}, \cdots, w_{bn} \rangle$ is defined as follows:

$$s(d_a, d_b) = \cos(\theta) = \frac{d_a \cdot d_b}{\|d_a\| \cdot \|d_b\|} \tag{13}$$

Where

$$\|d_a\| = \sqrt{w_{a1}^2 + w_{a2}^2 + \cdots + w_{an}^2} \tag{14}$$

[12] http://www.e-marketing-news.co.uk/Mar05/garcia.html

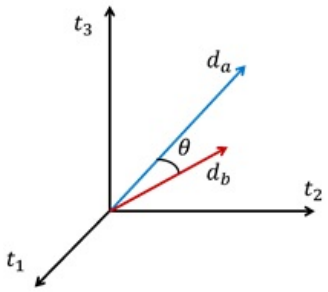

Fig. 3. Two documents in 3D vector space and the angle between their vectors

$$d_a \cdot d_b = w_{a1}w_{b1} + w_{a2}w_{b2} + \cdots w_{an}w_{bn} \tag{15}$$

Thus $\cos(\theta) = 1$ iff both documents are exactly the same, and $\cos(\theta) = 0$ for totally different documents.

There are two approaches for vector similarity searching:

- **Threshold** – to retrieve all documents that are similar to a given document $d \in D$, with similarity above a given threshold, e.g. for $th = 0.5$ to find

$$S = \{d_i \,|\, d_i \in D \quad and \quad s(d, d_i) > th\}$$

- **Ranking** – to find the top N most similar documents to a given document $d \in D$, ranked in order of similarity.

Web search engines and contemporary full-text search engines are using the second approach for page/document retrieval.

5.2 Adapting the Vector Space Model for Knowledge Graphs

To adapt VSM 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.

$$node_i = \langle po_{i1}, po_{i2}, \cdots, po_{in} \rangle \tag{16}$$

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.

6 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 queries and the results are presented in Appendix 1.

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

These experiments are extended in [2] to include weighing of PO-pairs by "popularity", graph analytics (namely PageRank) and news co-occurrence.

6.1 Identify shared edges for a node

The experiments starts with task for finding PO-pairs that given node shares with others. The SPARQL query `SH1` extracts all PO-pairs for `?node=Sofia` for which exists some `?different_node`, for which the same PO-pair appears as outgoing edge.

```
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?p ?o {
    BIND( dbr:Sofia AS ?node)
    ?node ?p ?o .
    FILTER EXISTS {
        ?different_node ?p ?o .
        FILTER(?node != ?different_node) }
}
```

The result contains 887 shared PO-pairs. Analysis of the results (Figure 4) show that the majority of the predicates of the shared pairs are `wikiPageWikiLink` modelling references between Wikipedia articles in DBPedia, `alternateName` from GeoNames, `wikiPageWikiLinkText`, `sameSettingsAs`, `wikiPageUsesTemplate`, etc.

According to the Information Theory highly expected events bear little information. Similar to the stop-words in document retrieval, some more frequent PO-pairs are excluded, because they are too general to have significant impact on the similarity measure. Thus, in this experiment we filter out PO-pairs with five of the predicates: `dbo:wikiPageWikiLinkText, dul:sameSettingAs, dbp:wikiPageUsesTemplate, rdf:type, owl:sameAs`. The updated SPARQL query is presented on Figure 5 in Appendix 1. The result contains 800 PO-pairs, i.e. 87 PO-pairs were filtered out.

6.2 Nodes similarity as count of shared edges

Coordination level matching (CLM) assumption for KG: Two nodes are more similar if they share more PO-pairs.

