Random Indexing for Finding Similar Nodes within Large RDF graphs

Danica Damljanovic[1], Johann Petrak[2], Mihai Lupu[3], Hamish Cunningham[1],
Mats Carlsson[4], Gunnar Engstrom[4], and Bo Andersson[4]

[1] Department of Computer Science, University of Sheffield, United Kingdom
d.damljanovic@dcs.shef.ac.uk, h.cunningham@dcs.shef.ac.uk
[2] Austrian Reseach Institute for Artificial Intelligence, Vienna, Austria
johann.petrak@ofai.at
[3] Information Retrieval Facility (IRF), Vienna, Austria
m.lupu@ir-facility.org
[4] AstraZeneca, Lund, Sweden
Mats.Carlsson, Gunnar.Engstrom, Bo.H.Andersson@astrazeneca.com

Abstract. In this paper, we propose an approach for searching large
RDF graphs, using advanced vector space models, and in particular, Random Indexing (RI). We first generate documents from an RDF Graph,
and then index them using RI in order to generate a semantic index,
which is then used to find similarities between URIs, literals, and RDF
subgraphs. We have experimented with large RDF graphs in the domain
of life sciences and engaged the domain experts in two stages: firstly, to
generate a set of keywords of interest to them, and secondly to judge
on the quality of the output of the Random Indexing method, which
generated a set of similar terms (literals and URIs) for each keyword of
interest.

Key words: random indexing, vectors space models, information retrieval, RDF graphs, ontologies

1 Introduction

Recent years have seen a massive increase of highly structured data being made
available in the form of RDF triple representations. Both legacy data and new
data have been made available in RDF triple format and this representation has
also made it worthwhile and feasible to create mappings between RDF data that
originates from different legacy sources, leading to potentially very large RDF
repositories. Initiatives such as Linked Open Data[5] are working on creation,
publication and interlinking of huge RDF graphs.

Traditionally, RDF spaces are being searched using an RDF query language
such as SeRQL [2] or SPARQL [15]. These languages allow the formulation of
fine-grained queries by their ability to match whole graphs and to create complex
conditions on the variables to be bound in the query. This level of complexity

---

[5] http://linkeddata.org/

and flexibility is very useful in many situations, especially when the query is created automatically in the context of an application. However, for end-users who want to explore the knowledge represented in an RDF store, this level of detail is often more of a hindrance: querying the repository is not possible without a detailed knowledge of its structure and the names and semantics of all the properties and classes involved. This is especially the case for large and unknown data structures which may have thousands of classes and properties, for example Linked Life Data[6] (5 billion statements), or FactForge[7] (2 billion statements).

In this paper we investigate whether advanced Information Retrieval (IR) methods can bring a new dimension to the task of searching huge RDF graphs. We propose a complementary approach based on word space model, more concretely Random Indexing (RI) [14], for building a *semantic index* for a large RDF graph. Traditionally, a *semantic index* captures the similarity of *terms* based on their contextual distribution in a large document collection, and the similarity between *documents* based on the similarities of the terms contained within. By creating a semantic index for an RDF graph, we are able to determine contextual similarities between graph nodes (e.g., URIs and literals) and based on these, between arbitrary subgraphs. These similarities can be used for finding a ranked list of *similar* URIs/literals for any given input term (a literal or a URI), which can then be used for exploring the repository or enriching SPARQL queries.

We evaluate our approach on subsets of the Linked Life Data (LLD) repository – a large integrated repository which contains 5 billion RDF statements from various sources covering the biomedical domain, including UniProt[8], PubMed[9], EntrezGene[10] and many more[11]. Our evaluation is based on human judgment by clinical research scientists (from AstraZeneca pharmaceutical company) who were involved in two stages: firstly, to generate a set of keywords of interest to them, and secondly to judge on the quality of the output of the Random Indexing method, which generated a set of similar terms (literals and URIs) for each topic of interest.

2 Related work

A considerable amount of work has been done in the area of using Information Retrieval methods for the task of selecting and retrieving RDF triples. However, most of these approaches do not take advantage of the latent semantics included in an RDF Graph, as their primary intention is finding the RDF files on the Web relevant to the given keyword and/or a URI. These systems are semantic search engines such as Swoogle [9] or Sindice ([18]). They collect the Semantic

---

[6] `www.linkedlifedata.com`

[7] `http://factforge.net`

[8] `www.uniprot.org/`

[9] `http://www.ncbi.nlm.nih.gov/PubMed/`

[10] `www.ncbi.nlm.nih.gov/sites/entrez?db=gene`

[11] see the full list at: `www.linkedlifedata.com/sources`

Web resources from the Web and then index the keywords and URIs against the RDF files containing those keywords and URIs, using the inverted index scheme. These search engines use traditional weighting mechanisms such as TF-IDF, and in [11] the authors introduce the *ReConRank* algorithm, which adapts the well-known PageRank algorithm to Semantic Web data. This method ranks the nodes in a topical subgraph that is selected based on keyword matching from the RDF files. In other words, it ranks the results of a query based on the RDF links in the results. The subgraph that the algorithm identifies includes both the subject nodes related to the query, and also the context of the subject nodes (i.e. the provenances or sources of the subjects), in order to improve the quality of ranking.

