# Applying Random Indexing to Structured Data to Find Contextually Similar Words

**Danica Damljanović[*], Udo Kruschwitz[†], M-Dyaa Albakour[†], Johann Petrak[*2], Mihai Lupu[†2]**

[*]Department of Computer Science, University of Sheffield, United Kingdom, d.damljanovic@dcs.shef.ac.uk

[†]School of Computer Science and Electronic Engineering, University of Essex, United Kingdom, udo,malbak@essex.ac.uk

[*2] Austrian Research Institute for Artificial Intelligence, Vienna, Austria, johann.petrak@ofai.at

[†2] Vienna University of Technology, Vienna, Austria, lupu@ifs.tuwien.ac.at

**Abstract**

Language resources extracted from structured data (e.g. Linked Open Data) have already been used in various scenarios to improve conventional Natural Language Processing techniques. The meanings of words and the relations between them are made more explicit in RDF graphs, in comparison to human-readable text, and hence have a great potential to improve legacy applications. In this paper, we describe an approach that can be used to extend or clarify the semantic meaning of a word by constructing a list of contextually related terms. Our approach is based on exploiting the structure inherent in an RDF graph and then applying the methods from statistical semantics, and in particular, Random Indexing, in order to discover contextually related terms. We evaluate our approach in the domain of life science using the dataset generated with the help of domain experts from a large pharmaceutical company (AstraZeneca). They were involved in two phases: firstly, to generate a set of keywords of interest to them, and secondly to judge the set of generated contextually similar words for each keyword of interest. We compare our proposed approach, exploiting the semantic graph, with the same method applied on the human readable text extracted from the graph.

**Keywords:** rdf, ontologies, synonyms, contextually related words, random indexing

## 1. Introduction

Language resources extracted from structured data (e.g. Linked Open Data cloud[1]) have already been used in various scenarios to improve conventional Natural Language Processing (NLP) techniques. For example, in the Question-answering system PowerAqua (Lopez et al., 2006), the OWL:SAMEAS relation is used to find synonyms in addition to those found using conventional methods such as through WordNet (Fellbaum, 1998). The meanings of words and the relations between them are made more explicit in semantically structured knowledge sources such as RDF graphs, in comparison to human-readable text, and hence have a great potential to improve legacy applications. Statistical semantics methods such as Latent Semantic Analysis (LSA) and Random Indexing (RI) can be applied to derive the indirect relations between words. Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is one of the pioneer methods which has been used for finding words based on contextual similarity (e.g. 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 (Cohen et al., 2009). The 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 (Cohen et al., 2009)). RI can be seen as an approximation to LSA which is shown to be able to reach similar results (see (Karlgren and Sahlgren, 2001) and (Cohen and Hunter, 2008)).

In this paper, we describe an approach that can be used to extend or clarify the semantic meaning of a word through a list of contextually related terms. Our approach is based on exploiting the structure inherent in an RDF graph and then applying the statistical semantics methods. In particular, we use Random Indexing to discover contextually related terms. We evaluate our approach in the domain of life sciences using the dataset previously generated with domain experts from a large pharmaceutical company (AstraZeneca) (Damljanović et al., 2011b). The experts were involved in two phases: firstly, they generated a set of keywords of interest to them, and secondly, they judged the set of generated contextually similar words for each of their chosen keywords. The generated dataset is available through the LREC Map program. This paper is therefore an extension of our previous work with the main difference that our focus here is in finding contextually related words (not URIs). We also changed the way we set up the experiment, in order to get a better understanding of the stability of the method. In that sense, we widened the variation span for parameter values to cover a larger spectrum. In addition to the previous presentation, the current article makes a comparison with the same method applied to the human

[1]`http://linkeddata.org`

readable text. This latter approach is practically the conventional way of finding contextually related terms using distributional semantics methods.

## 2. Related Work

Researchers in NLP have shared a long standing interest in constructing domain models or semantic networks to associate terms to each other, e.g. (Widdows, 2004). It is in fact an area that goes way beyond NLP (Clark et al., 2011).

