



LarKC

The Large Knowledge Collider

a platform for large scale integrated reasoning and Web-search

FP7 – 215535

D2.7.3

Conclusions from Experimental Data and Combinatorics Analysis

Coordinator: Hansjörg Neth (MPG)

With contributions from:

**Hansjörg Neth, Jose Quesada, Lael J. Schooler (MPG),
Arjon Buikstra, Annette ten Teije, Frank van Harmelen (VUA),**

Danica Damljanovic (USFD), Ivan Peikov (Onto),

Yi Zeng, Xu Ren, Yan Wang (WICI)

Quality Assessor: Iker Larizgoitia (STI)

Quality Controller: Danica Damljanovic (USFD)

Document Identifier:	LarKC/2008/D2.7.3
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0.0
Date:	September 30, 2011
State:	Final
Distribution:	Public

Executive Summary

This document (LarKC D2.7.3) provides an overview over ongoing and completed efforts to evaluate data retrieval and selection methods within LarKC. As many of these efforts have been reported in previous project deliverables (e.g., D2.1.1, D2.1.2, D2.3.2, D2.3.3 and D2.7.2) we will refer to those documents whenever appropriate and only repeat information when it is relevant for the understanding of the present document. Its main content chapters report the efforts of various LarKC consortium partners towards evaluating data selection methods:

Chapter 3 reports data selection experiments conducted by USFD and Ontotext. A basic question that has already been addressed in previous deliverables (e.g., LarKC D2.7.2 and D2.3.3, Neth et al., 2011; Quesada et al., 2011) is whether subsetting can reduce the time for SPARQL query evaluation, by first retrieving the relevant subset to a query and then reasoning over that subset, as opposed to reasoning over all statements. The research reported in this chapter compares three different selection methods (baseline, Random Indexing and Spreading Activation) and measures their efficiency (execution time) and effectiveness (recall).

Chapter 4 refines WICI's research on using user-interests to provide recommendations on personalized semantic literature searches and an active academic visit recommendation system (see also LarKC D2.3.1 and D2.3.2, Quesada et al., 2009, 2010). In a quantitative evaluation of interests ranking and interests-based selection three approaches to answer a specific query (standard, interests-based query refinement, and interests-based selection) are compared. Interests-based selection is found to provide the most relevant results as quickly as possible.

Chapter 5 reports the results of the experiments conducted by VUA and MPG to develop and evaluate a heuristic method for data selection on the basis of structural similarity. Faced with the problem of how to select the correct answers to a query from partially incorrect answers the proposed method uses a cognitively inspired similarity measure to filter the correct answers from the full set of answers. As dissimilar results are removed a reduced recall corresponds to an increase in precision that mimics the benchmark of a human rater. Although the resulting trade-off curve does not outperform a human rater it is remarkable how well the heuristic performs in the absence of any sophisticated background knowledge.

Document Information

IST Project Number	FP7 – 215535	Acronym	LarKC
Full Title	Large Knowledge Collider		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	2.7.3	Title	Conclusions from Experimental Data and Combinatorics Analysis
Work Package	Number	2	Title	Selection and Retrieval

Date of Delivery	Contractual	M42	Actual	31-September-2011
Status	version 1.0.0		final	<input checked="" type="checkbox"/>
Nature	prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/>			
Dissemination Level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

Authors (Partner)	Hansjörg Neth, Jose Quesada, Lael J. Schooler (MPG), Arjon Buikstra, Annette ten Teije, Frank van Harmelen (VUA), Danica Damljanovic (USFD), Ivan Peikov (Onto), Yi Zeng, Xu Ren, Yan Wang (WICI).		
Resp. Author	Hansjörg Neth (MPG)		E-mail neth@mpib-berlin.mpg.de
	Partner	MPG	Phone +49 (30) 82406-696

Abstract (for dissemination)	<p>This document (LarKC D2.7.3) provides an overview over ongoing and completed efforts to evaluate data retrieval and selection methods within LarKC. As many of these efforts have been reported in previous project deliverables (e.g., D2.1.1, D2.1.2, D2.3.2, D2.3.3 and D2.7.2) we will refer to those documents whenever appropriate and only repeat information when it is relevant for the understanding of the present document. Its main content chapters report the efforts of various LarKC consortium partners towards evaluating data selection methods:</p> <p>Chapter 3 reports data selection experiments conducted by USFD and Ontotext. A basic question that has already been addressed in previous deliverables (e.g., LarKC D2.7.2 and D2.3.3, Neth et al., 2011; Quesada et al., 2011) is whether subsetting can reduce the time for SPARQL query evaluation, by first retrieving the relevant subset to a query and then reasoning over that subset, as opposed to reasoning over all statements. The research reported in this chapter compares three different selection methods (baseline, Random Indexing and Spreading Activation) and measures their efficiency (execution time) and effectiveness (recall). Chapter 4 refines WICI's research on using user-interests to provide recommendations on personalized semantic literature searches and an active academic visit recommendation system (see also LarKC D2.3.1 and D2.3.2, Quesada et al., 2009, 2010). In a quantitative evaluation of interests ranking and interests-based selection three approaches to answer a specific query (standard, interests-based query refinement, and interests-based selection) are compared. Interests-based selection is found to provide the most relevant results as quickly as possible.</p> <p>Chapter 5 reports the results of the experiments conducted by VUA and MPG to develop and evaluate a heuristic method for data selection on the basis of structural similarity. Faced with the problem of how to select the correct answers to a query from partially incorrect answers the proposed method uses a cognitively inspired similarity measure to filter the correct answers from the full set of answers. As dissimilar results are removed a reduced recall corresponds to an increase in precision that mimics the benchmark of a human rater. Although the resulting trade-off curve does not outperform a human rater it is remarkable how well the heuristic performs in the absence of any sophisticated background knowledge.</p>
Keywords	retrieval and selection, precision/recall, data quality, benchmarks, user feedback, evaluation

Version Log			
Issue Date	Rev No.	Author	Change
2011 08 15	0.1	Hansjörg Neth	Basic structure and content.
2011 08 30	0.2	Hansjörg Neth	Including chapter on data model and metrics.
2011 09 02	0.3	Hansjörg Neth	Including WICI parts on interests-based selection.
2011 09 08	0.4	Hansjörg Neth	Including draft of selection experiment by USFD and Onto.
2011 09 12	0.5	Hansjörg Neth	Including MPG parts on selection heuristic.
2011 09 14	0.6	Hansjörg Neth	Preliminary version 0.8 submitted to quality assessment.
2011 09 15	0.7	Hansjörg Neth	Including new results by USFD (version 0.8.5).
2011 09 20	0.8	Hansjörg Neth	Revising based on quality assessor's comments.
2011 09 21	0.9	Hansjörg Neth	Working in revision from WICI (version 0.9.5).
2011 09 22	0.10	Hansjörg Neth	Working in revisions from Danica Damljanovic (version 1.0.0).

Project Consortium Information

Acronym	Partner	Contact
Semantic Technology Institute Innsbruck http://www.sti-innsbruck.at		Prof. Dr. Dieter Fensel Semantic Technology Institute (STI) Innsbruck, Austria Email: dieter.fensel@sti-innsbruck.at
AstraZeneca AB http://www.astrazeneca.com		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL SCRL. http://www.cefriel.it		Prof. Dr. Emanuele Della Valle CEFRIEL SCRL. Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O. http://cyceurope.com		Dr. Michael Witbrock CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universität Stuttgart http://www.hlrs.de		Matthias Assel Höchstleistungsrechenzentrum, Universität Stuttgart Stuttgart, Germany Email : assel@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FÖRDERUNG DER WISSENSCHAFTEN E.V. http://www.mpib-berlin.mpg.de		Dr. Lael Schooler, Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext Lab, Sirma Group Corp. http://www.ontotext.com		Atanas Kiryakov, Ontotext Lab, Sirma Group Corp. Sofia, Bulgaria Email: atanas.kiryakov@sirma.bg
SALT LUX INC. http://www.saltlux.com/EN/main.asp		Kono Kim SALT LUX INC Seoul, Korea E-mail: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT http://www.siemens.de		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT München, Germany Email: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD http://www.shef.ac.uk		Prof. Dr. Hamish Cunningham THE UNIVERSITY OF SHEFFIELD Sheffield, UK Email: h.cunningham@dcs.shef.ac.uk
VRIJE UNIVERSITEIT AMSTERDAM http://www.vu.nl		Prof. Dr. Frank van Harmelen VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY http://www.iwici.org		Prof. Dr. Ning Zhong THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan E-mail: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RESEARCH ON CANCER http://www.iarc.fr		Dr. Paul Brennan INTERNATIONAL AGENCY FOR RESEARCH ON CANCER Lyon, France Email: brennan@iarc.fr




INFORMATION RETRIEVAL FACILITY http://www.ir-facility.org		Dr. John Tait INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA http://www.utcluj.ro		Prof. Dr. Eng. Sergiu Nedevschi TECHNICAL UNIVERSITY OF CLUJNAPOCA Cluj-Napoca, Romania Email: sergiu.nedevschi@cs.utcluj.ro
SOFTGRESS S.R.L. http://www.softgress.com		Dr. Ioan Toma SOFTGRESS S.R.L. Cluj-Napoca, Romania Email: ioan.toma@softgress.com

Table of Contents

1	INTRODUCTION	1
2	DATA MODEL AND EVALUATION METRICS	2
2.1	Data Model	2
2.2	Evaluation Metrics	3
2.3	Evaluating Data Selection and Retrieval in LarKC	5
3	EVALUATION OF DATA SELECTION METHODS	8
3.1	Introduction	8
3.1.1	Objective	8
3.1.2	Evaluation Setup	9
3.2	Results	10
3.2.1	Baseline and Random Indexing	10
3.2.2	Spreading Activation	18
3.2.3	Summary and Conclusion	20
4	THEORY AND EVALUATION OF INTERESTS-BASED SELECTION	22
4.1	A Theoretic Framework for Interests-based Selection	22
4.1.1	Interests-based Selection Based on Single Interest Type	22
4.1.2	Interests-based Selection Based on Multiple Interest Types	24
4.2	Scenarios and Usecases for Interests-based Selection	24
4.2.1	Personalized Semantic Literature Search	25
4.2.2	Active Academic Visit Recommendation	27
4.3	Evaluation for Interests-based Selection	30
4.4	Plug-in Status for Interests-based Selection	31
4.5	Lessons Learned Through Interests-based Selection	31
5	RANKING RETRIEVAL RESULTS BY SEMANTIC SIMILARITY	33
5.1	Introduction and Motivation	33
5.2	Background and Related Work	34
5.3	Benchmark Construction	36
5.4	Measuring Human Performance	36
5.5	A Selection Heuristic Based on Semantic Similarity	39
5.6	Heuristic Performance	40
5.7	Conclusion and Outlook	41
6	CONCLUSIONS	43
	REFERENCES	45
	References	45
A	QUERIES USED IN EVALUATION OF DATA SELECTION METHODS	49
A.1	MusicBrainz queries	49
A.2	Dbpedia queries	52



B	QUESTIONS AND QUERIES FOR SIMILARITY-BASED DATA SELECTION	56
B.1	General Knowledge Questions	56
B.2	SPARQL Queries	58
B.2.1	Namespaces	58
B.2.2	Queries	58
C	LIST OF ABBREVIATIONS	71

List of Figures

2.1	An illustration of the trade-off between precision and recall.	4
3.1	Recall and the execution time using the baseline method and virtual documents of depth 1 and 2. Time (% of original) is a relative measure showing how the execution time against the subset at the specific cutoff point compares to the execution time of the queries against the fully materialised repository which we call ‘original’, % of original = $(timeS/timeM) * 100$	11
3.2	Recall and the execution time using Random Indexing and virtual documents of depth 1 and 2.	12
3.3	The average number of statements in the subsets across all queries. For a comparison, the fully materialised repository contains around 19.2 million statements.	12
3.4	The average recall for the baseline and random indexing methods using the same cutoff points across all queries and depths 1 and 2.	13
3.5	Recall by query for the baseline method and depth 1.	14
3.6	Recall by query for the baseline method and depth 2.	14
3.7	Recall by query for the RI method and depth 1.	15
3.8	Recall by query for the RI method and depth 2.	15
3.9	Recall and the execution time for the DBpedia dataset, depth 1.	16
3.10	Recall and the execution time for the DBpedia dataset, depth=1, with TBox loaded in addition to the identified subsets.	18
4.1	Academic visit recommendation for “Frank van Harmelen” to the U.K.	29
5.1	Example of a SPARQL query and corresponding FactForge answer-set.	37
5.2	Performance of human rater: (a) on an example query and (b) accumulated over all queries.	38
5.3	Performance of the similarity heuristic: (a) on the example query and (b) accumulated results.	41

List of Tables

3.1	Results of the first SA selection evaluation experiment.	19
3.2	Results of the second SA selection evaluation experiment.	20
4.1	A comparative study of top- K research interests for “Olof Selroos” from various perspectives.	26
4.2	Sample results for querying with interests-based selection	27
4.3	Interests level definition for <i>AAVRA</i>	28
4.4	Different levels of interests for “Frank van Harmelen” in <i>AAVRA</i>	29
4.5	A comparative study of scalability on query time for three different strategies.	31

1 Introduction

The primary objective of LarKC WP2—entitled *Selection and Retrieval*— is to explore and develop method and tools for retrieving and selecting data in the context of the Large Knowledge Collider (LarKC), a platform for massive distributed incomplete reasoning that will remove the scalability barriers of currently existing reasoning systems for the Semantic Web (see <http://www.larkc.eu> and Fensel & Harmelen, 2007; Fensel et al., 2008). Within this context, the goal of this document (LarKC deliverable 2.7.3) is to provide an overview of and conclusions from the efforts to evaluate various data retrieval and selection methods. As some of these efforts have already been reported in previous project deliverables (see especially D2.1.1, D2.1.2, D2.3.2, D2.3.3 and D2.7.2, Cunningham et al., 2008, 2008; Quesada et al., 2010, 2011; Neth et al., 2011) we will refer to those documents whenever appropriate and only repeat information contained therein when it is relevant for the understanding of the present document. The main chapters of this document are structured as follows:

- Chapter 2 summarizes the data model and evaluation metrics of information retrieval in general and in the specific context of the LarKC project.
- Chapters 3, 4 and 5 report the efforts of various LarKC consortium partners towards evaluating data selection methods.

Chapter 3 reports data selection experiments conducted by USFD and Ontotext. A basic question that has already been addressed in previous deliverables (e.g., LarKC D2.7.2 and D2.3.3, Neth et al., 2011; Quesada et al., 2011) is whether subsetting can reduce the problem space for reasoning, by first retrieving the relevant subset to a query and then reasoning over that subset, as opposed to reasoning over all statements. The research reported in this chapter compares three different selection methods (baseline, Random Indexing and Spreading Activation) and measures their efficiency (execution time) and effectiveness (recall).