In comparison to these approaches we use the neighbouring nodes as semantic context for each node in an RDF graph. The nodes and their contexts are used as virtual documents for Random Indexing.

In [16], the authors describe an approach for generating a virtual document for each URI reference in an RDF triple store (or, equivalently, each node in an RDF graph). The virtual document contains the local name and labels of the URI reference, other associated literals such as those in *rdfs:comment*, and the names of neighbouring nodes in the RDF graph. These virtual documents are then used for ontology matching and also for generating object recommendations for users of Falcons [3]. In comparison to our approach, their neighbouring operations involve only one-step neighbours without including properties. Our approach includes properties, and parts of the TBox, and also can operate on an arbitrarily large graph of neighbouring nodes.

Finally, to the best of our knowledge, none of the similar approaches investigate the usage of methods that can discover latent semantics, such as Random Indexing.

3 Semantic Index

Latent Semantic Analysis (LSA) [8] is one of the pioneer methods which has been used for finding synonyms. The assumption behind this and other statistical semantics methods is that words which appear in the similar context (with the same set of other words) are synonyms. Synonyms tend not to co-occur with one another directly, so indirect inference is required to draw associations between words which are used to express the same idea [4]. This method has been shown to approximate human performance in many cognitive tasks such as the Test of English as a Foreign Language (TOEFL) synonym test, the grading of content-based essays and the categorisation of groups of concepts (see [4]). However, one problem with this method is scalability: it starts by generating a $term * document$ matrix which grows with the number of terms and the number of documents and will thus become very large for large corpora. For finding the final LSA model, Singular Value Decomposition (SVD) and subsequent dimensionality reduction is commonly used. This technique requires the factorization of the term-document matrix which is computationally costly and does not scale well. Also, calculating

the LSA model is not easily end efficiently doable in an incremental or out-of memory fashion. The Random Indexing (RI) method [17] circumvents these problems by avoiding the need of matrix factorization in the first place.

RI can be seen as an approximation to LSA which is shown to be able to reach similar results (see [14] and [5]). RI can be incrementally updated and also, the $term * document$ matrix does not have to be loaded in memory at once – loading one row at the time is enough for computing context vectors. Instead of starting with the full term-document matrix and then reducing the dimensionality, RI starts by creating almost orthogonal random vectors (index vectors) for each document. This random vector is created by setting a certain number of randomly selected dimensions to either +1 or -1. Each term is represented by a vector (term vector) which is a combination of all index vectors of the document in which it appears. For an object consisting of multiple terms (e.g. a document or a search query with several terms), the vector of the object is the combination of the term vectors of its terms.

Random Indexing relies on the Johnson-Lindenstrauss lemma:

Lemma 1. *Given $0 < \epsilon < 1$, a set X of m points in R^N, and a number $n > n_0 = O(\frac{log(m)}{\epsilon^2})$, there exists a mapping $f : R^N \to R^n$ such that $(1-\epsilon)||u-v|| \leq ||f(u) - f(v)|| \leq (1+\epsilon)||u-v||$, for all $u, v \in X$.*

and particularly on the proof provided by Johnson and Lindenstrauss in their 1984 article [13], where they show that if one chooses at random a rank n orthogonal projection, then, with positive probability, the projection restricted to X will satisfy the condition in the Lemma. RI relies on the observation that, in a high dimensional space, a random set of vectors is always almost orthogonal.

In order to apply RI to an RDF graph we first generate a set of documents which represent this graph, by generating one *virtual document* for each URI in the graph (Section 3.1). Then, we generate a semantic index from the virtual documents (Section 3.2). This semantic index is then being searched in order to retrieve similar literals/URIs (Section 3.3).

3.1   Generating virtual documents

The task of deriving a set of documents from a huge RDF graph starts with generating a *representative subgraph* for each URI of interest. We shall refer to such an URI as a *representative URI*.