A number of methods have been proposed to extract term relationships that do not need to follow fully specified semantic relations. For example, in (Phillips, 1985), the author used the study of co-occurrence to build what is called *conceptual structures*, and *syntagmatic lexical networks* from science books. (Grefenstette, 1992) extended methods based on lexical patterns by quantifying the similarity of syntactic dependencies (e.g. modifiers) associated to a word, to cluster similar words together.

In contrast to the clusters of the previous methods which did not attempt to label relationships between terms or concepts, Sanderson and Croft introduced a hierarchical relationship, imposing a subsumption relation between concepts extracted from top matching documents retrieved for a given query (Sanderson and Croft, 1999).

Supervised with phrase patterns, co-occurrence analysis can also be used to build a network of hyponyms from text (Hearst, 1992).

Others, e.g. (Thelen and Riloff, 2002), (Snow et al., 2005), have taken this further, using entities already identified as being in a semantic class, or taking pairs of entities identified within WordNet (Fellbaum, 1998) as being in a hypernym/hyponym relation, as seeds to identify new phrasal patterns and entities belonging to that category or relationship.

Another stream of methods that goes under this category is the extraction of arbitrary relations from text between named entities in the form of subject-predicate-object triplets. For example, the REXTOR system in (Katz and Lin, 2000) used a finite state language model to extract what they call ternary expressions that describe relations between entities.

The main difference between our work and the previous work presented above is that we do not attempt to semantify large corpora of text to extract contextually related terms. We rather start from structured data in the form of large RDF graphs and build on the success of statistical semantic approaches, which were applied on text to enrich domain knowledge e.g. (Chen et al., 2008), to further enrich those knowledge structures.

Starting from semantically structured knowledge sources, there has been a lot of research in exploiting RDF structures in different applications, especially to assist users in searching and browsing RDF graphs. Examples include semantic search engines such as Swoogle (Ding et al., 2004) or Sindice (Tummarello et al., 2007). They collect the Semantic Web resources from the Web and then index the keywords and URIs against the RDF files containing those keywords and URIs, using an inverted index scheme. These search engines use traditional weighting mechanisms such as TF-IDF. In (Hogan et al., 2006) the authors introduce the *ReConRank* algorithm, which adapts the well-known PageRank algorithm to the Semantic Web data. The 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 (Qu et al., 2006), 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 (Cheng et al., 2008). 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.

## 3. Method

Although LSA has been shown to work effectively, the main problem of the method is its scalability: it starts by generating a $term \times document$ matrix which grows with the number of terms and the number of documents. In the case of a large corpus it will thus become very large. 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 ($O(mn^2)$, where $m, n$ are the numbers of terms and documents, with $m > n$). Also, calculating the LSA model is not easily and efficiently doable in an incremental or out-of memory fashion. Random Indexing (RI) (Sahlgren, 2005) circumvents these problems by avoiding the need of matrix factorization in the first place. RI can be incrementally updated and the $term \times document$ 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 $term \times document$ matrix and then reducing the dimensionality, RI starts by creating almost orthogonal *random vectors* (*index vectors*) of a fixed, reduced dimensionality, for each document. This random vector is created by setting a number of randomly selected dimensions to either +1 or -1. Each term is then represented by a vector (term vector), which is a combination of all index vectors of the documents in which the term 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.

In order to apply RI to an RDF graph we first generate the set of documents representing this graph, by generating one *virtual document* (Damljanović et al., 2011a) for each node (i.e. URI) in the graph. We then generate a semantic index from these virtual documents, which we use to retrieve similar literals or URIs for a given term or set of terms. As URIs are not useful for domain experts, they are replaced using the labels that describe them.

### 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 the 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 essentially traverse the graph using Breadth First Search starting from the representative node and going to depth $N$.
In addition, we include or exclude certain parts of the TBox. For example, direct classes for instances are excluded ($P_N != rdf : type$), while other annotation properties such as $rdfs : label$ are included.
An example is shown in Figure 1, for $N = 1$ (i.e. a representative subgraph of order 1). *Virtual documents* are generated from all paths in the representative subgraph where:

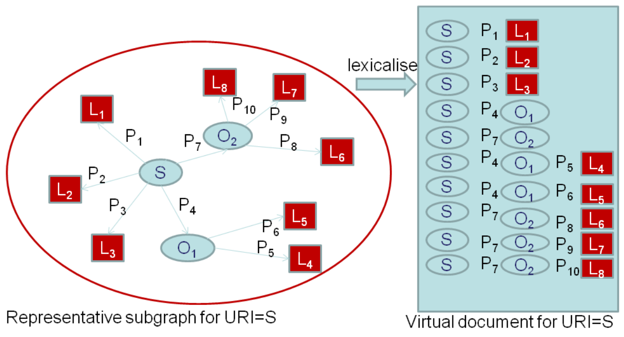

Figure 1: From a representative subgraph to the virtual document

- all URIs 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. Preprocessing literals is very important and depends on the domain represented by the RDF graph. Removing gene and protein sequences is important in the domain of life sciences represented by Linked Life Data, while different rules would apply for the general knowledge represented by e.g. `www.factforge.net`, or any other graph from the Linked Open Data cloud.

As a consequence of the building method just described, the terms in these virtual documents are either URIs or literals. To illustrate this with an example, a representative subgraph for $URI = geo : colorado, N = 1$ would look as follows:

```
geo:colorado rdfs:label colorado
geo:colorado geo:abbreviation co
geo:colorado geo:statePopulation 104000
geo:colorado geo:borders geo:utah
geo:utah rdfs:label utah
geo:utah geo:abbreviation ut
geo:utah geo:statePopulation 84900
```

The virtual document generated from this subgraph is the following:

```
geo:colorado rdfs:label colorado
geo:colorado geo:abbreviation co
geo:colorado geo:statePopulation 104000
geo:colorado geo:borders geo:utah
geo:colorado geo:borders geo:utah
             rdfs:label utah
geo:colorado geo:borders geo:utah
             geo:abbreviation ut
geo:colorado geo:borders geo:utah
             geo:statePopulation 84900
```

The rationale behind including the full paths starting with the *representative URI* and not only the neighbouring nodes is the frequency and the way the TF-IDF works. We want the virtual document to represent the URI in question (i.e. *geo:colorado* for the example above) and not its neighbouring nodes. If we included only the neighbouring nodes, *geo:utah* would appear many more times than *geo:colorado* in the virtual document representing *geo:colorado*.
The next step is generating the semantic index. Once it has been created, it can be used to find similarities between URIs and literals. 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.
There are several parameters which can influence the process of generating the semantic index, or vectors using RI:

- **Seed length** Number of +1 and -1 entries in a sparse random vector.
- **Dimensionality** Dimension of the semantic vector space, i.e. the 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.

In what follows we describe the experiments that study how variations of these parameters influence the quality of the results and how sensitive the method is to that variation. In our previous work (Damljanović et al., 2011b) we conducted a similar experiment and found that the variation of the parameters did not influence the quality of results and that the method is quite stable to that variation. However, we now expand the variation span in order to further investigate the issue.

## 4. Experiments

In this section we describe the experiment with RI. Due to the stability problem of RI (see (Sitbon and Bruza, 2008)), and to increase the significance of the results, we repeat the experiments for each combination of parameters six times. We begin by describing in more detail the dataset used.

## 4.1. Dataset

Linked Life Data (`www.linkedlifedata.com`) is a dataset covering the life sciences domain. The latest version has more than 5 billion statements in total. We generated 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*. The predefined limit of statements is controlled by the number of representative subgraphs in the extracted datasets and was set to 5000 and 50000 for LLD1 and LLD2 respectively. Thus the main difference between the two datasets is the size, but it should be noted that they both start with the same set of seed URIs and hence they contain equal number of abstracts (one for each seed URI).
In addition, we extract only abstracts for these articles and we use this dataset for our baseline model. We refer to this dataset as *Pubmed abstracts*. The seed SPARQL query which is used for this task is the following:

```
PREFIX pubmed:
<http://linkedlifedata.com/resource/pubmed/>
SELECT DISTINCT ?x
WHERE {
  ?x a pubmed:Citation.
  ?x pubmed:year "2009" .
  ?x pubmed:month "12" .
}
```

The query was evaluated against the LLD SPARQL endpoint[2].
Table 1 shows the sizes of LLD1 and LLD2.
The full dataset, including the terms used for training and testing, as well the manually created relevance judgements are available online at `https://sites.google.com/site/thelarkclifesciencesdataset/`.