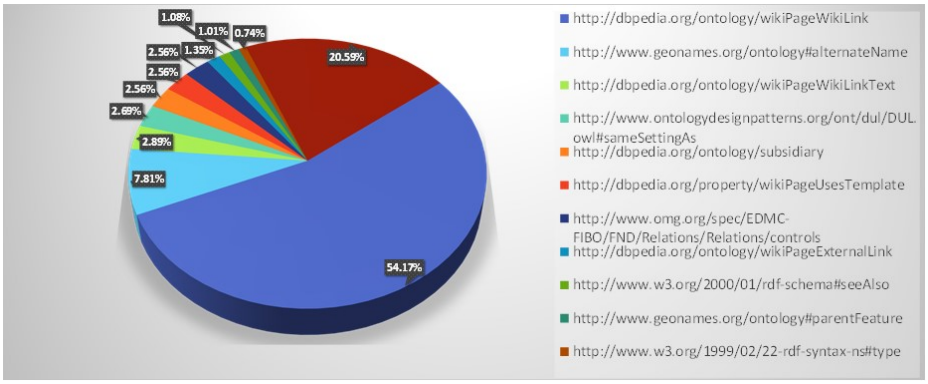

Fig. 4. Distribution of PO-pairs by Predicates

```
12  SELECT DISTINCT ?p ?o {
13      BIND( dbr:Sofia AS ?node)
14      ?node ?p ?o .
15      FILTER EXISTS { ?different_node ?p ?o .
16                      FILTER(?node != ?different_node) }
17      FILTER(?p NOT IN (rdf:type, owl:sameAs, dul:sameSettingAs,
18              dbo:wikiPageWikiLinkText, dbp:wikiPageUsesTemplate))
19  }
```

Fig. 5. Query SH2: Shared edges, filtered by predicate

We retrieve the top-100 nodes, most similar to a given one, based on the count of the shared PO-pairs. This way we resemble the Simple matching (coordination level matching, CLM) [5], which assumes that documents whose index records have n different terms in common with the query are more likely relevant than documents with $n-1$ different terms held in common.

In query SH3 (Figure 6) we count the distinct PO-pairs that `?node=Sofia` shares with other nodes. We start with sub-query similar to SH2 to get the first 500 distinct pairs, which Sofia shares with other nodes. Next all nodes that share at least one of these PO-pairs with `?node=Sofia` are grouped and sorted in descending order by the number of shared PO-pairs. The top 10 results (Figure 7) results come close to what we expected: Varna, Plovdiv, Bulgaria, Europe, History of Bulgaria, etc. While we are getting cities which are really similar to Sofia, we also get countries, continents and historical articles, which is not what we expected. To overcome this in SH4 we set additional restriction on the type of the `?similar_node` to be `dbo:City` (Figure 8).

The results (Figure 9) show significant improvement: Plovdiv, Skopje, Budapest, Thessaloniki, Zagreb, etc. This are all similarly sized cities in Eastern Europe. Note that the query does not use geographic proximity, population or

```
SELECT ?similar_node (COUNT(DISTINCT ?pair) AS ?shared_pairs) {
    { SELECT DISTINCT ?node ?p ?o {
        BIND( dbr:Sofia AS ?node)
        ?node ?p ?o .
        FILTER EXISTS { ?different_node ?p ?o .
                FILTER(?node != ?different_node) }
        FILTER(?p NOT IN ( rdf:type, owl:sameAs, dul:sameSettingAs,
                    dbo:wikiPageWikiLinkText, dbp:wikiPageUsesTemplate))
    } LIMIT 500 }

    ?similar_node ?p ?o . FILTER(?similar_node != ?node)
    BIND(CONCAT(STR(?p), STR(?o)) AS ?pair)
} GROUP BY ?similar_node ORDER BY DESC(?shared_pairs) LIMIT 100
```

Fig. 6. Query SH3: Similar nodes by count of shared edges

	similar_node	shared_pairs
1	dbr:Varna	"74"^^xsd:integer
2	dbr:Plovdiv	"68"^^xsd:integer
3	dbr:Bulgaria	"65"^^xsd:integer
4	dbr:Europe	"61"^^xsd:integer
5	dbr:History_of_Bulgaria	"61"^^xsd:integer
6	dbr:Skopje	"54"^^xsd:integer
7	dbr:Budapest	"53"^^xsd:integer
8	dbr:Timeline_of_country_and_capital_changes	"53"^^xsd:integer
9	dbr:Chiprovtsi	"52"^^xsd:integer
10	dbr:Thessaloniki	"52"^^xsd:integer