A representative subgraph represents the context of a URI i.e. the set of other URIs and literals directly or indirectly connected to that URI. For a representative URI S, the representative subgraph of order N is a set of all paths of triples $(S, P_1, O_1; O_1, P_2, O_2; \cdots ; O_{N-1}, P_N, O_N)$. If O_N is not a literal we also include all triples O_N, P_{N+1}, L_J where L_J is a literal. In other words, we apply the Breadth-First-Search starting with the representative node, and extend this to the Depth Search which is defined by N. In addition, we include or exclude certain parts of the TBox: direct classes for instances are excluded

($P_N! = rdf : type$), while other annotation properties such as $rdfs : label$ are included. In the experiments reported in this paper, the representative subgraphs are of order 1 ($N = 1$).

We create *virtual documents* by including all paths from representative subgraphs where:

– all URIs of nodes or appearing inside literals are included unchanged;
– for literals we remove punctuation and stop words, and then lowercase the text; we also remove number literals, gene and protein sequences, complex names, and HTML tags.

3.2 Generating semantic index

There are several parameters which can influence the process of generating semantic index, or vectors using the RI method:

– **Seed length** Number of +1 and -1 entries in a sparse random vector.
– **Dimensionality** Dimension of the semantic vector space – predefined number of dimensions to use for the sparse random vectors.
– **Minimum term frequency** Minimum frequency of a term to get included in the index.

Our experiments study how variations of these parameters influence the quality of the results and how sensitive the method is to that variation.

3.3 Search

Once the semantic index has been created, it can be used to find similarities between URIs, literals, and RDF subgraphs. We use the cosine function to calculate the similarity between the input term (literal or URI) vector and the existing vectors in the generated vector space model. We can perform the following kinds of searches:

1. *finding similarities between two terms*: given a keyword, find similar literals and URIs; this can be used in several ways for example for refinement of SPARQL queries (see [7]); also, it can be used as an alternative way of browsing and finding URIs or literals related to a topic of interest (expressed through a keyword or a set of keywords)
2. *finding documents related to a specific term*: this task would be useful for suggesting a set of representative URIs related to a given keyword.
3. *finding documents related to a document*: this task would be useful for suggesting a set of representative URIs related to a set of URIs.
4. *finding terms related to the specific documents*: this can be used for describing a representative URIs through a set of literals and URIs.

While in the context of large RDF graphs such as LLD, we find all of these searches useful, in the experiments we present next, we focus on term-term search (Item 1) only. As the LLD dataset covers the life sciences domain, we have conducted a study with the clinicians from AstraZeneca, who are domain experts and understand the knowledge available in this large dataset.

4 Experiments

Our goal in using the Random Indexing method is to investigate whether it can offer an alternative way of searching large RDF spaces, by suggesting literals or URIs which are similar to the topic of interest. We conduct an evaluation experiment with clinical research scientists from AstraZeneca, with the aim to assess this.

4.1 Dataset

Linked Life Data is a dataset covering the life sciences domain, and the latest version 0.6 contains 5,052,047,661 statements in total (for a comparison, one year ago it contained 4,179,999,703 statements). Advanced IR methods based on Vector Space Model (VSM) are computationally expensive, and therefore, before we apply the Random Indexing method on the whole dataset, we evaluate it on two smaller subsets of LLD.

We have generated the two subsets as follows. For 1528 seed URIs (the URIs representing all MEDLINE articles from December 2009) we retrieve neighbouring subgraphs (of order 1) recursively until we reach certain predefined limit of statements, and we refer to these as LLD1 and LLD2. Table 1 shows the sizes of LLD1 and LLD2.

	LLD 1	LLD2
number of statements	595798	4573668
number of virtual documents	64644	473742
number of terms	417753	1713349

Table 1. Sizes of LLD1 and LLD2 datasets

4.2 Evaluation measures

In order to calculate the correctness of the retrieved terms, there are standard Information Retrieval measures such as *precision*, *recall* and *Mean Average Precision (MAP)*. Precision is defined as the number of relevant documents retrieved divided by the total number of documents retrieved and is usually calculated for certain number of retrieved documents (e.g., Precision@10, Precision@20). Recall is the number of relevant documents retrieved divided by the total number of existing relevant documents (which should have been retrieved).

Mean Average Precision (MAP) is by far one of the most popular measures in IR evaluation because, for each system and set of topics, it provides a single value to measure its performance [6]. Average Precision (AP) is computed for each topic by first calculating precision for each relevant document that is retrieved and then averaging these values. Mean Average Precision is then the

mean of these values for all keywords. Furthermore, by the nature of the averaging process, MAP is more sensitive to ranking than *precision* at a specific point, favouring systems which return more relevant documents at the top of the list than at the bottom, whereas *precision* does not make this distinction as long as the results are within the cut-off range.

As our task is to retrieve most relevant literals and URIs first, we used MAP@10. Recall is extremely difficult to measure due to the number of terms in our datasets (see Table 1). In addition, our task is to help domain experts explore large RDF graphs, which is similar to Web search in the sense that there is a vast amount of terms to be searched through, and also a significant number which is relevant for each input term. Hence, for these kinds of tasks, users care more about precision than about recall. Indeed, they care most about the top ranked results, which is exactly what is captured by MAP.