## 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 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 (Croft et al., 2009). 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 on the top 10 retrieved elements. Recall is extremely difficult to measure due to the number of terms in our datasets (see Table 1). However, due to the nature of our task the users care most about the top ranked results, which is exactly what is captured by MAP.

[2]`www.linkedlifedata.com/sparql`

## 4.3. Experimental Setup

The experimental setup is as follows:

1. **Extracting topics of interest**. Here we use 23 queries generated by domain experts from the pharmaceutical company AstraZeneca. We split this set into two halves as shown in Table 2.

| Group 1: training | Group 2: testing |
|---|---|
| 5-HT receptors | acetylcholinesterase |
| Adverse events | antagonist |
| bladder cancer | antioxidant |
| cholinergic signaling | cognitive |
| clinical trial | cystectomy |
| Exposure | efficacy |
| non-human primates | lung |
| Posttraumatic Stress Disorder | lung cancer |
| PTSD | trauma |
| synergistic effect | migraine |
| trial | Raynaud disease |
|  | magnesium |

Table 2: Topics of interest divided into two groups for training and testing

2. **Training the model**: we generated the RI models for several variations of the following RI parameters for three datasets (LLD1, LLD2 and Pubmed abstracts):
   - Vector dimension: 50, 100, 150, 300, 500, 1000, 2000, 5000
   - Seed length: 4, 10
   - Term frequency: 1, 5, 10, 15, 25

   As each of these settings were repeated 6 times, this resulted in 960 runs (480 per dataset). The parameters combinations that lead to the best results (measured through MAP) were considered as the best setting for testing the method in the next step.

3. **Testing the model**: for the models generated using the parameters retrieved in the previous step, we retrieved 10 similar words for each topic 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 (Damljanović et al., 2011b) for more details).

|  | LLD1 | LLD2 |
|---|---|---|
| number of representative subgraphs | 5000 | 50000 |
| number of statements | 595798 | 4573668 |
| number of virtual documents | 64644 | 473742 |
| number of terms | 417753 | 1713349 |

Table 1: Sizes of LLD1 and LLD2 datasets

|  | Dataset | Mean | Std. Deviation |
|---|---|---|---|
| **Group 1** | LLD1 | 0.54 | 0.28 |
|  | LLD2 | 0.46 | 0.31 |
|  | Pubmed abstracts | 0.33 | 0.28 |

Table 3: The dispersion values for the distribution of MAP across three datasets for the training model

## 5. Results

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

### 5.1. 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 future applications of the method.
As we can see in Table 3, results were better with LLD1 in comparison to LLD2, and this difference is statistically significant ($p < 0.0001$, Independent-samples Mann-Whitney U Test). The reason is the high frequency of 0 values for the LLD2 dataset. However, results with LLD2 are still better in comparison to those with the baseline.
Looking closely into the differences of Average Precision per keyword for LLD1 and LLD2 datasets, the value for MAP seems to be constantly better for LLD1 in comparison to LLD2, with the term *PTSD* being the only exception (see Figure 2).

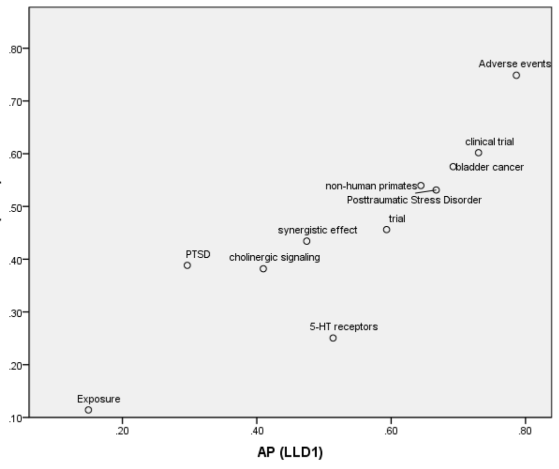

Figure 2: Correlation of Average Precision (AP) for LLD1 (X axis) and LLD2 (Y axis) per keyword