Chapter 4 refines WICI’s research on using user-interests to provide recommendations on personalized semantic literature searches and an active academic visit recommendation system (see also LarKC D2.3.1 and D2.3.2, Quesada et al., 2009, 2010). In a quantitative evaluation of interests ranking and interests-based selection three approaches to answer a specific query (standard, interests-based query refinement, and interests-based selection) are compared. Interests-based selection finds the most relevant results more quickly than the other two approaches.

Chapter 5 reports the results of the experiments conducted by VUA and MPG to develop and evaluate a heuristic method for data selection on the basis of structural similarity. Faced with the problem of how to select the correct answers to a query from partially incorrect answers the proposed method uses a cognitively inspired similarity measure to filter the correct answers from the full set of answers. As dissimilar results are removed a reduced recall corresponds to an increase in precision that mimics the benchmark of a human rater. Although the resulting trade-off curve does not outperform a human rater it is remarkable how well the heuristic performs in the absence of any sophisticated background knowledge.

- The final chapter (Chapter 6) summarizes and concludes this document.

2 Data Model and Evaluation Metrics

Any attempt to retrieve data that is considered to be relevant to answer a given search query faces questions like the following:

- *Availability and relevance*: Which data sources are available? Which (parts) of those are relevant? What is the origin and context of available data sources?
- *Benefits and costs*: What are the benefits or costs of selecting certain data sources, e.g., in terms of the amount of data retrieved and the time taken to retrieve it? Is there a trade-off between data quantity and quality?
- *Stopping and switching*: How much data should be retrieved? When should data selection be stopped and possible post-processing steps (e.g., materialization, reasoning) be started?
- *Data use and users*: Who is asking for data and for which purpose?

Providing sound answers to these questions is not only the purpose of LarKC’s WP2 (*Selection and Retrieval*) but an endeavour of the entire LarKC enterprise. Whereas other project parts are primarily concerned with providing valuable data repositories, the key objective of our current efforts is to assess to what extent we can successfully select data that is both relevant and useful to answer user queries. To this end, we will first briefly describe the specific data model used by the LarKC project (Section 2.1), before recapitulating basic evaluation metrics for data selection and retrieval (Section 2.2). We conclude this chapter with some thoughts on the specific conditions and requirements on the evaluation of data selection and retrieval methods in the context of the LarKC project (Section 2.3).

2.1 Data Model

An elementary precondition to select data requires it to be in a format that allows it to be retrieved. We briefly recapitulate the key elements of LarKC’s underlying data model (see deliverables 2.1.1 and 2.1.2. Cunningham et al., 2008, 2008, for a more comprehensive discussion of this topic).

- RDF (short for “Resource Description Framework” and defined in Klyne & Carroll, 2004) is the basic data format used by the Semantic Web community. RDF’s atomic data element is a triple of the format `<Subject, Predicate, Object>`, for instance `<John, loves, Mary>` or `<Germany, hasCapitalCity, Berlin>`.
- An RDF graph represents a set of RDF triples. Within the graph abstraction, each triple defines an edge that is directed from the subject node to the object node and is labeled by the predicate. The nodes of the RDF graph can consist of URIs (unified resource identifiers, e.g., a URL), blanks (auxiliary nodes), or a XML literals. However, predicates need to be URIs. For most purposes, literals are not allowed to be in the subject position, i.e., they cannot be the start of an edge in the graph. Intuitively, literals are used to describe resources identified by URIs, but it makes no sense to describe literals because they represent primitive data values.

- A **data set** consists of a graph that integrates multiple RDF graphs so that the graphs can still be distinguished and managed separately. A more technical definition of this term is provided by the specification of the SPARQL query language (Prud'hommeaux & Seaborne, 2008).

A data set can be represented as an RDF multi-graph, which in turn can be represented as set of quadruples $\langle S, P, O, G \rangle$, where the initial triple $\langle S, P, O \rangle$ represent an RDF statement and the final element contains the name of a named graph that designates its origin.

- A **triple set** is part of a data set, i.e., a named multi-graph represented as a subset of its quadruples and identified by a unique name. Named graphs and triple sets allow tracking the provenance of data, e.g., when several graphs from distinct data sources are merged or referenced. Similarly, triple sets allow tagging parts of data sets when selecting them for further processing.

2.2 Evaluation Metrics

The academic discipline of information retrieval (IR) defines several measures of performance that indicate the quantity and quality of data selection results. Measures that typically serve as dependent variables in such evaluation efforts include:

1. The **number** of items retrieved, where items will typically be expressed in terms of RDF triples, although some applications may specify the **amount of data** selected more appropriately in terms of KBytes, MBytes, or GBytes.
2. The **speed** of data retrieval is a fairly straightforward measurement, bearing in mind that it also depends on the type and amount of computational resources deployed. As measurements of speed are relevant for every component of a LarkC workflow they are not specific to (but relevant for) data selection methods.
3. The **accuracy** of a data retrieval process can be expressed in terms of its precision and recall (see Frakes & Baeza-Yates, 1992; Singhal, 2001; Croft, Metzler, & Strohman, 2009, for more extensive treatments):

- **Precision** (or *hit rate* in signal-detection terminology) is defined as the number of relevant documents retrieved (i.e., *hits*) divided by the total number of documents retrieved (*hits + false alarms*) and is typically calculated for a certain number of retrieved documents (e.g., precision10, precision20, etc.):

$$\text{Precision} = \frac{\text{Hits}}{\text{Hits} + \text{False alarms}}$$

- **Recall** is the number of relevant documents retrieved (*hits*) divided by the total number of existing relevant documents that should have been retrieved (*hits + misses*):

$$\text{Recall} = \frac{\text{Hits}}{\text{Hits} + \text{Misses}}$$

Note that we do not consider the signal detection-theoretic notion of *correct rejections* (or irrelevant documents that were not retrieved) here, as

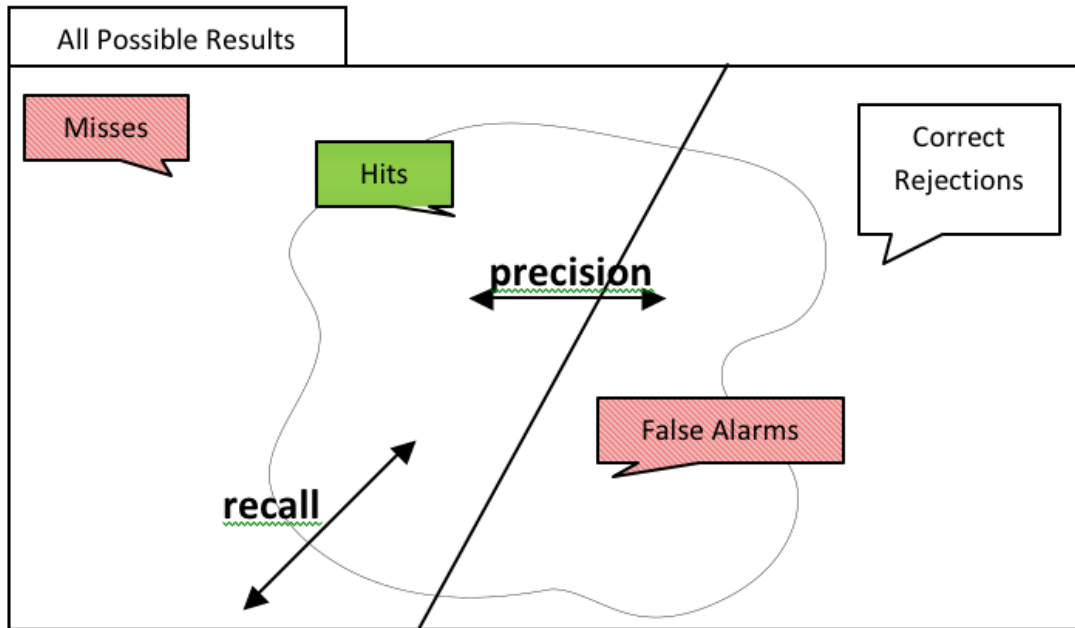


Figure 2.1: An illustration of the trade-off between precision and recall.

this would encompass knowledge of the complete set of possible answers contained in the data set, which—in a realistic semantic web setting—may include millions or even billions of irrelevant results.

Importantly, the accuracy of data retrieval is mostly characterized in terms of a trade-off between precision and recall. Figure 2.1 illustrates this trade-off. Increasing the amount of documents retrieved (denoted by the oval contour in Figure 2.1) to eventually encompass the entire possible data set will inevitably include all relevant documents (or hits). However, if the criterion for including relevant documents (denoted by the diagonal split in Figure 2.1, with relevant documents on its left) remains the same this increase in recall will simultaneously increase the number of irrelevant documents retrieved (false alarms), i.e., lower precision. By contrast, it would be easy to increase precision by only including a very small number of documents that are known to be relevant, but this will most likely lead to a large number of missed relevant documents, i.e., lower recall.

How the trade-off between precision and recall is best resolved will depend on the particular benefits and costs under the circumstances of a concrete task. For instance, if a clinician was to decide on a treatment of a patient’s flu-like symptoms by searching through a set of possible treatments she may prefer a very high rate of recall (as she would not want to miss a good treatment) and accept the comparatively small nuisance of false alarms (or lower precision). However, if the diagnosis resulted in surgery—as in the case of positive mammography or prostate cancer screenings—it would be advisable to lower recall to guard against false positives (and aim for higher precision), as those would cause considerable harm.

The two measures of precision and recall are often combined into a F_β -measure, where β modulates the relative importance of precision and recall within the measure, to balance the trade-offs between the two measures. The F_1 score

provides a simple integrative measure of the overall accuracy of a result and is a standard measure in information retrieval (see Goutte & Gaussier, 2005). It is calculated as the harmonic mean of precision and recall:¹

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Given this definition, the values of F_1 range between 0 and 1, where 0 signifies the minimum possible value and 1 signifies maximal precision and recall (i.e., perfect precision and recall ratios of 1).

4. The evaluation **quality** of data retrieval results is most difficult, as it typically depends on a combination of multiple measures. For many practical purposes, quality is equated with achieving a satisfying combination of speed and accuracy. More specifically, a quality result may be defined as achieving a satisfying solution on the precision-recall spectrum (or a high F -measure), but more general measures (like the benefits and costs of information retrieval) should also be taken into account.

2.3 Evaluating Data Selection and Retrieval in LarKC

This section reiterates and develops some thoughts on the specific conditions and requirements on the evaluation of data selection and retrieval methods in the context of the LarKC project. These thoughts have been initiated early in the project—e.g., in LarKC deliverables D2.1.1 and D2.1.2 (Cunningham et al., 2008, 2008)—and further refined throughout its maturation (e.g., in D2.7.2, Neth et al., 2011). Several points are relevant when considering the application of standard IR evaluation methodologies to the LarKC project:

- *LarKC is not static*: The LarKC platform consists of multiple components (i.e., any selector is only one part of a workflow that potentially includes identifiers, reasoners, and deciders) and thus exists in a large number of different instantiations, which potentially vary within and across different domains and tasks. It is difficult or even impossible to evaluate data selection independently from the requirements of a particular workflow that solves a specific task. Thus, the evaluation requirements within LarKC are complex and multi-faceted and need to be considered in the context of particular tasks.
- *LarKC vs. traditional IR tasks*: In a similar vein, there are substantial differences between evaluating a semantic web platform that selects and transforms RDF triples and evaluating success in classical IR tasks. In IR we typically distinguish between structured and unstructured information. Semantic web data is by definition structured, whereas classical IR uses the bag of words approach where the units of analysis are the type/token versus the document. In fact, the notion of the document plays a central role in classical IR, but only a subordinate role within a RDF-centric approach.

¹Also see http://en.wikipedia.org/wiki/Precision_and_recall and http://en.wikipedia.org/wiki/F1_score for variants.

- *From optimal to practical:* The benefits and costs of IR are frequently characterized by trade-offs (e.g., between precision and recall, see Section 2.2). In such cases, no meaningful single “optimal” solution exists. Consequently, any evaluative effort has to settle for a weaker interpretation and more modest goals: We want to optimize retrieval performance relative to the current baseline and develop best practices that can provide better methods and recommendations for data selection and retrieval.
- *RDF impedes human evaluation:* Working within the RDF data model (as defined in Section 2.1) implies that human raters will find it difficult to read and evaluate materials and retrieval results, as RDF is simply not designed to be human-readable. This makes several traditional evaluation methods unusable or impractical, as they require human raters to provide feedback on data selection results.
- *Data quantity vs. data quality:* A pervasive problem of the Semantic Web is a lack of quality in retrieval results. In the initial stages of the Semantic Web the main emphasis of data retrieval was on quantity, with the implicit assumption that ‘more is better’. However, as more and more semantic data repositories can conveniently be queried, it is becoming increasingly clear that less can be more. Especially when assuming the perspective of an application user it is obvious that providing a few very good results (e.g., three good links as a response to a search for medical information on Google) may be far preferable to an abundance of bad or merely mediocre ones. Detecting which answers to a query are ‘very good’ ones is a difficult challenge that shifts the focus of data selection efforts from retrieval quantity to data quality.
- *Need for evaluation benchmarks:* A major difficulty in evaluating the quality of retrieval results is the lack of validated benchmarks. For instance, when issuing a complicated SPARQL query to a large semantic repository with in-built selection mechanisms and reasoning capabilities it is very difficult to judge the quality of the returned results *unless* those results are already known. However, developers within LarKC do not currently possess a set of demanding queries with validated answers. Creating such a set will require that human raters evaluate and score the RDF results returned, which—given the machine-readable properties of the format—is a considerable obstacle. However, without better evaluation benchmarks it remains impossible to compare alternative selection methods.
- *Need for user feedback:* A related problem is that data selection methods within LarKC are not routinely evaluated by human users. It is an intrinsic problem of platform development that the programmers implementing a workflow must focus on getting something to run at all and may lack the time, resources or methods required to evaluate the specific needs of actual application users. In fact, even if developers tried to envisage all possible tasks that users may want to accomplish by eventually using their tools, they often lack the same background knowledge or expertise as those users. In short: developers differ from users. Consequently, LarKC will eventually need to implement mechanisms for the systematic collection and use of user feedback.

- *Evaluation is effortful and should aim to yield general results:* Finally, conducting informative evaluative efforts and taking comprehensive measurements (e.g., by testing multiple components and exploring large parameter spaces) is extremely expensive, and the costs of such endeavors need to be balanced against development work in this work-package (WP2), and elsewhere in LarKC. Consequently, a goal of developing any particular evaluation method ought to be to generalize (at least in principle) to other topics or tasks.

The following chapters each addresses some specific aspects of these issues.

3 Evaluation of Data Selection Methods

3.1 Introduction

The goal of subsetting in LarKC is to reduce the problem space for reasoning. That is, the subsetting aims at selecting only the statements relevant to a query and sufficient for reasoning. This is expected to result in a more efficient query evaluation even if sacrificing on quality (e.g. by producing an incomplete answer).