Relevance of retrieved terms was evaluated by two clinical research scientists. All scientists looked at all retrieved terms. *Relevant* were considered only those terms which *both* scientists marked as relevant. In order to measure agreement between scientists on this particular task, we measured the Inter Annotator Agreement (IAA) between the two clinicians based on the words which both of them marked as relevant/irrelevant.

IAA has been used mainly in the classification tasks, where two or more annotators are given a set of instances and are asked to classify those instances into some pre-defined categories. The two commonly used IAA measures are *observed agreement* and *Kappa (κ)* [12].

Observed agreement is the portion of the instances on which the annotators agree. For our case, with the two annotators and two categories (relevant and irrelevant), it is defined as

$$A_o = \frac{a + d}{a + b + c + d} \tag{1}$$

where a refers to the number of terms *both annotators agreed as relevant*, d refers to the number of terms *both agreed as irrelevant*, b refers to the number of terms *annotator 1 marked as relevant, and annotator 2 as irrelevant*, c refer to the number of terms *annotator 1 marked as irrelevant, and annotator 2 as relevant*.

A certain amount of agreement is expected by chance which is not captured by the observed agreement. The Kappa measure is a chance-corrected agreement. **Kappa** is defined as the observed agreements A_o minus the agreement expected by chance A_e and is normalized as a number between -1 and 1.

$$k = \frac{A_o - A_e}{1 - A_e} \tag{2}$$

$k = 1$ means perfect agreement, $k = 0$ means the agreement is equal to chance, $k = -1$ means 'perfect' disagreement.

There are two different methods for estimating A_e: in **Cohen's Kappa**, each annotator has a personal distribution, based on his distribution of categories. In **Siegel & Castellans Kappa**, there is one distribution for all annotators,

derived from the total proportion of categories assigned to all annotators (see [10] for more details and for the comparison of the two). We used Cohen's Kappa in our experiments.

4.3 Experimental setup

We have performed our experiment through the following steps:

1. **Extracting topics of interest** represented as query terms which are present in both LLD1 and LLD2. In order to avoid exposing the scientists to learning SPARQL, we have formed a team of one computer scientist and one clinical research scientist. The computer scientist was executing the SPARQL query and browsing through the links and URIs, while the clinical research scientist was only looking at the abstracts which the computer scientist selected. As a result, we obtained 18 keywords which all appeared in both LLD1 and LLD2 datasets. We split this set into two halves as shown in Table 2 , and then perform the following two steps in two iterations: first, *Group 1* is used for *training* the model, and *Group 2* for *testing* it. In the second iteration, the two sets are swapped.

Group 1	Group 2
acetylcholinesterase	Posttraumatic Stress Disorder
synergistic effect	trial
cholinergic signaling	bladder cancer
PTSD	Adverse events
antagonist	trauma
efficacy	antioxidant
clinical trial	magnesium
cognitive	cystectomy
lung	5-HT receptors

Table 2. Topics of interest divided into two groups for training/testing the Random Indexing method

2. **Training the model**: we generated RI models for several variations of the following RI parameters for both LLD1 and LLD2:
   - vector dimension: 500, 1000, 1500, 1800, 2500
   - seed length: 10, 50, 100, 300, 500, 1000
   - term frequency: 1, 2, 5, 8, 10

   This resulted in 290 runs (145 per dataset[12]). We then searched for similar words for each topic of interest from the *training* set, and presented them to clinicians who accessed the relevance. The combinations for parameters which lead to the best results (measured through MAP) were considered as the optimal setting for testing the method in the next step.
3. **Testing the model**: for the models generated using the optimal parameters retrieved in the previous step, we retrieved 10 similar words for each topic

---

[12] 5 runs are missing from this count, corresponding to the situation where the seed size is 1000, and the vector dimensionality is 500, which is impossible

of interest from the *testing* set and calculated MAP. The correctness of the retrieved terms was assessed by clinical research scientists to whom we gave the terms in the form of a survey (see below).