The seed length parameter did not have any significant influence (p=0.714 for LLD1 and p=0.914 for LLD2, Independent-samples Mann-Whitney U Test), see Figure 3. We see this as a positive result, given that the computational resources (RAM in particular) are proportional to the value of seed length.
However, the wider span of dimensionality parameter made a significant difference to the value of MAP for LLD1 and LLD2 datasets, see Figure 4 ($p < 0.0001$, Independent-samples Kruskal-Wallis test). The peak for LLD1 was the dimensionality 500, while surprisingly the peak for the larger dataset was quite low, with the dimensionality set to 150. The variation of dimensionality parameter did not have a significant difference on MAP for the baseline model.
Similarly, the variation of *minimum term frequency* caused the fluctuation of the results, see Figure 5, with peaks for both LLD1 and LLD2 datasets at the maximum value of this parameter which was 25. For the baseline model, the peak was much lower (5). The difference in MAP caused by the variation of *minimum term frequency* for all datasets is statistically significant ($p < 0.0001$, Independent-samples Kruskal-Wallis test).

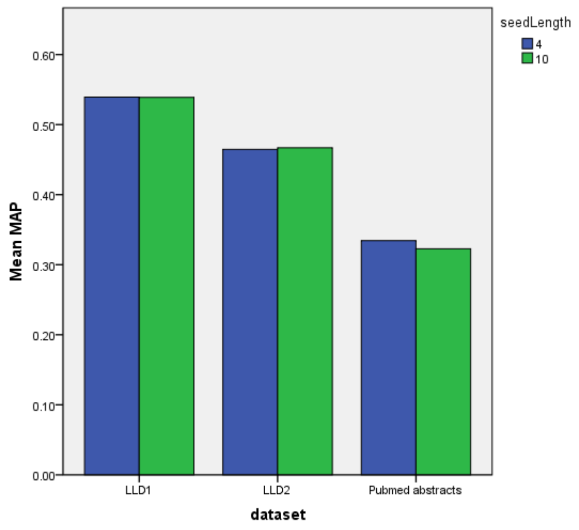

Figure 3: The effect of the variation of seed length on *MAP*, for *Group 1* used as the training set

Table 4 summarizes the best parameters: those that we chose to use in the testing phase.

### 5.2. Testing the model

Using the best parameters selected in the training phase, we now test whether the same setting can be used for a different, future set of terms.
We ran the search method using *Group 2* as a *testing* set against the RI model trained with *Group 1*. Results are

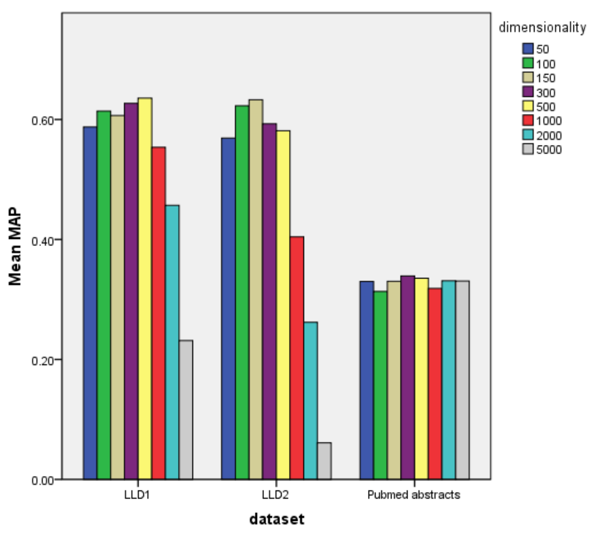

Figure 4: The effect of the variation of dimensionality on *MAP*, for the Group 1 used as the training dataset

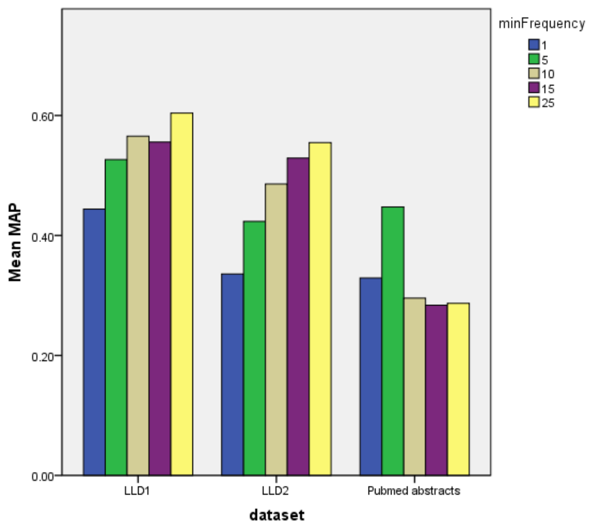

Figure 5: The effect of the variation of minimal term frequency on *MAP*, for the Group 1 used as the training dataset

shown in Table 5. The RI method results in a smaller value of MAP in the testing phase in comparison to the best trained model for all datasets.