In the previous version of this deliverable (see LarKC D2.7.2, Neth et al., 2011) we reported the experiments with Random Indexing used for the task of subsetting, which is shown to be ineffective for that particular task. However, due to the promising ranking features it has been moved towards the task for noise and outliers detection (see Section 2.2.2 of D2.7.2). In this section, we report results from using the other selection methods: baseline, Random Indexing and Spreading Activation. Note that Random Indexing that we present here is based on a different methodology in comparison to the one evaluated previously in 2.7.2 (Neth et al., 2011) and 2.3.3 (Quesada et al., 2011). More specifically, the Random Indexing method presented here operates on a set of virtual documents generated exclusively from RDF (as described in LarKC D2.5.3, Damjanovic et al., 2011), unlike the method previously tested which combines RDF with human-readable text (e.g., Wikipedia articles). All methods are implemented as a part of the LarKC platform, and the workflow descriptions as well as details about each of the plugin is available from <http://wiki.larkc.eu/LarkcProject/WP2/workflows>.

3.1.1 Objective

We evaluate whether subsetting can reduce the time for SPARQL query evaluation, by first retrieving the relevant subset to a query and then reasoning over that subset, as opposed to reasoning over all available explicit statements. As a result we expect to see how subsetting using different methods (and the corresponding parameters) influence the final set of results (e.g. if there is a decrease in quality as the relevant statements are omitted by the method, etc.). To that end, we conduct an experiment as follows:

- Using three different methods (baseline, Random Indexing (RI), and Spreading Activation (SA)) and different variations of the relevant parameters find the subsets relevant to a query.
- Evaluate the query against the subset: measure efficiency through execution time (time S) and effectiveness through recall (recall S).
- Evaluate the query against the fully materialised repository: measure efficiency through execution time (time M) and effectiveness through recall (recall M).
- Compare time S and time M, recall S and recall M (which is always expected to be 100%).

By conducting this experiment we aim to answer the following questions:

- Is the query evaluation more efficient with any of the selection methods? (time $S < \text{time } M$, while recall $S = \text{recall } M$)

- Does the quality of results measured through recall degrades as the time is reduced? (time $S < \text{time } M$, while recall $S < \text{recall } M$)
- How do various parameters of individual methods influence the execution time and quality of results?
- On which queries the specific method works or does not work?

3.1.2 Evaluation Setup

As precision refers to a set of correctly retrieved statements divided by the number of the retrieved statements, in our case we did not measure it as it was always 100%. This is due to the way we set up the experiment, where the final result is found through evaluating the SPARQL query against the subset, as opposed to evaluating it against the full materialised repository. Therefore, we aim to get as high recall as possible. Recall here refers to the number of correctly identified statements out of those that should have been retrieved if the query was evaluated against the fully materialised repository.

At the same time we measured the time for evaluating the SPARQL query against the subset. We compare that against the time to evaluate the query against the fully materialised repository, which includes both explicit and inferred statements. We used OWLIM with *ruleset = rdfs*. To increase the significance of our results we evaluated each query 1000 times and here we report the average values.

Dataset We experimented with two datasets, MusicBrainz and DBpedia 3.6, which are the datasets used in the recent Question-Answering over Linked Data (QALD-1) challenge¹. We selected a set of queries and then slightly modified them to make them suitable for testing the selection methods. This is especially the case for DBpedia dataset as none of the original queries required any kind of reasoning. Our modifications include adding specific constraints so that the query can not return any results unless reasoning is used (see Queries 4-10 in Appendix A.2). Due to the ontology structure we focus on *rdfs* reasoning only as DBpedia contains around 360 thousand statements with the *rdfs:subClassOf* predicate. On the other hand, the MusicBrainz dataset is different as even without the requirement for reasoning in the queries they become challenging due to the ontology structure which relies on blank nodes (see queries in Appendix A.1). Another challenge with the MusicBrainz dataset is the existence of a large number of duplicate names making thus the task of selection more difficult.

Methodology To perform selection, as our primary goal is to reduce the problem space for reasoning, the method needs to operate on the repository which does not include any inferred statements. As we use OWLIM which materialises the triples based on the given ruleset we load the repository with *ruleset = empty* which means that no statements will be inferred. When the SPARQL query is received, all keywords are first extracted and the selection method uses them to find the relevant subsets from the repository with explicit statements only. Keywords include all URIs and literals used in the query.

¹<http://www.sc.cit-ec.uni-bielefeld.de/qald-1>

Different methods can then use these keywords differently. They can be used as individual keywords, or they can be used to construct the document which is then used to find the most relevant subset for that document.

Once the subset is found, it is loaded into the OWLIM repository with *ruleset = rdfs* to simulate reasoning and the original SPARQL query is evaluated against that subset repository in order to find the correct answer.

In what follows we present the results when this methodology is applied to three different selection methods. First we present the results from the application of the Random Indexing and baseline on the task of selection using the MusicBrainz and Dbpedia datasets. These two methods are comparable due to the similar parameter that can be used. Next, we present the results from the application of the Spreading Activation for the task of selection using the DBPedia dataset.

3.2 Results

3.2.1 Baseline and Random Indexing

In order to apply any Information Retrieval method to an RDF Graph the first step is to decide on the abstractualisation which is to be used as a document. Naturally, all URIs and literals become terms. Below we briefly describe how the documents are generated. More details can be found in D2.5.2 and D2.5.3.

Let us recall that each URI in the graph is represented by:

- *A representative subgraph*: a set of statements that represent a URI. The number of statements in the subgraph is controlled by the parameter *depth*. This parameter defines the number of steps from the representative URI that will be taken to extend the subgraph – the higher the value for the parameter *depth* the larger number of statements in the subgraph.
- *A virtual document* being indexed which is essentially a bag of URIs and literals generated from a representative subgraph. The depth parameter also influences the size of the virtual document – the higher the depth, the larger the virtual document.

After the RDF graph is transformed into a set of virtual documents, we can proceed to indexing them and generating:

- A Lucene index where we analyse the virtual document content and store its representative URI. Each representative URI will then be treated as a document, and each URI or literal inside the virtual document will be treated as a term. We use this index as the *baseline*, and also as an input to the Random Indexing method².
- The semantic space using Random Indexing. Based on experiments in D2.5.3. we decide to build the space using dimensionality 150 and seed length 4.

²For the experiments reported in this deliverable we used the SemanticVectors library which uses the Lucene index to build the RI semantic space. This can be achieved using the LarkC plugin RISearchPlugin, platform v.2

For each SPARQL keyword, we identify a list of ranked virtual documents – we then generate a subset from the statements that are used to generate these virtual documents. As both methods rank the results (tf.idf for baseline, and cosine similarity for random indexing), we experimented with the cutoff employing the following heuristics:

1. cutoff=1;
2. retrieve results for cutoff;
3. evaluate recall for the identified subset
 - (a) if recall < 100% or different from the recall in the previous round then increase cutoff and go to 2;
 - (b) else return results.

MusicBrainz: Results Figure 3.1 shows how the average recall using the baseline method across all queries changes when the depth parameter is increased from 1 to 2. We can see that the overall recall is much higher for depth 2, which is not surprising given that this dataset relies on blank nodes. With regard to the execution time, it looks very similar for both the subset generated using depth 1 and 2, with, surprisingly, depth 2 being even slightly faster.

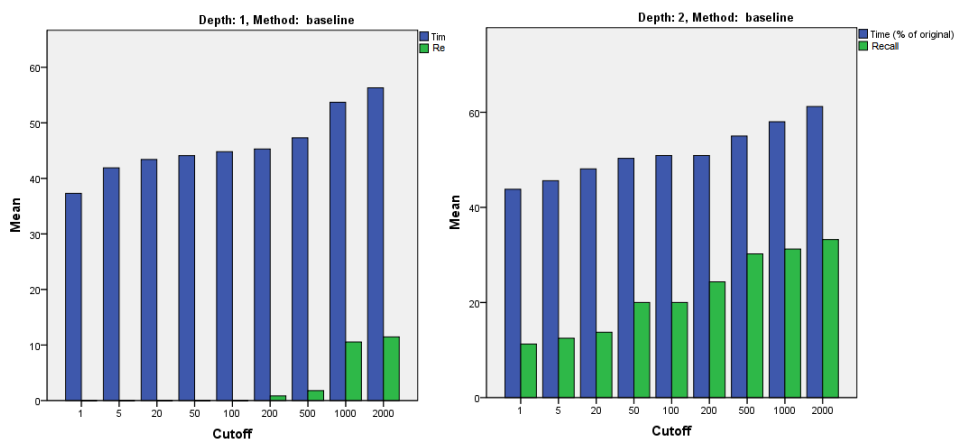


Figure 3.1: Recall and the execution time using the baseline method and virtual documents of depth 1 and 2. Time (% of original) is a relative measure showing how the execution time against the subset at the specific cutoff point compares to the execution time of the queries against the fully materialised repository which we call ‘original’, $\% \text{ of original} = (timeS/timeM) * 100$

Figure 3.2 shows how the average recall using Random Indexing across all queries changes when the depth parameter is increased from 1 to 2. Indeed, depth 2 produced twice as good results as depth 1. We can see that for both methods, the increase in cut-off and depth values both increase the average recall. However, this also increases the execution time. This is because the number of statements in the subsets grows almost exponentially as the cutoff and depth parameters are increased, see Figure 3.3. Interestingly, similar to the baseline, the increase in depth increases recall but decreases the overall execution time.

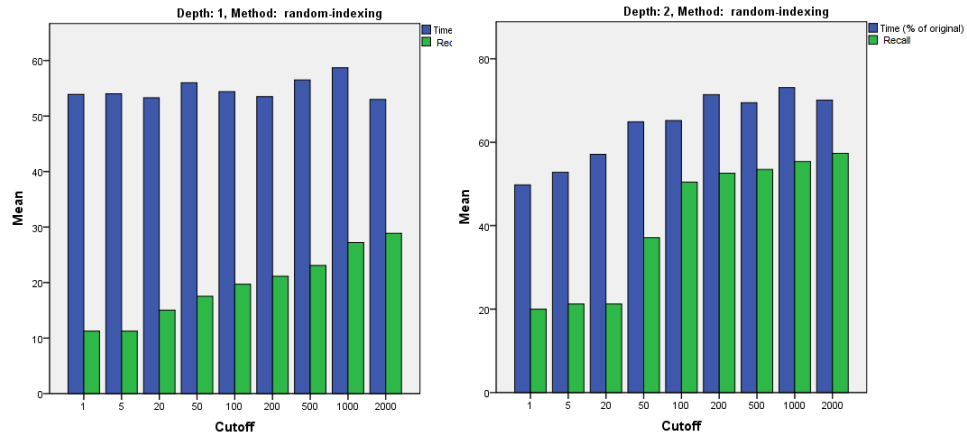


Figure 3.2: Recall and the execution time using Random Indexing and virtual documents of depth 1 and 2.

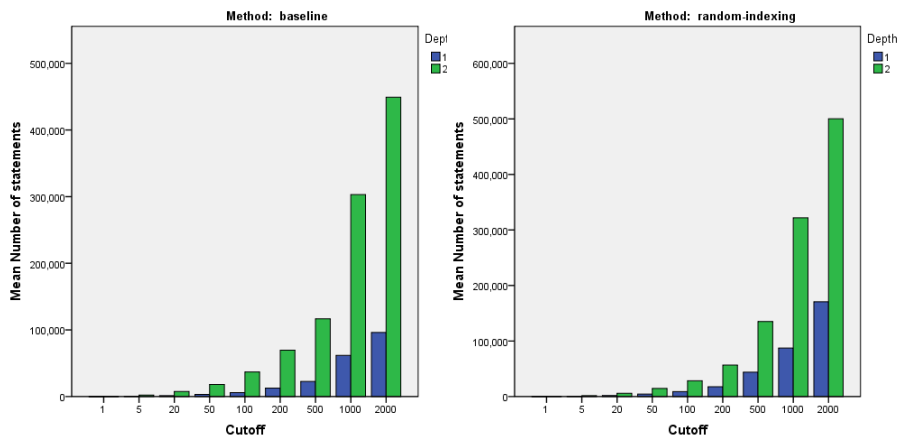


Figure 3.3: The average number of statements in the subsets across all queries. For a comparison, the fully materialised repository contains around 19.2 million statements.

Figure 3.4 shows a comparison between the Random Indexing and the baseline. We can see that the RI method arrives much faster to the higher recall in comparison to the baseline. The average recall for the RI method and depth 2 is reaching almost 60%, while for the baseline and the same depth the recall is just below 30%.

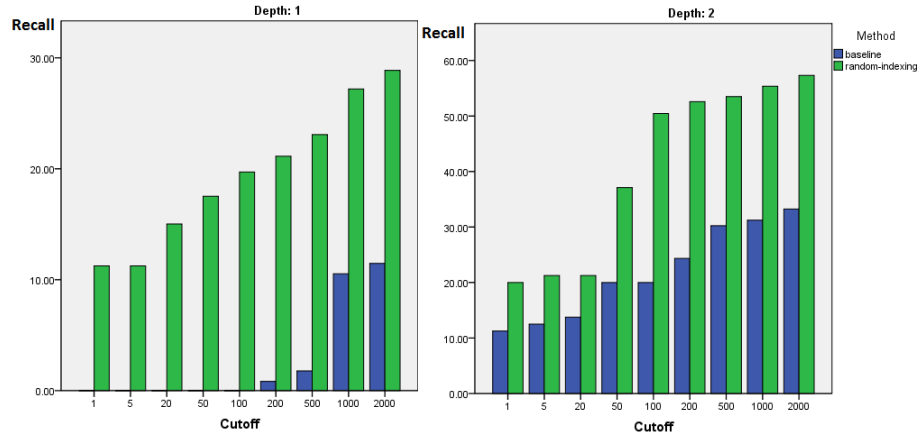


Figure 3.4: The average recall for the baseline and random indexing methods using the same cutoff points across all queries and depths 1 and 2.

However, looking at performance on the query level, it seems that half of queries arrived at the recall of 100% using RI, while only 33.33% reached the same performance using baseline. We can also see that with the increased cutoff the recall increases even if not reaching the maximum for all queries. The recall remained zero only for one query (Query 10) even for the maximum cutoff that we used. Figures 3.5 and 3.6 show the detailed distribution of recall by query for the baseline, while Figures 3.7 and 3.8 show recall by query for the RI method.

Although on average the results of the RI method were better than the baseline, it is interesting to see that in some cases the baseline found the correct result by selecting a smaller subset, even with cutoff 1 (Query 11, depth 2), while for the same query the RI required 50 subgraphs. This is because all keywords mentioned in the query exist in the virtual document which describes the URI which has *dc:title* “Queen”.

A very similar trend exists for Queries 4 and 8. For Query 4 the baseline found the correct answer within the cutoff 50, while the RI method reached the same performance within the cutoff 200. Query 4 has similar characteristics to Query 11 with the difference that it requires the blank nodes to be used in order to find the answer. This is why although baseline with depth 2 returned the answer the baseline with depth 1 did not.