Human assessment The retrieved keywords for each topic of interest in both *training* and *testing* sets were assessed by humans. We merged the results from all searches into one pool, and gave this list to the scientists in the form of a survey. When the similar term was a URI, we have extracted the label from LLD and showed it in brackets. This is to ensure that the scientists can concentrate on meaning of these rather than looking and searching LLD in order to find the label. An example task looked similar to this:

```
-----------------------------------------------------------
Is 'trauma' related to (delete URIs/words which are not related):
-----------------------------------------------------------
arteriopathy
back-projection
barotraumas
gunshot
http://linkedlifedata.com/resource/umls/id/C0003048 (Animal
                                              Experimentation)
http://linkedlifedata.com/resource/umls/id/C0004601 (Back Injuries)
http://linkedlifedata.com/resource/umls/id/C0005604 (Birth trauma)
............
```

The most difficult task when designing this experiment was to define the meaning of *relevant*. Relevant, in this context, is any word related to the given keyword. This is a quite broad definition, which has, as it has been reported by clinical research scientists who were involved in this experiment, posed a number of difficulties due to many different levels of relevance. One of them stated that it would not be easy to repeat the same tasks and mark the same words as relevant if they had to repeat the same task again. We consider those that are not deleted as relevant. Only those words which have been marked as relevant twice (by two different clinicians) were eventually used when evaluating our results.

4.4   Results

In this section we first look into the results of training the model and finding the best parameters with two separate groups independently. Then, we look at the results of testing the RI method using these best parameters.

Training the model We expect to see variations of MAP, for different values of dimensionality, seed length, and minimum term frequency parameters. Our goal is to find the combination of parameters for which MAP is highest, so as to use those in the testing phase.

Figure 1 shows the distribution of MAP across all cases, and for each group used for training. It seems that the keywords from *Group 1* were more challenging for the method, as MAP values are much lower on average. However, as we can
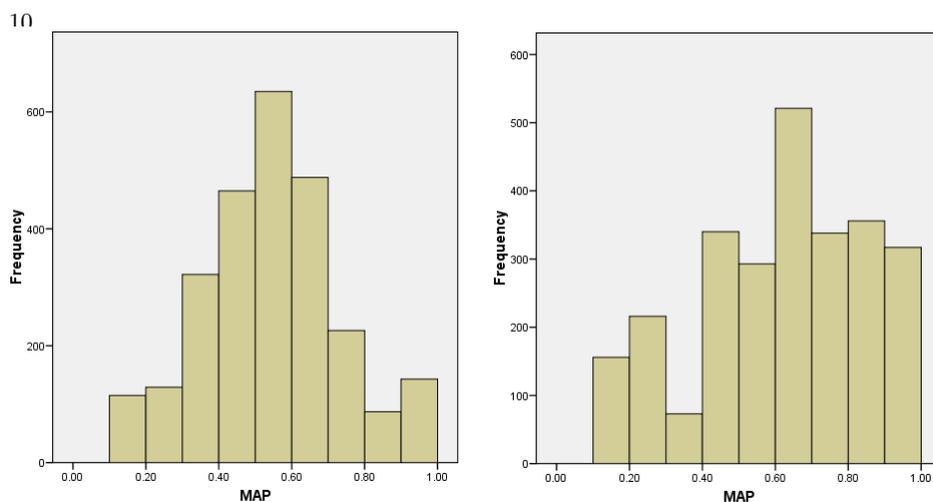
Fig. 1. The distribution of the Mean Average Precision for all combinations of parameters, for *Group 1* (left) and *Group 2* (right) used as training sets

see in Table 3 results for *Group 1* were better with LLD2 in comparison to LLD1, while for *Group 2* results were better with LLD1. The reason is a high difference in MAP for keywords: *5-HT receptors*, *trauma* and *trial*. All other keywords from Group 1 performed similarly for both datasets. However, looking closely into results of *Group 2* and the differences of MAP per keyword, there is a fluctuation with one half performing better with LLD1, while the other half performing better with LLD2 (see scattergram in Figure 2).

|	Dataset	Number of runs	Mean	Std. Deviation
Group 1	1	1305	0.50	0.16
	2	1305	0.59	0.19
	1&2	2610	0.54	0.18
Group 2	1	1305	0.65	0.22
	2	1305	0.59	0.24
	1&2	2610	0.62	0.23

Table 3. The dispersion values for the distribution of MAP across two datasets

Looking closely into the effect of parameter variations, considering LLD1 and LLD2 groups independently, the results reveal that variations of the parameters did not have major influence on MAP. Detailed diagrams outlining the influence of the variation of all three parameters are shown in Figures 3, 4, and 5.