Figure 6 shows the values of average precision per keyword, divided by dataset. The results vary across different keywords and all datasets seem to have peaks for some of the keywords. However, only the baseline resulted in producing 0 for as much as 25% of the observed keywords, unlike the two other datasets which always yielded positive average precision. The baseline yielded best results for 2 out of 12 keywords (16.67%), while LLD1 yielded better results for 4/12 keywords (33.33% of the cases). LLD2 produced best results in majority of the cases, namely for 6 out of 12 keywords (50% of the cases).

| | **Group 1** | | |
|---|---|---|---|
| **Dataset** | LLD1 | LLD2 | Pubmed abstracts |
| **Min frequency** | 25 | 25 | 5 |
| **Seed length** | 4 | 4 | 4 |
| **Dimensionality** | 500 | 150 | 300 |
| **MAP** | 0.60 | 0.60 | 0.47 |

Table 4: Optimal parameters chosen for *Group 1* used as training sets

| | **Group 2** | | |
|---|---|---|---|
| **Dataset** | LLD1 | LLD2 | Pubmed abstracts |
| **Min frequency** | 25 | 25 | 5 |
| **Seed length** | 4 | 4 | 4 |
| **Dimensionality** | 500 | 150 | 300 |
| **MAP** | 0.425 | 0.48 | 0.26 |

Table 5: Testing the Random Indexing method using *Group 2* as the *testing* set

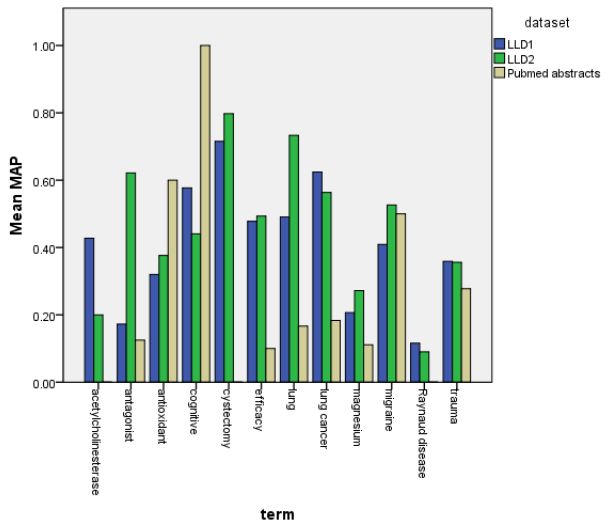

Figure 6: Mean Average Precision per keyword, by each dataset

**Human assessment**. In our previous work (Damljanović et al., 2011b), we reported the Inter-annotator agreement using *Observed agreement* and *Cohen's Kappa agreement*. 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. Relative to this difficulty, we can conclude that our proposed method reaching the average MAP of 0.45 across LLD1 and LLD2 datasets is a very promising starting point to explore this approach further. Although this can be further improved, it is still much higher in comparison to the baseline model which applies the same method on the text of the abstracts instead of exploring the RDF structure.

## 6. Conclusion

We described an approach that can be used to extend or clarify the semantic meaning of a word through suggesting a list of contextually related terms. Our approach is based on exploiting the structure inherent in an RDF graph and then applying statistical semantics methods. In particular, we used Random Indexing, in order to discover contextually related terms. We evaluated our approach in the domain of life sciences on the datasets generated with domain experts from a large pharmaceutical company (AstraZeneca). The domain experts first selected a set of keywords of interest to them, and then judged the set of the terms that our method found to be most similar to the given terms, based on contextual information. The results show that the RI method on our data is reasonably stable with MAP reaching on average 43-60%. With larger number of documents in the corpus, the larger minimum term frequency parameter should be used, as it can filter out the noise present in the data. To some extent surprisingly, we found that for the larger dataset, a larger number of dimensions decreased the performance. The seed length parameter was not found to have any significant influence and it is reasonable to use the minimum 4. We compared the results of the method as applied to the graph and as applied to the human readable text serving as our baseline, and found that the method that exploits the graph structure yields approximately 16-22% higher results in terms of Mean Average Precision.