For Query 8, while the baseline reached 100% within 500 cutoff, the same query reached only 50% with RI within the cutoff 2000. This is surprising given the complexity of the query.

Query 5 was problematic as it required finding the albums which are live, and at the same time by the Beatles. This means that the method required a selection of 88 statements from 3008 which is how many albums by the Beatles exist in the fully materialised repository.

Query 10 could not at all be answered using any of the generated subsets. The reason is the complexity of this query which contains only one keyword useful for selection (Abby Road), and then it requires going two steps from the URI with *dc:title* ‘Abby Road’ in order to find all answers.

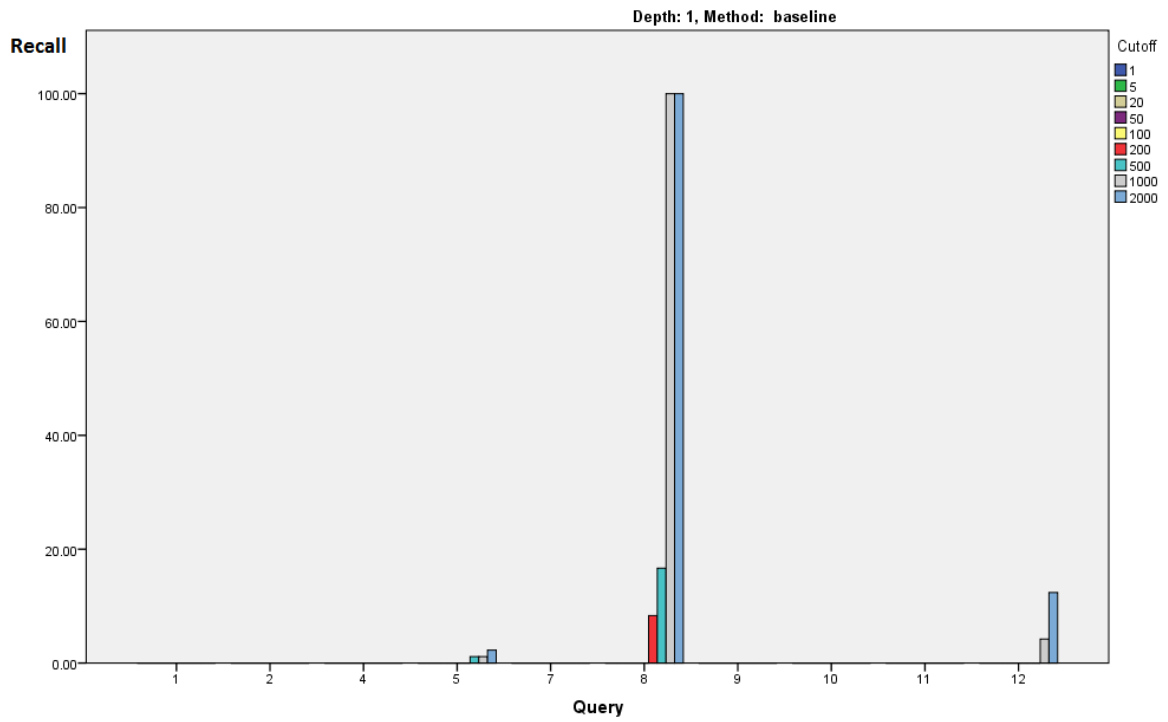


Figure 3.5: Recall by query for the baseline method and depth 1.

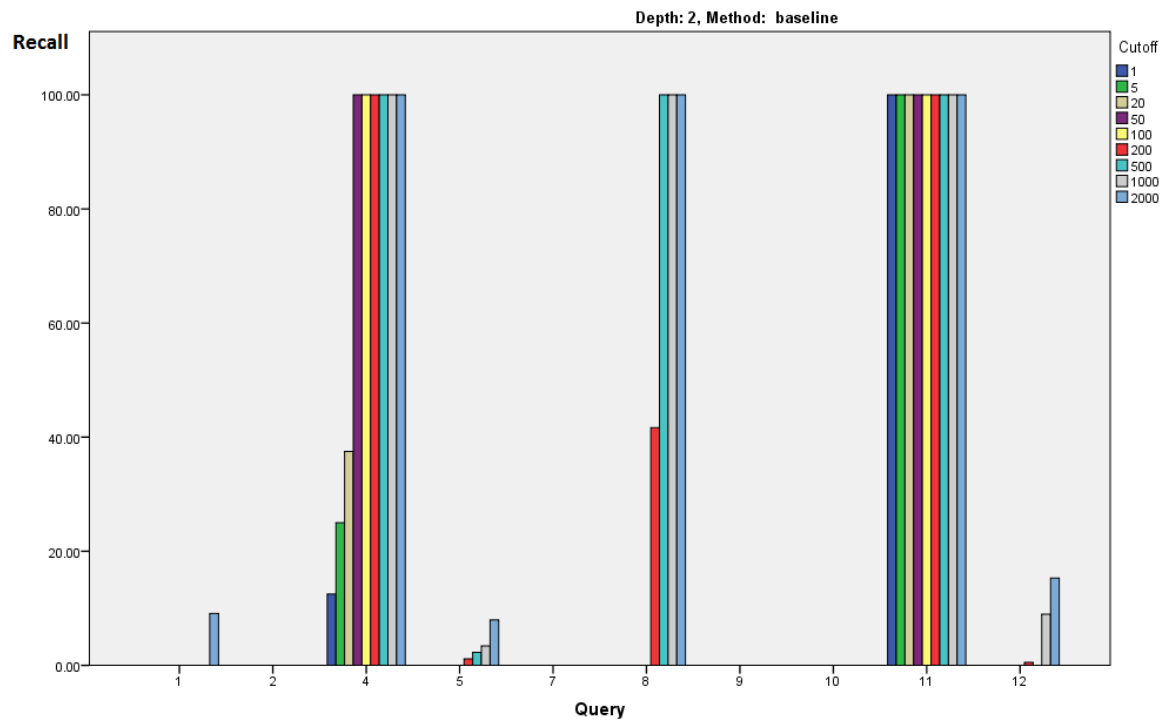


Figure 3.6: Recall by query for the baseline method and depth 2.

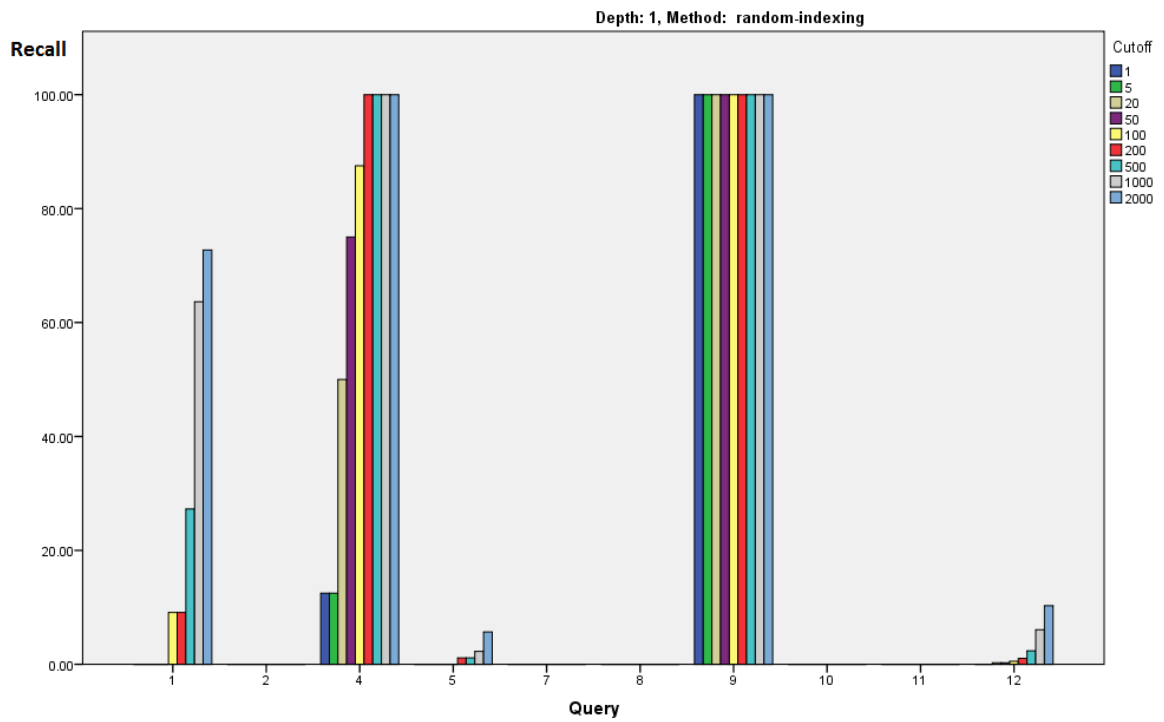


Figure 3.7: Recall by query for the RI method and depth 1.

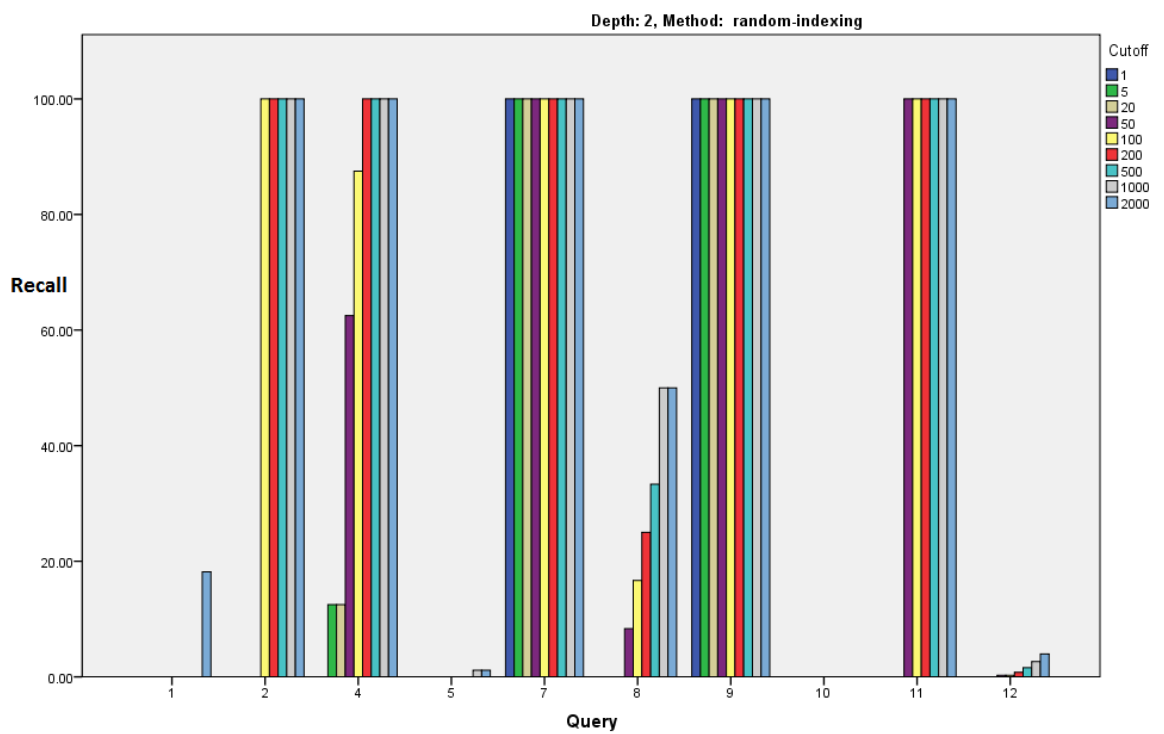


Figure 3.8: Recall by query for the RI method and depth 2.

DBpedia: Results The overall recall for the DBpedia dataset was much lower than for the MusicBrainz dataset using both methods, see Figure 3.9.

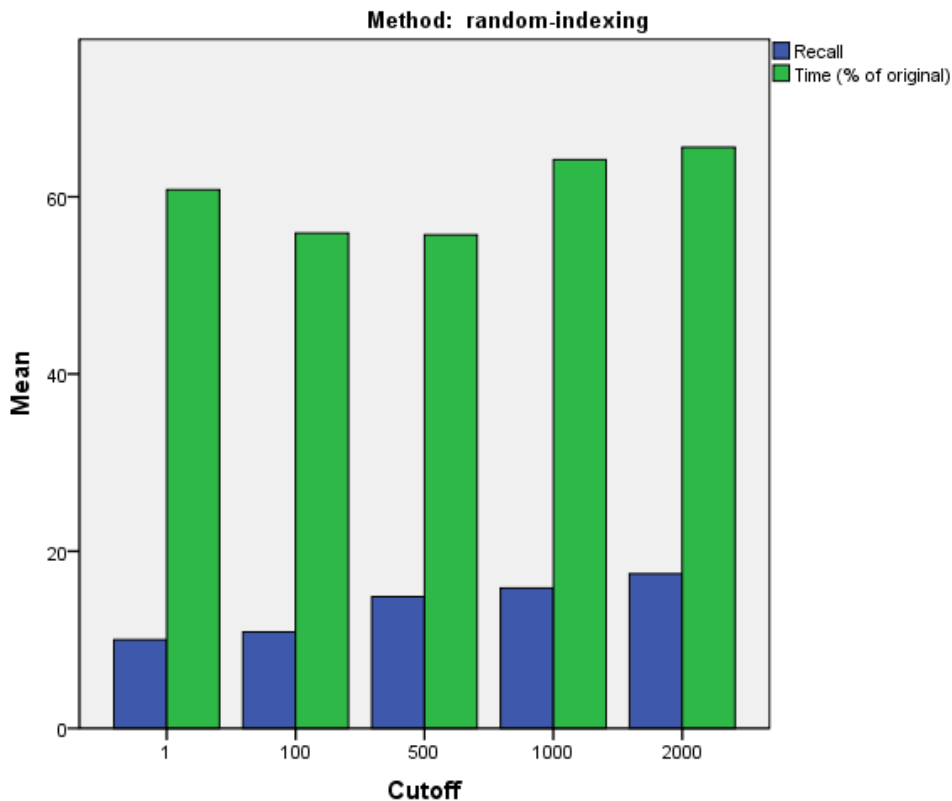


Figure 3.9: Recall and the execution time for the DBpedia dataset, depth 1.