Although parameter variations seem not to have significant influence on MAP, there are certain patterns which are visible. Namely, the best minimum frequency parameter is in the bottom range (1 for *Group 2* and 2 for *Group 1*) for the smaller dataset, while for the larger, it seems to be in the top (10 for both groups). This might be an explanation for MAP being lower for the larger dataset: more data causes more noise which seems to be filtered nicely using the minimum frequency parameter.
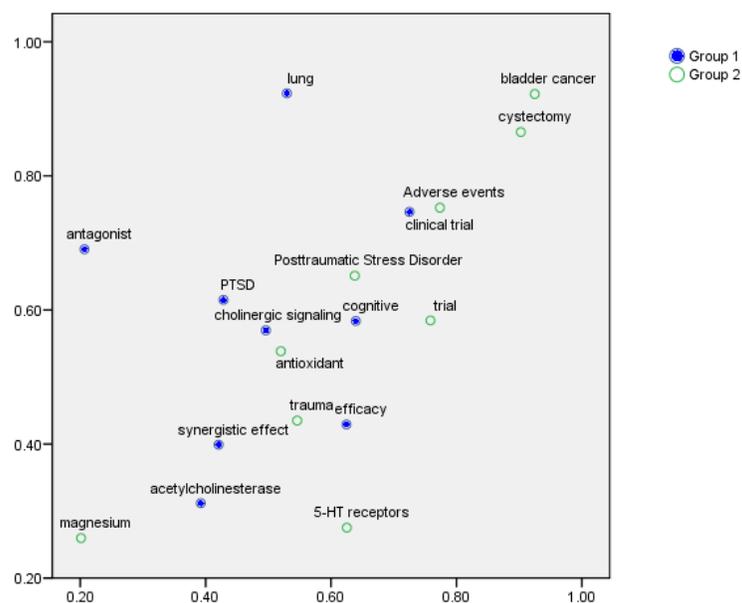
Fig. 2. Correlation of MAP for LLD1 (X axis) and LLD2 (Y axis) for all keywords
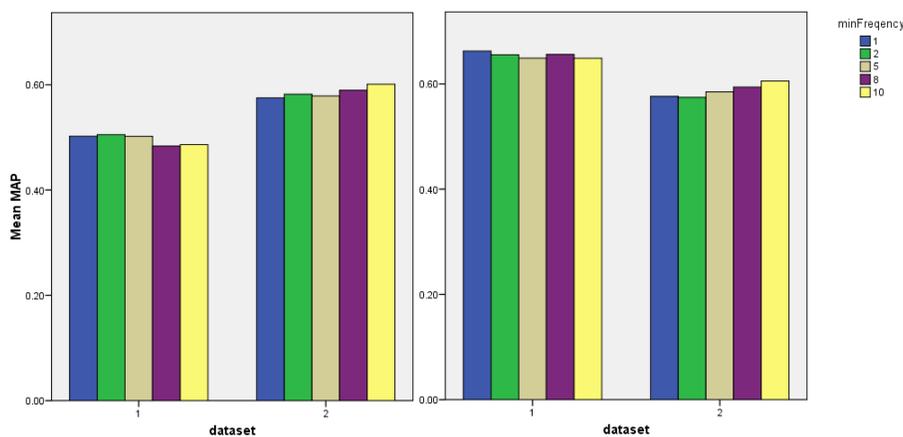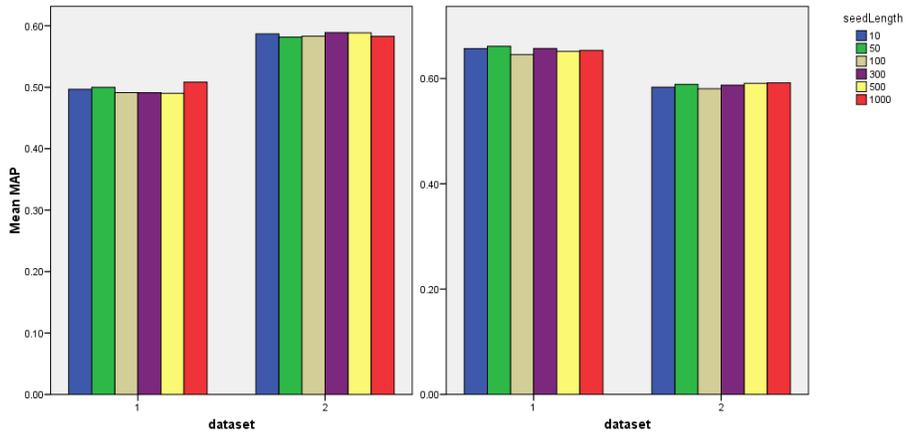

Fig. 3. The effect of the variation of minimal term frequency on *MAP*, across two datasets, for *Group 1* (left) and *Group 2* (right) used as training sets. The distribution of MAP across all categories of minimum term frequency is the same (independent samples Kruskal-Wallis test, p=0.44 and p=0.444 for LLD1 and LLD2 respectively, Group 1; p=0.808 and p=0.784 for LLD1 and LLD2, Group 2).