Fig. 7. Results of SH3: Nodes similar to Sofia by count of shared edges

another other formal criteria. We achieved one of our goals - to retrieve similar objects without exploration of the structure of the graph in order to figure out which specific predicates and patterns should be used in the query. And we did so with a fairly simple query, which implements a very basic IR technique and takes only couple of seconds to evaluate against a graph of 2 billion statements.

```
SELECT ?similar_node (COUNT(DISTINCT ?pair) AS ?shared_pairs) {
    { SELECT DISTINCT ?node ?p ?o {
        BIND( dbr:Sofia AS ?node)
        ?node ?p ?o .
        FILTER EXISTS { ?different_node ?p ?o .
                FILTER(?node != ?different_node) }
        FILTER(?p NOT IN ( rdf:type, owl:sameAs, dul:sameSettingAs,
                dbo:wikiPageWikiLinkText,dbp:wikiPageUsesTemplate))
    } LIMIT 500 }

    ?similar_node ?p ?o . FILTER(?similar_node != ?node)
    ?similar_node a dbo:City .
    BIND(CONCAT(STR(?p), STR(?o)) AS ?pair)
} GROUP BY ?similar_node ORDER BY DESC(?shared_pairs) LIMIT 100
```

Fig. 8. Query SH4: Similar nodes by count of shared edges, typed

	similar_node	shared_pairs
1	dbr:Plovdiv	"68"^^xsd:integer
2	dbr:Skopje	"54"^^xsd:integer
3	dbr:Budapest	"53"^^xsd:integer
4	dbr:Thessaloniki	"52"^^xsd:integer
5	dbr:Zagreb	"43"^^xsd:integer
6	dbr:Belgrade	"41"^^xsd:integer
7	dbr:Ljubljana	"41"^^xsd:integer
8	dbr:Prague	"38"^^xsd:integer
9	dbr:Split,_Croatia	"38"^^xsd:integer
10	dbr:Bucharest	"37"^^xsd:integer

Fig. 9. Results of SH4: Nodes similar to Sofia by count of shared edges, typed

7 Discussion

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 11^{th} or 12^{th} which does not make similarity measure less accurate. Thus the golden standard 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.

8 Conclusion and Further Work

We demonstrated that IR techniques can be adapted to perform search for similar nodes in a knowledge graph. We implemented the VSM, treating each node in the graph as document and its outgoing edges predicate-object pairs as terms, which describe it. Reasoning also contributed to the final result by increasing the connectivity in the graph and making sure that commonalities between nodes will not be missed, because, for instance, a more specific predicate (sub-property) has been used for one node and more general one for the other.

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 and such trivial IR techniques may not be able to bring decent results on graphs which are less interconnected and concise.

These experiments are extended in [2] to include weighing of PO-pairs by "popularity", graph analytics (namely PageRank) and news co-occurrence and perform proper evaluation.

While it delivers useful results, when tuned for a specific case (similarity of cities), 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 in [2] 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.

References

1. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: Owlim: A family of scalable semantic repositories. Semantic Web **2**(1), 33–42 (2011)
2. Boytcheva, S., Kiryakov, A., Gyurov, P.: Similarity search in knowledge graphs: Vectorspace model and embedding. In: Knowledge, Language Models, (eds. Milena Slavcheva, Kiril Simov, Petya Osenova, Svetla Boytcheva). Incoma (2020)
3. Hliaoutakis, A., Varelas, G., Voutsakis, E., Petrakis, E.G., Milios, E.: Information retrieval by semantic similarity. International journal on semantic Web and information systems (IJSWIS) **2**(3), 55–73 (2006)
4. 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)
5. Losee, R.: Probabilistic retrieval and coordination level matching. Journal of the American Society for Information Science **38**(4), 239–244 (1987)
6. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18**(11), 613–620 (1975)
7. Vijaymeena, M., Kavitha, K.: A survey on similarity measures in text mining. Machine Learning and Applications: An International Journal **3**(2), 19–28 (2016)

A Appendix 1: SPARQL Queries

This paper provides queries that implement VSM-like node similarity in FactForge along with snapshots from query results. FactForge represents a graph of over 2 billions statements (see section 4) loaded in GraphDB and publicly available for exploration at `http://factfroge.net` These queries are also available in FactForge as Saved queries.

Follows a list of prefixes used across the queries

```
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 dct: <http://purl.org/dc/terms/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX ff-map: <http://factforge.net/ff2016-mapping/>
PREFIX wd: <http://www.wikidata.org/entity/>
```