Similarly, the average recall with the Random Indexing method outperformed the average recall with the baseline, however, the positive results were for three queries only. This is because, within the cutoff range from 1 to 2,000, none of the queries that required reasoning could return any results. The reason is that the methods could not identify as highly relevant those virtual documents that included the necessary *rdfs:subClassOf* relations. There are several ways that potentially might improve these results:

- *Increase the cutoff point.* However, this is not a reasonable solution due to the time to find the subset becoming intolerable. In addition, while this will work for queries that require fetching a small number of statements with *rdfs:subClassOf* relation, it will not work for all of them. For example, it might work for Query 4, but not for Query 5. For Query 4, the statement that is missing from the retrieved subsets is³:

```
yago:Peptides rdfs:subClassOf yago:Compound114818238
```

If this statement is added to the smallest subset (with cutoff=1) the query would return the correct answer. However, none of the subsets included this statement

³We use *yago* instead of the full namespace <http://dbpedia.org/class/yago/>

meaning that according to both methods, `yago:Peptides` is not found as similar to `yago:Compound114818238` although both URIs appear together in the virtual document that represents `yago:Peptides`.

For Query 5, there are 14 different classes which are *rdfs:subClassOf* `yago:Amide114724264`, however, only five out of these are direct subclasses. This complicates the issue as in order to answer this query we would need to find all statements that are used to infer that the 14 classes are subclasses of the specified class. One example set of those statements include:

```
yago:BetaEndorphin114809247> rdfs:subClassOf yago:Endorphin114809057
yago:Endorphin114809057      rdfs:subClassOf yago:Peptide114743046
yago:Peptide114743046       rdfs:subClassOf yago:Amide114724264
```

- Expanding the list of SPARQL keywords by fetching all subclasses of the class mentioned in the query. Obviously this can only work for a certain kind of the query – those that mention the class URI. In other words, for each SPARQL keyword we can execute the following query:

```
select ?x {
?x rdfs:subClassOf KEYWORD
}
```

and then find the subset by including all representative subgraphs for the expanded set of keywords. The problem with this approach is that, as we are in the context of explicit statements only, we need to recursively repeat this query until we reach the top class in order to find all subclasses of our keyword.

- *Loading TBox* together with the identified subset into the rdfs repository. This option looks as most feasible and we considered using it in order to see whether our results can improve. However, this added an improvement of only 10% as the Query 4 was successfully answered even with cutoff 1, however, for all other queries it did not have any effect (see Figure 3.10).
- *Increasing depth parameter* from 1 to 2 increased the result for the MusicBrainz dataset. Therefore, we experimented with this option and the results show that depth 2 increases recall by 40% as Queries 6, 7, 8 and 10 get answered and the execution time remains similar to the previously discussed (73% of original). This means that the final recall for DBpedia reached 63.8% (note that we have added TBox to each subset). However, one problem with the virtual documents of depth 2 is that they can be extremely large for large datasets and thus require a long preprocessing step before these documents are indexed and used for finding subsets. In addition, once the semantic space is generated the time to find subsets is such that it makes the reduction of the execution time negligible. Queries 5 and 9 were problematic and could not be answered because the variable that needs to be retrieved (`?uri`) is a subject and the virtual documents are generated so that for each URI we include those statements where that URI is a subject. It is worth exploring in the future generating virtual documents for a URI so that it includes also the statements where that URI is an object.

- *Modify the methodology for generating virtual documents* so to include not only statements with literals for all URIs but also their `rdf:type` relation. This way the virtual documents of depth 1 are likely to produce the same results as the current depth 2 (this becomes especially significant for large datasets in the context of LarKC as depth 2 becomes impractical for large graphs), and the execution time is expected to be reduced.

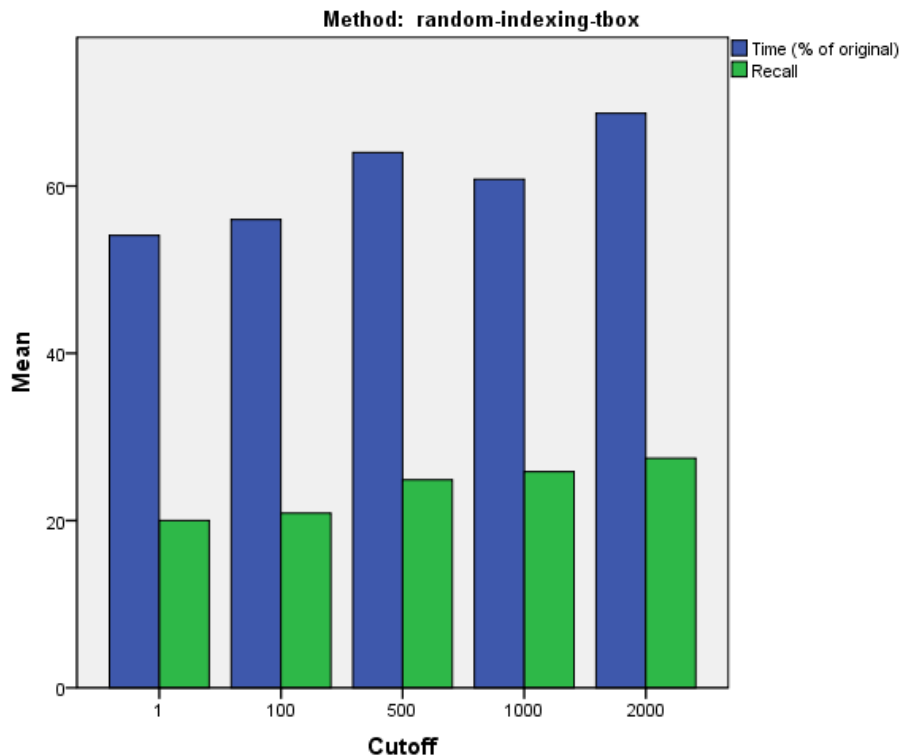


Figure 3.10: Recall and the execution time for the DBpedia dataset, depth=1, with TBox loaded in addition to the identified subsets.

3.2.2 Spreading Activation

LarKC partner Ontotext has implemented *spreading activation* (SA) mechanisms (see Todorova et al., 2009) and conducted basic benchmarking and efficiency tests on their implementation (Peikov et al., 2010). This led to the proposal of two approximate approaches to SA—a node-selection-based SA algorithm and a cluster-based SA method. The implementation of these approaches is described in the LarKC deliverable 2.4.3 (Grinberg, Stefanov, Stefanov, & Peikov, 2011), which also reports some measures of precision and recall along with a scalability analyses.

Spreading Activation (SA) selection method in LarKC works on an input SPARQL query by extracting all URIs from it and then using them as source nodes to spread activation across the RDF graph. After the SA process decays away the active subset of the graph is imported in a separate repository (OWLIM *ruleset = rdfs*) and the input SPARQL query is evaluated against the smaller repository instead of on the original bigger and fully materialised one.

The SA process itself is controlled by several parameters the most important being:

- decay factor
- initial activation
- filter threshold
- firing threshold
- number of iterations
- max nodes fired per iteration

We experimented with different values for those parameters aiming to produce a relatively small subset of the original dataset still providing decent recall for query evaluation. Here we present two results of the experiment obtained by different parameter values.

As first attempt we executed 2 iterations of SA with initial activation of 1.0, decay factor of 0.3 and firing threshold of 0.7. We fired no more than 100,000 nodes per iteration and consider active those nodes that had activation no less than 0.9 after the process finished. Table 3.1 summarizes the results:

Table 3.1: Results of the first SA selection evaluation experiment.

Query No.	Results in dataset	Results in subset	Recall (in %)	Time (% original)	Statements in subset	% of original dataset
1	1	1	100.00	67.00	99,820	0.03
2	34	12	35.29	59.00	199,383	0.06
3	53	2	3.77	48.00	130,811	0.04
4	1	1	100.00	76.00	1,594	0.00
5	406	0	0.00	54.00	118	0.00
6	1	1	100.00	89.00	23,939	0.01
7	1	1	100.00	95.00	34,180	0.01
8	2	2	100.00	77.00	553,466	0.17
9	21	0	0.00	9.00	307,083	0.10
10	2	2	100.00	62.00	346,382	0.11

The 0% recall in queries 5 and 9 is due to the fact that they contain URIs on object positions and variable in subject position. Because SA always spreads activation from subject to object and not the other way around the subset produced did not contain triples to satisfy the triple pattern in those queries. This immediately shows a deficiency of the method which could be solved in the future by implementing the SA method in both directions (object to subject as well as subject to object).

In an attempt to increase the recall for queries 2 and 3 (5 and 9 cannot be improved due to the reasons stated above) we decided to run SA for 3 iterations (instead of 2) and also to increase the decay factor to 0.5, effectively increasing the number of selected nodes. As this increased dramatically the selected subset we also increased the firing threshold to 0.8 and limited further the maximum number of selected nodes per iteration to 10,000. Table 3.2 summarizes the results from those settings:

The results from the second experiment confirm the intuition that by increasing the subset through parameter adjustments we can increase the recall to any desired level.

Table 3.2: Results of the second SA selection evaluation experiment.

Query No.	Results in dataset	Results in subset	Recall (in %)	Time (% original)	Statements in subset	% of original dataset
1	1	1	100.00	75.00	2,693,606	0.84
2	34	14	41.18	89.00	1,775,546	0.55
3	53	9	16.98	81.00	1,902,065	0.59
4	1	1	100.00	89.00	1,594	0.00
5	406	0	0.00	48.00	118	0.00
6	1	1	100.00	85.00	23,942	0.01
7	1	1	100.00	90.00	357,295	0.11
8	2	2	100.00	78.00	3,465,984	1.08
9	21	0	0.00	14.00	2,003,054	0.62
10	2	2	100.00	86.00	1,444,277	0.45

Of course this comes at the cost of also increasing the subset volume and therefore decreasing the speed-up.

3.2.3 Summary and Conclusion

In this section we looked at how three different selection methods can be used for the problem of subsetting. We varied the relevant parameters of the methods in order to see how they influence the results.

For the baseline and Random Indexing, we first examined how the parameter *depth* of the virtual documents influence the results for the MusicBrainz dataset, especially as this dataset relies to a large extent on blank nodes. The overall results show that a depth of 2 reached a much higher recall in comparison to depth 1. With regard to the average recall across the two methods, Random Indexing scored higher (57.32% vs. 33.23%). However, for those queries for which the baseline reached 100% recall, the Random Indexing method was less efficient as it required retrieving a much larger subset in order to reach the same performance.

With regard to the execution time the highest recall for the baseline resulted in 61.2% of the original time, meaning that it is around 1.6 times faster to evaluate the SPARQL query against the relevant subset found using the baseline, in comparison to evaluating it against the fully materialised repository.

With the RI method, the average execution time is 70.10% of the original meaning, which means that it is around 1.4 times faster to evaluate the query against the subset generated by the RI method as opposed to evaluating the query against the full repository. In terms of the quality of results, the maximum recall was 57.3% with a depth of 2 and a cutoff of 2,000.

The DBpedia dataset seemed as more challenging mainly due to the queries that required rdfs reasoning. While the overall recall was higher for the RI method in comparison to the baseline, it was only 13.8% with the highest cutoff (2,000) with depth 1 and our original algorithm to find the subsets. This is because queries from 4 to 10 returned 0 results, as the selection methods could not retrieve successfully all necessary statements with *rdfs:subClassOf* relation. Adding the TBox to the subsets improved the overall recall to 23.8% which is still a very low. However, increasing

depth from 1 to 2 for the virtual documents, the results outperformed MusicBrainz dataset, with recall 67.3%.

It should be noted that these experiments do not report the time to actually find the subsets, which, even if in seconds, can add a significant barrier to using any of the two methods for the task of subsetting. We reported previously in D2.5.3 (Damljanovic et al., 2011) some statistics on how expensive it is to use the methods for searching large indexes and semantic spaces, and we also described some parallelisation possibilities.

With regard to Spreading Activation, the overall recall reached 65.8%, and the average execution time was 42.5% of the original. Hence, this method seems to be the most efficient one. Most queries that did not reach 100% with the reported parameters and any level of recall could probably be reached, however, for the specific kinds of queries none of the methods was shown to be suitable. These are the queries that contain a URI on the object position and a variable in the subject position. The reason for this is that all methods use keywords in the SPARQL queries as a starting point for finding subsets, and while Spreading Activation starts from the URI (taking all statements where the URI is subject and spreading activation further in the same direction), the other two methods operate on virtual documents which are generated by looking into the statements where the particular node is a subject.

4 Theory and Evaluation of Interests-based Selection

4.1 A Theoretic Framework for Interests-based Selection

The interests-based selection method emphasizes that user interests are one of the most important heuristic factors for finding the most relevant RDF triples to a specific user in the context of Web-scale Knowledge bases (Zeng et al., 2011a, 2011b).

Interests can be divided into different types. Such as research interests, interests for visiting a place, interests for collaborating with someone on a paper, etc. In the simplest case, there is only one type of interest related relationship, such as someone “collaborate with” someone, or someone “visited” a place (Here “likes” and “visited” are relationships that define one type of interest respectively).

For the simplest case, interests-based selection is based on one type of interests related relationship. In this case, we propose that the selection process can be based on a quantitative interests-based selection function.

If the number of relationship types that are related to user interests are more than one, different relationships as well as related interests may need to be investigated and evaluated separately. Different types of interests may be ranked qualitatively.

4.1.1 Interests-based Selection Based on Single Interest Type

For single interest type, we firstly consider that a specific user only have one interest, and the selection process is only based on this specified interest.

Let i be the interest of a specific user, and $f(t, i)$ be an evaluation function that represents the frequency of the interest i 's appearance in the triple t (namely, $f(t, i)$ is a nonnegative integer) in the triple set T ($T = \{t \mid t = \langle s, p, o \rangle\}$). The value is based on the appearance of the specific interest i in the triple t . Triples in T are ranked according the value of $f(t, i)$. It is emphasized that interests-based selection strategy assume the end user will be interested in the triples that contain i , no matter which positions it appears in (Namely, the positions of subject, predicate, and object are all considered).

Let $R(t, i)$ be the ranked position of the triple t in the ranked triple set T' according to $f(t, i)$ (Namely, $R(t, i)$ is a positive integer). $R(t, i)$ is negative relevant to $f(t, i)$, namely, for two arbitrary triples $t, t' \in T'$, they satisfy that:

$$f(t, i) > f(t', i) \implies R(t, i) < R(t', i).$$

Algorithm 1 (denoted as ISSI algorithm) illustrates the major steps for interests-based selection on single data source based on a single interest. The selection process is an iterative process. Each time, Top- K triples according to $f(t, i)$ are selected for processing. If there is still extra time, another Top- K triples will be selected from the rest of the triple set.

In most cases, a specific user may have multiple interests, which can be denoted by different interests terms. Each interest may be quantitatively evaluated. Hence, the selection of interesting triples is based on a weighted evaluation function.