With regards to dimensionality, its variation has more influence on MAP with the smaller dataset, than with the larger one. This indicates that the value span which we chose for dimensionality parameter for LLD2 needs to be expanded in order to have any effect on results. However, MAP values are within reasonable range.
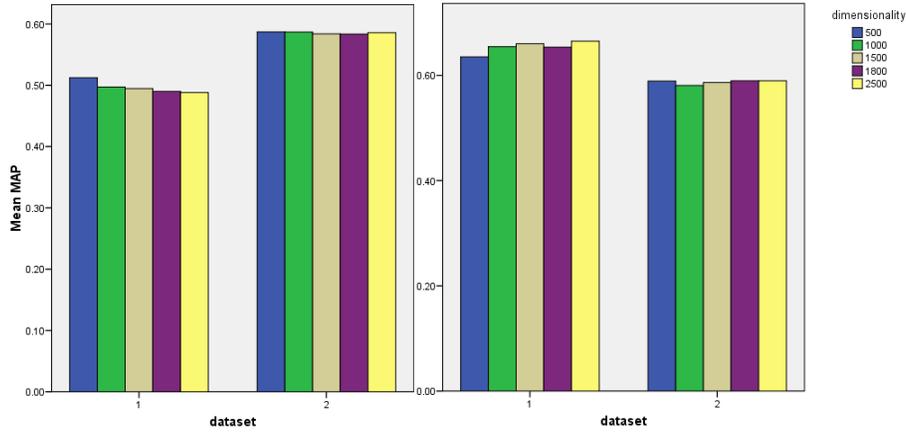
Fig. 4. The effect of the variation of dimensionality on *MAP*, across two datasets, for *Group 1* (left) and *Group 2* (right) used as training sets. The distribution of MAP across all categories of dimensionality is the same (independent samples Kruskal-Wallis test, p=0.676 and p=1.0 for LLD1 and LLD2 respectively, Group 1; p=0.587 and p=0.996 for LLD1 and LLD2, Group 2).

Fig. 5. The effect of the variation of seed length on *MAP*, across two datasets, for *Group 1* (left) and *Group 2* (right) used as training sets. The distribution of MAP across all categories of minimum term frequency is the same (independent samples Kruskal-Wallis test, p=0.931 and 0.997 for LLD1 and LLD2 respectively, Group 1; p=0.961 and 0.998 for LLD1 and LLD2, Group 2).

The variation of seed length parameter value seems not to cause any significant changes to MAP across both datasets, and hence, we consider the lowest value of this parameter as the optimal one, due to the fact that the the computational resources required to build and search the semantic space are proportional to the value of seed length. Table 4 outlines optimal parameters: those that we chose to use in the testing phase.

	Group 1		Group 2	
Dataset	LLD1	LLD2	LLD1	LLD2
Min frequency	2	10	1	10
Seed length	10	10	10	10
Dimensionality	500	500	1500	500
MAP	0.55	0.61	0.65	0.61

Table 4. Optimal parameters chosen for *Group 1* and *Group 2* used as training sets

Finally, the size of the dataset had a significant influence on MAP (Mann-Whitney U Test, $p < 0.0001$) for both Group 1 and 2 meaning that the larger set (LLD2) resulted in producing the higher value of MAP for Group 1, while for Group 2 the results were better with the smaller dataset (LLD1).

Testing the model In what follows we explore whether the model built using the optimal parameters just presented can be used to effectively test the model. In our context, testing the model means evaluating the set of related terms (literals and URIs) returned by our method for the set of testing keywords given as input.

We ran the search method using *Group 2* as a *testing* set against the RI model trained with *Group 1*, and then *Group 1* as a *testing* set against the RI model trained with *Group 2*. Results are shown in Table 5. The RI method results in as good or better MAP for *Group 2* in comparison to MAP for the best trained model (*Group 1* column in Table 4), while for *Group 1* the resulting MAP for LLD2 is as good as that of the best trained model (*Group 2* column in Table 4), while for LLD1 it is lower for 0.15. This is due to the distribution of keywords in *Group 1*, due to which MAP for the RI model with optimal parameters is only 0.05 higher (0.55).

In the testing phase, MAP across both groups reached 0.565 and 0.61 for LLD1, and LLD2 respectively.

	Group 2		Group 1	
Dataset	LLD1	LLD2	LLD1	LLD2
Min frequency	2	10	1	10
Seed length	10	10	10	10
Dimensionality	500	500	1500	500
MAP	0.63	0.61	0.5	0.61

Table 5. Testing the Random Indexing method using *Group 2* and *Group 1* as *testing* sets

Also important to observe is the fact that when the data corpus increases (e.g. LLD2 vs LLD1) the method becomes very stable, and observed MAP values in the training process are reproduced in the subsequent test phase. Arguably this is due to the small difference in MAP across parameters, but it still shows that RI is a stable method even in this unusual use-case we are dealing with.