## Acknowledgements

We would like to thank creators of the SemanticVectors[3] library which is used in the experiments reported in this paper. This research was supported by the EU-funded LarKC[4] (FP7-215535) project. Mihai Lupu was partially supported by the PROMISE NoE (FP7-258191).

## 7. References

R. Chen, I. Lee, Y. Lee, and Y. Lo. 2008. Upgrading domain ontology based on latent semantic analysis and group center similarity calculation. In *Proceedings of the 2008 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2008)*, pages 1495–1500. IEEE.

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

M. Clark, Y. Kim, U. Kruschwitz, D. Song, M-D. Albakour, S. Dignum, U. Cervino Beresi, M. Fasli, and A. De Roeck. 2011. Automatically Structuring Domain Knowledge from Text: an Overview of Current Research. *Information Processing and Management*. In press.

K. B. Cohen and L. Hunter. 2008. Getting started in text mining. *PLoS Comput Biol*, 4(1):e20, 01.

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

B. Croft, D. Metzler, and T. Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison Wesley.

D. Damljanović, U. Kruschwitz, and M-D. Albakour. 2011a. Using Virtual Documents to Move Information Retrieval and Knowledge Management Closer Together. In *ESAIR11*, Glasgow, Scotland, UK, October.

D. Damljanović, J. Petrak, M. Lupu, H. Cunningham, M. Carlsson, G. Engstrom, and B. Andersson. 2011b. Random Indexing for Finding Similar Nodes within Large RDF graphs. In Raúl García-Castro, Dieter Fensel, and Grigoris Antoniou, editors, *The Semantic Web: ESWC 2011 Workshops, Workshops at the 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Greece, May 29-30, 2011, Revised Selected Papers*, volume 7117. Springer.

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

L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. 2004. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA. ACM.

C. Fellbaum, editor. 1998. *WordNet - An Electronic Lexical Database*. MIT Press.

G. Grefenstette. 1992. Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of the 33rd international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1992)*, pages 89–97. ACM.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING 1992)*, volume 2, pages 539 – 545. ACL.

A. Hogan, A. Harth, and S. Decker. 2006. 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.

J. Karlgren and M. Sahlgren. 2001. From words to understanding. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 294–308. Stanford: CSLI Publications.

B. Katz and J. Lin. 2000. REXTOR: a system for generating relations from natural language. In *Proceedings of the ACL Workshop on Recent Advances in Natural Language Processing and Information Retrieval*, pages 67–77. ACL.

V. Lopez, E. Motta, and V. S. Uren. 2006. Poweraqua: Fishing the semantic web. In *ESWC*, pages 393–410.

M. Phillips. 1985. *Aspects of Text Structure: an investigation of the lexical organization of text*, volume 52 of *North-Holland Linguistic Series*. Elsevier.

Y. Qu, W. Hu, and G. Cheng. 2006. Constructing virtual documents for ontology matching. In *Proceedings of WWW2006*, pages 23–31.

M. Sahlgren. 2005. 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.

M. Sanderson and B. Croft. 1999. Deriving concept hierar-

[3]www.code.google.com/p/semanticvectors/
[4]www.larkc.eu

chies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pages 206–213. ACM.

L. Sitbon and P. Bruza. 2008. On the relevance of documents for semantic representation. In *Proceedings of the 13th Australasian Document Computing Symphosium*, Hobard, Australia, December.

R. Snow, D. Jurafsky, and A. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems*, 17:1297–1304.

M. Thelen and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, volume 10, pages 214–221. ACL.

G. Tummarello, R. Delbru, and E. Oren. 2007. Sindice.com: Weaving the Open Linked Data. In *Proceedings of the 6th International Semantic Web Conference*, Busan, Korea.

D. Widdows. 2004. *Geometry and Meaning*. CSLI Lecture Notes.