Let i_m be the m th interest in the interests set I (i.e. $i_m \in I$), $F(t, I)$ be the weighted evaluation function for selecting RDF triples, and it is defined as:

$$F(t, I) = \sum_{m=1}^n \omega_m f(t, i_m), \quad (4.1)$$

Algorithm 1: Interests-based Selection based on Single Interest (ISSI)

Input: user interest i , original RDF triple set T

Output: a set of Top- K triples

- 1 Begin
 - 2 Calculate $f(t, i)$ for each $t \in T$ and $R(t, i)$ according to $f(t, i)$;
 - 3 Re-rank each t according to $R(t, i)$ and generate an ordered triple set T' ;
 - 4 Select Top- K triples from T' as output and mark the selected triples for further processing;
 - 5 If time allows and the user are not satisfied with the selected triples, output the selected triples from Step 4, and check whether unselected triples are available. If yes, goto Step 4 to select new triples from T' , else, go to Step 6;
 - 6 End
-

where ω_m is the weight for the interest i_m according to a specific interest evaluation function. The interest evaluation function that assign a weight to a specific interest can be defined from different perspectives (e.g., cumulative interests, retained interests, interests longest duration, and the interests cumulative duration function introduced in (Zeng et al., 2011a, 2011b) can be used). The original triple set T is re-ranked to an ordered triple set T' according to $F(t, I)$. The rank number of the specific triple t in the ranked triple set T' is represented as $R(t, I)$, and it is negative relevant to $F(t, I)$. Namely, for any two arbitrary triples $t, t' \in T'$, they satisfy that:

$$F(t, I) > F(t', I) \implies R(t, I) < R(t', I).$$

Algorithm 2 (denoted as ISMI algorithm) illustrates how interests-based selection based on multiple interests is designed.

Algorithm 2: Interests-based Selection based on Multiple Interests (ISMI)

Input: a set of user interests I , original RDF triple set T

Output: a set of Top- K triples

- 1 Begin
 - 2 Select an interest evaluation function and acquire a set of interests I as well as their values;
 - 3 Calculate $f(t, i)$ for each $t \in T$ and $i \in I$;
 - 4 Calculate $F(t, I)$ for each $t \in T$ according to Equation 4.1 (the weights of each interest is decided by the interest value calculated in Step 2) and calculate $R(t, I)$ according to $F(t, I)$;
 - 5 Re-rank each t according to $R(t, I)$ and generate an ordered triple set T' ;
 - 6 Select Top- K' triples from T' as output and mark the selected triples for further processing;
 - 7 If time allows and the user is not satisfied with the selected triples, output the selected triples from Step 6, and check whether unselected triples are available. If yes, goto Step 6 to select new triples from T' , else, go to Step 8;
 - 8 End
-

We can conclude that the selection of RDF triples based on multiple interests is a combinatorics analysis process for interesting triples to a specific user based on his/her set of interests. Although theoretically, it would cost more processing time compared to only considering one interest, nevertheless, the selected triples by Algorithm 2 would be much more relevant to the user compared to the results from Algorithm 1.

4.1.2 Interests-based Selection Based on Multiple Interest Types

If the interests related triples only describe one type of user interests in the RDF triple sets, then Algorithm 1 and Algorithm 2 can be used for ranking and selecting RDF triples. Nevertheless, in the real world, the triple sets always contain many types of interests related relationships. For example, in the SwetoDBLP RDF dataset, there are many types of interest relationships, such as someone's collaboration interests with some authors, and research interests on some topics.

In this case, we categorize the RDF triples that contain different interests related relationships into different sets of triples. Ranking strategies introduced in Algorithm 1 and Algorithm 2 can be used within different sets of triples. Nevertheless, ranking strategies among different interests related triple sets need to be developed.

Since the comparison among different interests related triple sets may not be quantitatively evaluated, it may be qualitatively compared. The order of these triple sets can be manually assigned according to the requirement of specific applications.

Let T and T' be two arbitrary sets of interests related triples. Considering selection for user centered query applications, where ranking is manually assigned. For example, $R(T) < R(T')$. In this case, the selection process will be executed on the triple set T , when all the triples in T are selected, then the selection process will go for selection on T' . Within the triple set T (or T'), Algorithm 1 and Algorithm 2 are used for ranking of triples for Top- K triples selection.

Algorithm 3: Interests-based Selection based on Multiple Interest Type (IS-MIT)

Input: A set of user interests I , a set of ordered triple sets $\mathcal{T} = \{T, T', T'', \dots\}$, etc.

Output: a set of Top- K triples

- 1 Begin
 - 2 Select an RDF triple set from the set of ordered RDF triple sets \mathcal{T} , and delete the selected one from \mathcal{T} ;
 - 3 Apply Algorithm 1 and Algorithm 2 within the selected triple set;
 - 4 If time allows, goto step 2;
 - 5 End
-

The manual assignment order might be usecase sensitive, and may differ from each other in different scenarios according to different requirements in various usecases.

4.2 Scenarios and Usecases for Interests-based Selection

During the LarKC project, we focus on two scenarios for demonstrating the interests-based selection. Namely, personalized semantic literature search (Scenario 1), and active academic visit recommendation (Scenario 2). In Scenario 1, we focus on the application of interests based selection on single interest type (to verify methods introduced in Section 4.1.1). In Scenario 2, multiple interests types are considered (to verify methods introduced in Section 4.1.2).

4.2.1 Personalized Semantic Literature Search

In this scenario, we observe that a traditional semantic search algorithm provides exactly the same results to all users, and users may not be satisfied, as the result list does not take into account their background interests. Hence, we aim to find the most relevant results on the large scale dataset for users, and we aim to solve potential scalability issues by personalization.

Compared to a traditional personalized search strategy, such as query expansion, we aim to improve its efficiency through interests-based selection. A comparative study on query expansion and interests-based selection has been discussed in, and will be reviewed in Section 4.3.

For these two applications, we only extract authors' research interests from their publications, hence, there are only one type of relation that we focus on (namely, author-publication relation in both the SwetoDBLP dataset and the Linked Life Data RDF dump). Due to the characteristics of the interest relation in this scenarios, Algorithm 2 introduced in Section 4.1.1 is used.

Based on this study, we developed two personalized semantic literature search engine. Namely, DBLP Search Support Engine (DBLP-SSE) ¹ for computer scientists and Context-aware Linked Life Data search engine ² for life science researchers.

The RDF dataset that have been used in DBLP-SSE is the SwetoDBLP dataset³, while Context-aware Linked Life Data search engine is based on Linked Life Data.⁴

DBLP-SSE and Context-aware LLD is powered by interests-based selector contributed by WICI. User inputs are simple word terms and SPARQL queries are generated by the system automatically.

Interests-based selection require some basic information about users, such as user name or an URL that uniquely specify a user, so that it can create profiles for them. The following presents a typical query automatically generated by context-aware LLD when the specific user login the system with his name.

```

PREFIX pubmed: <http://linkedlifedata.com/resource/pubmed/>
PREFIX pubmed-author: <http://linkedlifedata.com/resource/pubmed/author/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?meshText ?year
WHERE {
  ?s pubmed:author ?author.
  ?author pubmed:foreName "Olof".
  ?author pubmed:lastName "Selroos".
  ?s pubmed:meshHeading ?meshHeading .
  ?s pubmed:year ?year .
  ?meshHeading pubmed:mesh ?mesh .
  ?mesh rdfs:label ?meshText .
}

```

This query is executed to find all mesh headings for the specific scientist “Olof Selroos”, mesh headings are used to rank user interests based on interests evaluation

¹DBLP-SSE: <<http://www.wici-lab.org/wici/dblp-sse/>>

²Context-aware LLD: <<http://www.wici-lab.org/wici/context-aware-LLD/>>

³SwetoDBLP: <<http://archive.knoesis.org/library/ontologies/swetodblp/>>

⁴Linked Life Data: <<http://linkedlifedata.com>>.

functions introduced in Zeng et al. (2011b). Table 4.1 presents an example that reflects the author’s Top-9 research interests from various perspectives.

Table 4.1: A comparative study of top- K research interests for “Olof Selroos” from various perspectives.

Cumulative Interests (CI)	Retained Interests (RI)	Interests Longest Duration (ILD)	Interests Cumulative Duration (ICD)
Humans [13.0] Asthma [9.0]	Humans [2.911665] Bronchodilator Agents [2.4685957] Asthma [2.395056]	Humans [3.0] Male [3.0]	Humans [6.0] Asthma [6.0]
Administration, Inhalation [9.0] Bronchodilator Agents [8.0] Budesonide [8.0]	Administration, Inhalation [2.2577446] Budesonide [2.1889484] Male [2.082583]	Double-Blind Method [3.0] Asthma [3.0]	Administration, Inhalation [6.0] Budesonide [6.0]
Male [7.0]	Female [2.082583]	Administration, Inhalation [3.0] Bronchodilator Agents [3.0]	Bronchodilator Agents [5.0] Male [5.0]
Female [7.0]	Double-Blind Method [1.8076792] Treatment Outcome [1.8076792]	Treatment Outcome [3.0] Time Factors [3.0]	Female [5.0]
Adult [6.0]		Budesonide [3.0]	Child [5.0]
Dose-Response Relationship, Drug [5.0]			Dose-Response Relationship, Drug [4.0]

After interests evaluation and Top- K interests acquisition, users are required to input some keywords for searching relevant articles. A SPARQL query is generated automatically based on user inputs. For example, if the user input a keyword “Bronchoconstriction” for search, the corresponding query is:

```
PREFIX pubmed: <http://linkedlifedata.com/resource/pubmed/>
PREFIX pubmed-author: <http://linkedlifedata.com/resource/pubmed/author/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT ?pmid ?articleTitle
WHERE
{ ?pmid pubmed:articleTitle ?articleTitle .
FILTER regex(?articleTitle, "Bronchoconstriction", "i")
}
```

The SPARQL query is submitted to and executed on the Linked Life Data website. The number of returned article is 128. With interests based selection, the final query results must contain both user interests (Here we choose Top-9 interests of this user based on the cumulative interest (CI) ranking function defined in (Zeng et al., 2011b).) and original query input. Finally, only 7 articles are obtained. Table 4.2 resents several examples for the query output.

As shown in Table 4.2, each of the query results contain the query keyword and at least one interest term.

Although the number of query results for interests-based selection driven application is much smaller than unrefined query, all the query results are highly relevant to their background interests.

Table 4.2: Sample results for querying with interests-based selection

User	Olof Selroos
Query	Bronchoconstriction
pmid	articleTitle
pubmed-article:21039204	Tidal airway closure during bronchoconstriction in asthma : usefulness of lung volume measurements.
pubmed-article:20965404	Exercise-induced bronchoconstriction in non-asthmatic athletes.
pubmed-article:21411834	Conjugated linoleic acid's lack of attenuation of hyperpnea-induced bronchoconstriction in asthmatic individuals in the short term.
pubmed-article:21085817	Performance of a word labeled visual analog scale in determining the degree of dyspnea during exercise-induced bronchoconstriction in children and adolescents with asthma .
pubmed-article:21348805	Association of the asthma control questionnaire with exercise-induced bronchoconstriction .

4.2.2 Active Academic Visit Recommendation

In this scenario, we aim to provide an Active Academic Visit Recommendation Application (*AAVRA* for short) for scientific researchers in an active way as long as their personal information can be acquired through (semantic) data on the Web. Users need to log in the system with their name or account, and specify which country or city they want to visit. Then, the recommendation system will extract their interests from multiple data sources to automatically build their interests profiles for recommendation.

In this scenario, the Semantic Web Dog Food (SWDF for short) dataset ⁵ is used. In addition, user related data (such as user profile information, persons whom the user follows, and tweets that the user wrote) are generated as an RDF triple set from Twitter in real-time (Through Twitter Interests Identifier contributed by WICI). It is assigned with a graph name and passes through different plugins.

User interests can be divided into many types. For SWDF data, author-publication related triples reflect specific authors' research interests and their interests in collaboration with other authors. For Twitter data, they reflect users' interests on certain topics and their interests in following someone, or comment on, or retweet their tweets. These types of interests are considered to be different from each other. Hence, this scenario is designed to be used to verify the interests-based selection methods based on multiple interests types.

In *AAVRA*, we want to help users find interesting places for academic visit automatically and actively. Since the scenario is restricted to academic visit, we emphasize that if a user is interested in collaborating with someone in a place (namely, the ex-

⁵RDF/XML dumps of Semantic Web Dog Food data: <http://data.semanticweb.org/dumps/>

plicit or potential collaborators are the user's interests), then he/she is interested in visiting the place. The recommendation is designed to be based on explicit and potential personal relationships. We consider 5 levels of interests for a specific user in this scenario. Here we have the following predicate denotations to formalize the interests in different levels.

- $SWDF(p)$: p is a person who is an author in the Semantic Web Dog Food dataset.
- $Coauthor_{SWDF}(p, u)$: p and u are two arbitrary persons who satisfy the condition that $SWDF(p) \wedge SWDF(u)$, and they are coauthors based on the record in the Semantic Web Dog Food dataset.
- $PCoauthor_{SWDF}(p, u)$: p and u satisfy that $SWDF(p) \wedge SWDF(u) \wedge \neg Coauthor_{SWDF}(p, u)$.
- $Twitter(p)$: p is a person who owns a Twitter account.
- $TFing(u, p)$: p and u are two arbitrary person who satisfy that $Twitter(p) \wedge Twitter(u)$ and u is following p .
- $SIT(p, u, K)$: p and u satisfy that $Twitter(p) \wedge Twitter(u)$, and they have at least K interests in common based on their Tweets.

Recommendations that belong to different levels of interests for *AAVRA* can be decided by the following formula in Table 4.3:

Table 4.3: Interests level definition for *AAVRA*

Level	Corresponding Triple Set	Formula
1	T_1	$Coauthor_{SWDF}(p, u) \wedge TFing(u, p)$
2	T_2	$Coauthor_{SWDF}(p, u) \wedge \neg TFing(u, p)$
3	T_3	$TFing(u, p) \wedge PCoauthor_{SWDF}(p, u)$
4	T_4	$TFing(u, p) \wedge SIT(p, u, K) \wedge \neg SWDF(p)$
5	T_5	$TFing(u, p) \wedge \neg SIT(p, u, K) \wedge \neg SWDF(p)$

As can be observed in Table 4.3, different levels of interest correspond to different combinatorics strategies for selection. According to Algorithm 3 introduced in Section 4.1.2, we manually assign that: $R(T_1) < R(T_2) < R(T_3) < R(T_4) < R(T_5)$. Hence, the interests-based selection process will be executed over these levels of interests sequentially.