Human assessment. In order to assess overall difficulty of the tasks which we solve using the RI method, we calculated Inter-annotator agreement, and indeed *Observed agreement* and *Cohen's Kappa agreement* (see Section 4.2). The observed agreement across all keywords was 0.81, and the Cohen's Kappa was 0.61 which indicates that the given task of selecting relevant keywords for a topic of interest was indeed difficult for domain experts.

The code and datasets from the described experiment, including generated virtual documents and semantic spaces, can be downloaded from the LarKC Wiki[13].

Performance The parameter values affect not only the quality of results but also the required resources and the indexing time. Increasing the value of dimensionality and seed length almost exponentially increases the time to generate the semantic space (from 0.67 minutes for 500 dimensions to 3 minutes for 2500, LLD1; from 3.78 minutes for 500 to 11.5 minutes for 2500 dimensions, LLD2). The higher the value for *seed length* and *dimensionality*, the higher the requirements for the computational resources and RAM in particular[14]. Application of RI to the whole LLD dataset poses the scalability issues related to the size of our corpus. While indexing is a one-off operation (that takes ˜16 hours on MDC computer with 256G RAM), the search for 'lung' after the space is generated takes 14 minutes. Therefore, in our related work reported elsewhere ([1]) we looked at the parallelisation of the RI search algorithm in order to make exploring large RDF graphs using the contextual similarities of the comprising nodes applicable in real time applications.

5   Conclusion and future work

We described the application of the Random Indexing method for the task of searching large and unknown RDF graphs. We tested our method on the subsets of the Linked Life Data, by training it using the variation of parameters, and then involving domain experts to judge on the relevance of retrieved terms. None of the parameters had a significant influence on MAP, apart from the size of the dataset. However, the values of MAP reaching 0.565 and 0.61 for LLD1, and LLD2 datasets respectively, indicate that the generation of virtual documents as described in this paper and generating the semantic index using the RI method has promising results. The reason for the stability of the RI method might have been the span of the parameters which we used, and hence in our future work we will expand the variation span and also repeat the runs across the same parameter variations in order to increase the significance of results.

References

1. Assel, M., Cheptsov, A., Czink, B., Damljanovic, D., Quesada, J.: MPI Realization of High Performance Search for Querying Large RDF Graphs using Statistical

---

[13] `http://wiki.larkc.eu/LarkcProject/statisticalSemantics`

[14] The experiments are conducted on the MDC super-computer: 2 IBM x3950M2, 32 Cores (4 quad core Intel Xeon@2.93GHz per node), 256 Gbytes of main memory, production cluster for Java software and serial code.

[15] `http://code.google.com/p/semanticvectors/`

[16] `http://www.larkc.eu/`

Semantics . In: Proceedings of the 1st Workshop on High-Performance Computing for the Semantic Web, Collocated with the 8th Extended Semantic Web Conference (ESWC 2011). Heraklion, Greece (June 2011)

2. Broekstra, J., Kampman, A.: Serql: A second generation rdf query language. In: In Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval. pp. 13–14 (2003)

3. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: Proceedings of WWW2008. pp. 1101–1102 (2008)

4. Cohen, T., Schvaneveldt, R., Widdows, D.: Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. Journal of Biomedical Informatics (2009)

5. Cohen, T.: Exploring medline space with random indexing and pathfinder networks. AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium pp. 126–130 (2008)

6. Croft, B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice. Addison Wesley, 1 edn. (February 2009)

7. Damljanovic, D., Petrak, J., Cunningham, H.: Random Indexing for Searching Large RDF Graphs. In: Poster Session at the Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010). Lecture Notes in Computer Science, Springer-Verlag, Heraklion, Greece (June 2010)

8. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407 (1990)

9. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Proceedings of the 13th ACM international conference on Information and knowledge management. pp. 652–659. ACM, New York, NY, USA (2004)

10. Eugenio, B.D., Glass, M.: The kappa statistic: a second look. Computational Linguistics 1(30) (2004), (squib)

11. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006). Athens, GA, USA (2006)

12. Hripcsak, G., Heitjan, D.: Measuring agreement in medical informatics reliability studies. Journal of Biomedical Informatics 35, 99–110 (2002)

13. Johnson, W.B., Lindenstrauss, J.: Extensions to lipschiz mapping into hilbert space. Contemporary Mathematics 26 (1984)

14. Karlgren, J., Sahlgren, M.: From words to understanding. In: Uesaka, Y., Kanerva, P., Asoh, H. (eds.) Foundations of Real-World Intelligence, pp. 294–308. Stanford: CSLI Publications (2001)

15. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation - 15 January 2008, W3C (2008)

16. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In: Proceedings of WWW2006. pp. 23–31 (2006)

17. Sahlgren, M.: An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005. Citeseer (2005)

18. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Proceedings of the 6th International Semantic Web Conference. Busan, Korea (2007)