The workflow for the *AAVRA* usecase is as follows:

Twitter-Interests Identifier \rightarrow SWDF Identifier \rightarrow Interests-based Combinatorics Selector \rightarrow Geo-Location reasoner

The Twitter-Interests Identifier is used to identify interesting contents from Twitter that are related to specific users (e.g. following, tweets, profiles). SWDF Identifier is used to identify relevant triples that are related to the specific user (e.g. coauthors and affiliation related triples). Interests-based Combinatorics Selector is used to select

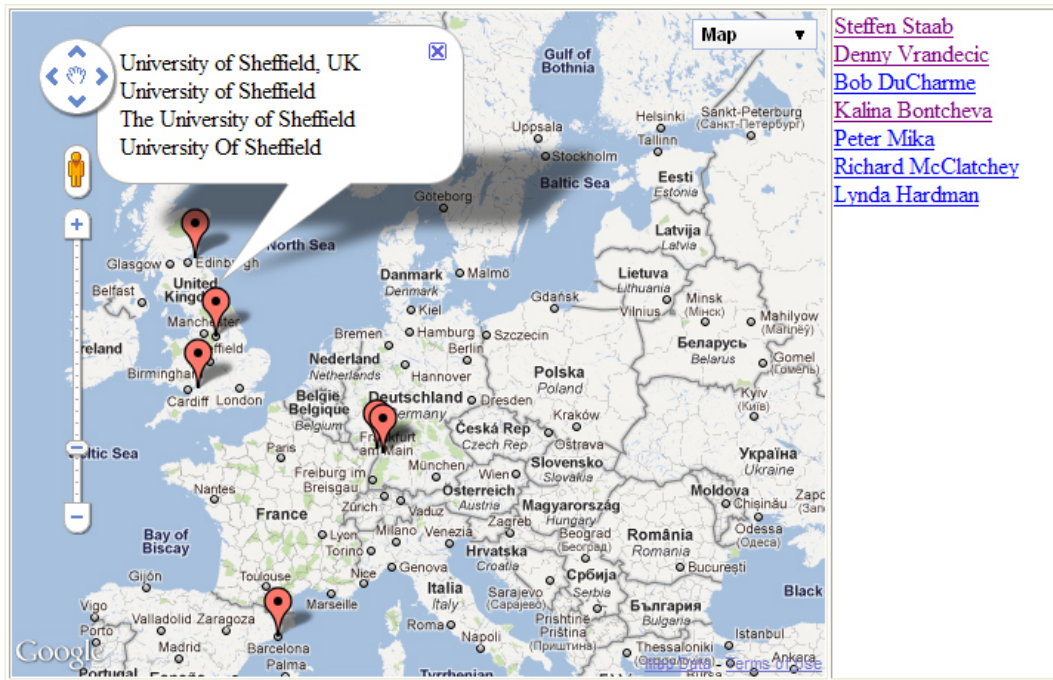


Figure 4.1: Academic visit recommendation for “Frank van Harmelen” to the U.K.

user interests related triples through different combinatorics analysis strategies. Geo-Location reasoner is used to reason over selected triples and mark recommendations on Google Maps.

Table 4.4 presents a partial example on different levels of interest for “Frank van Harmelen” in *AAVRA*. They will be provided to the end user levels by levels. The recommendation place for academic visit will be based on these selected people.

Table 4.4: Different levels of interests for “Frank van Harmelen” in *AAVRA*

Interests Levels	Number of Satisfying Results	Sample Results
1	1	Paul Groth
2	15	Frank van-Harmelen, Zhisheng Huang, Zharko Aleksovski, Eyal Oren, Jacopo Urbani, Spyros Kotoulas, Henri Bal, ...
3	11	Kalina Bontcheva, Lynda Hardman, Peter Mika, Steffen Staab, Denny Vrandecic, Ivan Herman, Michael Hausenblas, ...

Figure 4.1 presents an example of academic visit recommendation to the U.K for the user “Frank van Harmelen”. Some recommendations are made based on his potential collaborators (authors in the Semantic Web Dog Food, but still not collaborators) and at the same time may be based on persons whom the user follows on Twitter (interests discussed in Level 2). Namely, the recommended places are: University of Sheffield

(where Kalina Bontcheva is from), University of the West of England (where Richard McClatchey is from), etc.

4.3 Evaluation for Interests-based Selection

Our assumption is that user interests are explicitly or implicitly available in a RDF triple set. When a user logs into an application powered by interests-based selection, the selection process is started. In real-world use cases, user account information may be submitted in two different ways. It can either be submitted together with a query (Situation 1), or it can be submitted independently as a log in process, and the user query is submitted later (Situation 2). In Situation 1, the selection process will cost additional time compared to the one without interests-based selection. Hence, this requires additional processing with the goal to lead to a better quality of results. In Situation 2, the selection process starts (much) earlier than the user query processing. Hence, the time slot between the user logs into the application and submits a query can be used for data selection.

In LarKC Deliverable D2.3.2 (Quesada et al., 2010), we introduced the evaluation of interests-based selection based on the DBLP Search Support Engine usecase. Here we briefly review the results reported there. We compare three approaches for answering a specific query, namely, standard query by users (Strategy 1), interests-based query refinement (Strategy 2), and query after interests-based selection (Strategy 3, following Situation 2 discussed in the first paragraph of this section).

The result showed that Strategy 3 is much more effective than Strategy 1 and 2. We did the comparative study based on 10 subsets of SwetoDBLP with different scales. Note that SwetoDBLP is divided into 20 subsets (50M for each) with almost equal sizes, the subsets we use for evaluation follows that $\frac{T_{n+1}}{T_n} = \frac{n+1}{n}$, where n is a positive integer that satisfy $n \in [1,9]$, T_{n+1} and T_n are two neighborhood subsets for the scalability test. Namely, the 10 different test sets contain 2, 4, 6, 8, ..., 20 equal size subsets of SwetoDBLP, respectively.

For evaluation metrics, we mainly focus on the comparative study of processing speed (see Neth et al., 2011) of the above three strategies. The reason is that interests-based selection does not focus on providing users as many relevant results as possible or results as complete as possible. Instead, it focuses on providing the most relevant results that are related to user background interests as quickly as possible.

We can conclude from the experimental results in Table 4.5 that under each scale, Strategy 3 can save more than 90% of the time compared to Strategy 2, while it can save around 80% of the time compared to Strategy 1. In addition, each selected triples from Strategy 3 are relevant to the specific user's background interests.

For the next step, we are going to focus on the evaluation of active academic visit recommendation application (*AAVRA*) usecase using some of the evaluation metrics defined in Neth et al. (2011).

Table 4.5: A comparative study of scalability on query time for three different strategies.

Number of Datasets	Strategy 1 (ms)	Strategy 2 (ms)	Strategy 3 (ms)
2	438	1156	63
4	735	2422	109
6	938	3407	187
8	1156	4656	203
10	1406	5734	266
12	1609	6782	344
14	1860	7688	391
16	4203	8781	500
18	4594	10750	454
20	4594	10750	485

4.4 Plug-in Status for Interests-based Selection

Based on the described theoretical framework introduced in Section 4.1.1, we implemented a interests-based selector⁶ based on the LarKC platform version 2.5 (the plugin is available in the LarKC plugin marketplace⁷). It can automatically calculate the weights for each interests from specified interests related triple sets and rank triples based on interests selection functions.

Interests-based selection based on multiple interests types introduced in Section 4.1.2 is implemented through plugin combination in workflows (as illustrated in Section 4.2.2). Namely, interests-based selector may be invoked several times in an application, such as the academic visit recommendation usecase.

Interests-based selector is designed to be used for user-centric semantic Web applications, such as general or domain specific personalized semantic search, personalized recommendation, etc. It focuses on the analysis of user interests related semantic data to produce a context for large scale semantic data processing, and it aims at solving the scalability barrier by providing personalized query and reasoning results on the basis of user expectations. We want to clarify that if user interests-related data cannot be acquired, interest-based selection cannot be useful.

4.5 Lessons Learned Through Interests-based Selection

Through experimental studies on interests-based selection, we can conclude that it is generally useful for processing large-scale semantic data when the user expects more relevant results that are close to his/her background interests.

For interests-based selection in Situation 2 (introduced in Section 4.3 above) the selection process can start once the user logs into an application. Hence, the time slot between the user logs in and starts querying may be of no use for other selection methods, but can be utilized for interests-based selection.

⁶Interests-based selector:

<<http://wiki.larkc.eu/LarkcPlugins/Interests-based-Selector>>

⁷LarKC plugin marketplace: <<http://www.larkc.eu/plugin-in-marketplace>>

Although the proposed method may cost more processing time on interests evaluation and interesting triples selection, the quality of the query results are improved significantly. Overall, the time for querying based on the selected triples can be dramatically reduced when compared to processing queries without such selection.

5 Ranking Retrieval Results by Semantic Similarity

This chapter summarizes VUA and MPG’s efforts to develop and evaluate a heuristic method for data selection on the basis of structural similarity. In these efforts, we use FactForge (see <http://factforge.net> and Bishop et al., 2010a) as the primary data repository from which RDF triples are to be selected. FactForge is a collection of some of the most central datasources in the Linked Open Data cloud. It hosts 11 datasets, including DBpedia, Freebase, Geonames, UMBEL, WordNet, the CIA World Factbook, MusicBrainz and others.¹ Several schemata used in the datasets are also loaded into FactForge, such as Dublin Core, SKOS and FOAF (see Bishop et al., 2010a). FactForge uses the OWLIM reasoner (Bishop et al., 2010b) to materialise all inferences that can be drawn from the datasets and their schemata. This results in some 10 billion retrievable statements, describing just over 400 million entities. Although FactForge is a subset of the entire Web of Data, it is currently one of the largest available subsets that is both closed under inference and queryable. Thus, as FactForge is one of the most generic use cases in LarKC we trust that our methodologies in this context are broad enough to be applicable to other use cases.

5.1 Introduction and Motivation

A collaborative project between VUA and MPG addresses the problem of how to select the correct answers to a query from partially incorrect answer sets that result from querying the Web of Data.

The current working hypothesis is that cognitively inspired similarity measures can be exploited to filter the correct answers from the full set of answers. These measures could be extremely simple and efficient when compared to those proposed in the literature, while still producing results that allow to *satisfice* (in the sense of Simon, 1956). Thus, we aim to use structural measures of the semantic similarity between RDF triples to provide a domain-independent method of *heuristic search* (Simon, 1990, p. 9). If this succeeds, it will provide “a computational method for satisficing” (Simon, 1996, p. 30) for the task of retrieving results by issuing semantic web queries.

We will validate this hypothesis by comparing the performance of our heuristic to human-level performance on a benchmark of queries to Linked Open Data resources. As a first step, we will compare heuristic and human performance by using the data of an individual human rater who rates the quality of RDF results as a benchmark (see Section 5.3). Given that our heuristic is extremely simple and efficient our aim is to achieve similar accuracy levels as a human evaluator using his or her semantic background knowledge.

A secondary contribution of this work is a freely available benchmark of queries (both natural language and SPARQL versions) plus gold standard human answers and 2000 SPARQL answers that are human-ranked for their quality.

Motivation The Web of Data has grown to contain tens of billions of statements. After the initial years of rapid growth, it is now becoming clear that—just like the

¹For details on data sets, see the W3C SWEO community project LinkingOpenData (LOD) at <http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>.

traditional Web—the Web of Data will always be a messy place, containing much correct, but also much incorrect data. Although there has been surprisingly little structured research on this topic both anecdotal evidence and developer’s experience shows that even the highest rated and most central datasets on the Web of Data such as DBPedia (Bizer et al., 2009) and Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008) contain factually incorrect and even nonsensical assertions. Consider the following results from some of the benchmark queries that we will discuss later, when executed against a combination of DBPedia, Geonames and Freebase:

- “AmeriCredit” is not an American car manufacturer. Instead, it is a financial company owned by General Motors to help customers finance their cars.
- “Richard Bass” is not one of the highest summits on the seven continents. Instead, he was the first mountaineer that climbed all of them.
- “Cosima” is not a Nobel Prize for Literature Laureate. Instead, it is a novel written by Grazia Deledda, who received the 1926 Nobel Prize for Literature.
- “Stig Anderson” was not one of the members of ABBA. Instead, he was the band’s manager.

These examples (which are only a few of many) illustrate *the central problem* that we tackle in this research effort:

Given a query to the Web of Data and the resulting answer set,
how to separate the correct from the incorrect answers?

For well over a decade now, influential cognitive scientists have been proposing the notion of *simple heuristics* (Gigerenzer, Todd, & the ABC research group, 1999): heuristics that are surprisingly simple (sometimes even seemingly naïve), but that on closer inspection perform very well on complex cognitive tasks (Gigerenzer & Goldstein, 1996; Czerlinski, Gigerenzer, & Goldstein, 1999). Their findings have shown convincingly that such simple heuristics are not only justified by gaining computational efficiency at the expense of output quality, but that such simple heuristics can even outperform more complex decision rules (see Gigerenzer, 2008; Gigerenzer & Brighton, 2009, for explanations).

The main goal of this research effort is to find out whether cognitively inspired heuristics can indeed be exploited to filter the correct answers from the noisy answers obtained when querying the Web of Data.

The overall benefit from this work is that such heuristics would allow us to efficiently select the most likely correct answers when querying the Web of Data. As an additional benefit such a selection heuristic could be tuned to favour either recall or precision (see Section 2.2 for definitions).

5.2 Background and Related Work

Over the past decade, the Semantic Web community has built and adopted a set of synthetic benchmarks to test storage, inference and query functionality. Some of the most well known benchmarks are the Lehigh LUBM benchmark, (Guo, Pan, & Heflin, 2005), the extended eLUBM benchmark (Ma et al., 2006) and the Berlin SPARQL

benchmark (Bizer & Schultz, 2009). (Additional benchmarks are described at www.w3.org/wiki/RdfStoreBenchmarking.) However, all these are *synthetic* datasets. There is a shortage of *realistic* benchmarks that provide realistic queries plus validated (“Gold Standard”) answers. The sample queries on the webpages of Linked Life Data (LinkedLifeData.com/sparql), FactForge (FactForge.net/sparql) are examples of such realistic queries, but they do not come with a validated set of Gold Standard answers.²

The topic of “ranking” query results has been studied since the early days of the Semantic Web, and is itself based on even longer lines of research in fields such as Information Retrieval. Our space is insufficient here to provide an extensive literature survey. Instead, we will discuss a few salient differences between our current approach and the literature:

Some of the literature on ranking is concerned with *relevance ranking*: determining which answers are relevant to an unstructured query in natural language, or relevant for a user based on their profile (see, e.g., He & Baker, 2010; Stojanovic, Studer, & Stojanovic, 2003; Anyanwu, Maduko, & Sheth, 2005; Hurtado, Poulouvasilis, & Wood, 2009). Although interesting and important, this work is not pertinent to our present concern, since we start with SPARQL queries (hence query-relevance is not an issue), and we do not consider user-profiles, but we are trying to recognise objectively true answers.

Another part of the literature is concerned with ranking answers by *importance*. Typically, this is done by a variety of pagerank-style analysis of the structure of the Semantic Web, trying to locate which resources are more important, more authoritative, more trustworthy, etc. (Ding et al., 2004; Anyanwu et al., 2005). Our approach differs from all this work in an important way: We do not do any *a priori* analysis of the structure of the large RDF graph that we are querying (a graph with billions of edges and hundreds of millions of nodes). Instead, we will only take the URI’s that are returned as a result of a query, and we compute some very simple local properties of these URI’s (namely the number of shared feature-value pairs).

Some of the literature on ranking deals with ranking different kinds of objects from what we consider: Thomas, Alani, Sleeman, and Brewster (2005); Alani, Brewster, and Shadbolt (2006); Tartir and Budak Arpinar (2007) and others rank ontologies, Swoogle ranks Semantic Web documents (Ding et al., 2004), Vu, Hauswirth, and Aberer (2005) and others rank services, etc. All these works have in common that they rely on fairly sophisticated analyses of the to-be-ranked-entities: internal structure of the ontologies, semantic descriptions of the functionality of the services, etc. Instead, we aim to rank only sets of atomic URIs. Although it might initially seem harder to rank such simple objects, since they come with very little structure to base the ranking on, our preliminary results (reported in Section 5.5) indicate that a simple analysis of very little information may be sufficient to obtain good ranking results, which is consistent with the predictions by the simple heuristics framework proposed by Gigerenzer et al. (1999). It will be interesting to investigate if such simple (or even: simplistic) analyses also yield good results when applied to more complex objects, such as ontologies or services, potentially replacing the more sophisticated ranking techniques found in the literature until now.

²See the ongoing Question-Answering challenge at <http://www.sc.cit-ec.uni-bielefeld.de/qald-1> for a related effort.

A work that is quite close in aim to ours is Lopez et al. (2009). Their “semantic similarity” is similar in spirit to ours: It tries to spot wrong answers through their large semantic distance to many of the other answers. However, the semantic distance in Lopez et al. (2009) is calculated as the distance in a shared ontology. We will use a much simpler method: Rather than using an ontology we will calculate distance simply as the number of shared feature-value pairs.

5.3 Benchmark Construction

Set of questions To measure the accuracy (precision and recall) of data retrieval we first need a set of questions and a set of correct answers. For an experiment investigating how humans retrieve and search for information in their memory, Neth, Schooler, Quesada, and Rieskamp (2009) have designed a set of general knowledge questions which all ask for enumerations of objects of a given type, or with a given property, such as “Name members of the pop band ABBA”, “Name Nobel laureates in literature since 1945”, etc. (See Appendix B.1 for a full list of all questions used.)

Gold Standard answers Neth et al. (2009) determined a set of correct answers for each question which participants had to search for in their memory. Particular care was given to the completeness of this Gold Standard, since people were asked to give as many items belonging to a category as they could.

SPARQL queries We have developed a set of 47 SPARQL queries, made to resemble the questions from Neth et al. (2009). For this translation, we used a number of well-known namespaces, such as DBPedia, Freebase, Geonames, Umbel, etc. As an example, the question about ABBA members translates to the SPARQL query shown in Figure 5.1.

SPARQL answers To complete this benchmark collection, we executed all of our queries against FactForge (FactForge.net). Running our 47 queries against FactForge (in the version of August 2010) resulted in 1896 answers. An example answer-set is shown in Figure 5.1. (See Appendix B.2 for a full list of all queries used.)

The entire resource (original questions, their SPARQL translations, the Gold Standard answers, as well as query-results against FactForge) will be made available online at <http://www.larkc.eu/resources/published-data-sources> (see also Buikstra, Neth, Schooler, Teije, & Harmelen, 2011).

5.4 Measuring Human Performance

In order to judge how good our heuristics will be at recognising correct answers, we first need to measure how good a human is at this task. To this end, a human rater (educated at university level) ranked all 1900 answers on a 5-point Likert scale, with 5 indicating that the rater was most confident that an answer was correct, and 1 indicating that the rater was confident that an answer was incorrect. (These human rankings will also be available from the aforementioned URL.)

```

SELECT DISTINCT ?member ?label
WHERE {
  ?member skos:subject dbp-cat:ABBA_members
  ?member rdfs:label ?label
  FILTER(lang(?label) = "en")
}

dbpedia:Agnetha_Fältskog    Agnetha Fältskogen
dbpedia:Agnetha_Fältskog    Agneta øase Fältskogen
dbpedia:Anni-Frid_Lyngstad  Anni-Frid Lyngstaden
dbpedia:Anni-Frid_Lyngstad  Frida Lyngstaden
dbpedia:Benny_Andersson     Benny Anderssonen
dbpedia:Björn_Ulvaeus       Björn Ulvaeusen
dbpedia:Ola_Brunkert        Ola Brunkerten
dbpedia:Stig_Anderson       Stig Andersonen

```

Figure 5.1: Example of a SPARQL query and corresponding FactForge answer-set.

By comparing the result of these ratings with our Gold Standard of correct answers we can assess whether the human rater was able to recognize correct answers with sufficiently high confidence. For this, we introduce the following notations:

Notation We use Q to indicate the query, with Q ranging from #1 to #47 in our benchmark collection:

- The set of Gold Standard answers to query Q is denoted by $G(Q)$.
- The set of retrieved answers to query number Q are denoted by $A(Q)$.
- The set of answers to query Q that were scored with a confidence ranking of T or higher is denoted as $A_T(Q)$, $T = 1, \dots, 5$.

Obviously, $A_1(Q) = A(Q)$ (all answers are included at confidence threshold $T = 1$), and the size of $A_T(Q)$ decreases with increasing T .

In our experiment, the size of $G(Q)$ is typically a few dozen items (since this is how the original queries were designed). The size of $A(Q)$ varies greatly from a dozen to several hundreds, showing that some answer sets contain many wrong results, i.e. $A(Q) \not\subseteq G(Q)$, for some Q . We will see below that also $G(Q) \not\subseteq A(Q)$ for some Q , i.e. FactForge is not complete for all of our queries.

To judge the performance of our human subject in recognising correct answers, we will plot the recall and precision of $A_T(Q)$ as a function of his confidence threshold T , where the correctness of the answers in $A_T(Q)$ is determined against $G(Q)$. The comparison of the SPARQL results in $A_T(Q)$ against the (natural language) elements in $G(Q)$ is done using the `rdfs:label` of the elements in $A_T(Q)$.

Example query As an illustration, Figure 5.2(a) shows the performance of our rater on query $Q = \#31$: “What are the highest mountains (peaks) of each continent”. At threshold level $T = 5$ (i.e., when aiming to select only correct answers for which the rater is most confident), the raters scores a precision of 1.0 but recognises

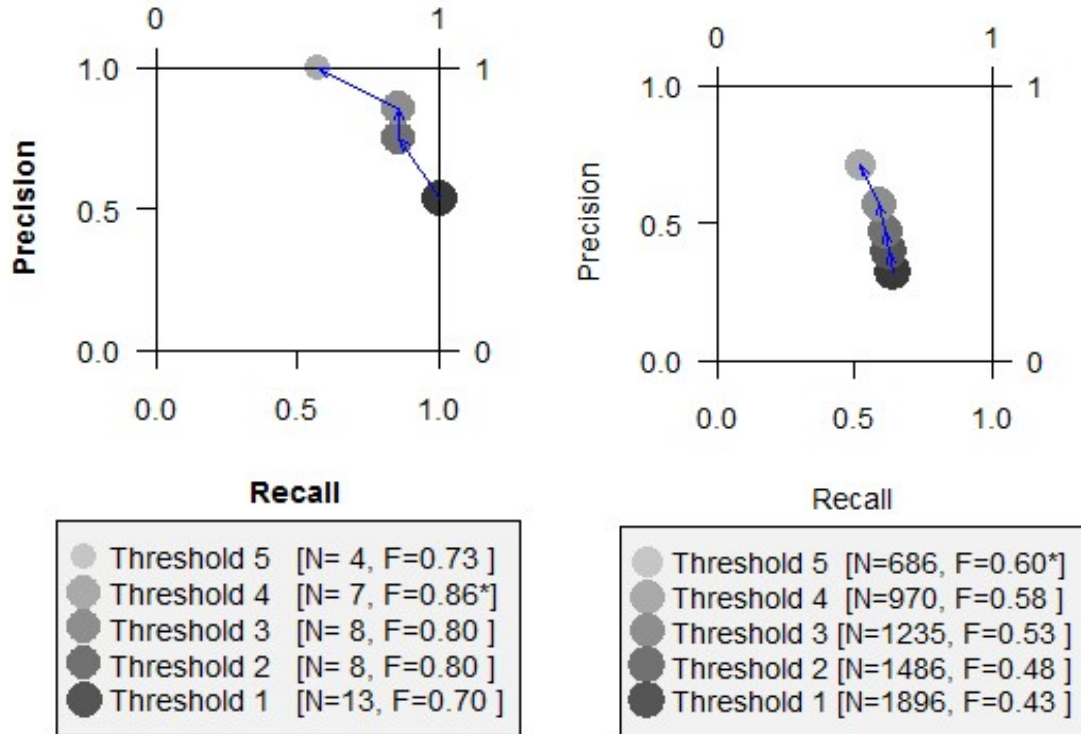


Figure 5.2: Performance of human rater: (a) on an example query and (b) accumulated over all queries.

only $N = 4$ out of the 7 summits, i.e., recall is only at 0.57. When including answers at lower confidence levels, the recall increases, finally reaching 1.0 at $T = 1$. This shows that FactForge does indeed contain all correct answers for this query, i.e., $G(\#31) \subset A(\#31)$. However, the increase in recall comes at the cost of also including some incorrect answers, with precision dropping to a final value of 0.5. The maximal performance (using the F_1 -measure to combine precision and recall) is $F=0.86$, and is reached at confidence threshold $T = 4$.

Accumulated results Figure 5.2(b) shows the recall and precision figures of our human subject accumulated over all 47 queries. This illustrates that even at $T = 1$ overall recall is only just above 0.6. This tells us that FactForge is indeed incomplete for our particular set of queries, and it is simply impossible for any rater (human or machine) to do any better on this set of queries.

Figure 5.2(b) shows a near perfect performance by our human rater. When increasing the value of the confidence level T , the precision of $A_T(G)$ increases from 0.25 to 0.75 while paying almost no penalty in decreasing recall (which is dropping merely from 0.6 to 0.5). In other words, when stepping up the confidence level from T to $T + 1$, the sets $A_{T+1}(G)$ have lost some of the wrong answers that were still in $A_T(G)$ while maintaining most of the correct answers in $A_T(G)$. Or, stated informally: our rater is actually rather good at recognizing the correct answers from among $A_T(G)$. In terms of the graph in Figure 5.2(b), a perfect performer would result in a vertical plot (increasing precision without a corresponding loss of recall). The human rater comes close to that perfect plot. Consequently, the highest score ($F=0.60$) is obtained at confidence threshold $T = 5$.

5.5 A Selection Heuristic Based on Semantic Similarity

Simple heuristics Biological cognitive agents—be they human or non-human animals—have to perform semantic classification tasks on a daily basis: Given a set of possible alternatives, which ones are “correct” or “acceptable”, e.g., when distinguishing edible from inedible food items or deciding if another agent is a friend or foe. As early as 1969, Herbert Simon (Simon, 1969) noted that this selectivity is based on rules of thumb, or heuristics, which cut problems down to manageable size. The basic idea is that the world contains an abundance of information and the solution is not necessarily to integrate as much information as possible, but rather to select useful information and use that for reasoning (see also Simon, 1956).

Within cognitive science, there is an ongoing debate between decision theorists on whether these rules of thumb have mainly positive or negative consequences on the quality of human decisions. In the so-called *heuristics and biases* tradition, researchers emphasize that people frequently make faulty decisions in situations under uncertainty, when compared against the normative standards of utility maximization and probability theory (Tversky & Kahneman, 1974). On the other side of the debate, researchers within the *simple heuristics* program emphasize that simple rules that ignore a lot of information and require little integration of information often perform as well as (and frequently outperform) much more complex optimization algorithms (Gigerenzer et al., 1999; Gigerenzer, 2000).

Such *simple heuristics* which in given situations outperform classical methods while using substantially less information and computational resources, are also relevant outside the area of cognitive science. Irrespective of whether the heuristics people (or animals) use lead to good or to bad decisions, we can ask whether simple heuristics can be tailored for decision making in complex computational settings.

The relevance to the semantic web lies in that these algorithms could improve quality of data selection by improving accuracy without damaging speed. Ultimately, we hope to retrieve high-quality results at lower computational costs, but without the guaranteeing the optimality or completeness of results. Or, to use Simon’s term, we wish to satisfice our data selection result (Simon, 1956).

Semantic similarity in structured data If one thinks about a query as defining a target region in some semantic space, one would expect the results of the query to be clustered in that target region. That is, the results that are most similar to each other are most likely to be close to the center of the targeted region. It seems reasonable to assume the further away a result is from the center of this estimated target region, the more likely it is to have been included in the results due to error.

Two classical approaches to formalizing similarity are featural approaches, epitomized in Tversky’s contrast model (Tversky, 1977), and spatial models (Shepard, 1957). In spatial models, similarity is defined as the distance in a defined metric space between two items. Spatial models have predefined spaces and each item is a separate point in the space, making symmetrical similarity natural. Tversky’s contrast model focuses on features shared between items and features not shared between items.

A computational definition of similarity Tversky’s similarity model based on shared features fits very naturally with the RDF data model (as described in Section 2.1): An “item” is a URI s_1 , a “feature” is a triple $\langle s, p, o \rangle$, and two features are

shared between two items s_1 and s_2 if they have the form $\langle s_1, p, o \rangle$ and $\langle s_2, p, o \rangle$. For example, two objects share a feature if they both have a `skos:subject` property with object `dbp-cat:ABBA_members`. Formally this can be expressed as follows:

Definition 1 *the similarity $S(s_1, s_2)$ between two resources s_1 and s_2 in a graph G is defined as:*

$$S(s_1, s_2, G) = ||\{(p, o) | \langle s_1, p, o \rangle \in G \text{ and } \langle s_2, p, o \rangle \in G\}||$$

i.e., similarity is defined as the number of feature-value pairs in G that are shared between s_1 and s_2 . This looks even simpler as a schematic SPARQL query:

```
SELECT COUNT(?p)
WHERE {<s1> ?p ?q
      <s2> ?p ?q}
```

where `<s1>` and `<s2>` must be replaced by specific URI's.

This similarity measure can now be used to define a heuristic confidence estimate for query-answers:

Definition 2 *The confidence estimate $C(a, Q, G)$ for an answer $a \in A(Q)$ to a query Q over a graph G is defined as*

$$C(a, Q, G) = \Sigma_{a' \in A(Q)} S(a, a', G)$$

i.e., the confidence estimate of an answer a is defined as the aggregate similarity of a to every other answer a' .

Alternative definitions A number of variations on this definition are possible. Instead of counting the total number of shared features $\langle s_1, p, o \rangle$, we could calculate the *fraction* of shared features, as suggested in (Tversky, 1977). Because of the fairly uniform arity of the nodes in RDF graphs, we would not expect this to make much difference.

We could also have used the weaker definition of only counting shared properties p without demanding that they have the same values: $\langle s_1, p, - \rangle$ and $\langle s_2, p, - \rangle$. For example: two objects are similar if they both have a `dbp-prop:manufacturer` property, even if that property has different values. However, due to the weak nature of many of the features (e.g. `rdf:type`, `skos:subject`) we expect that this will generate too high similarity ratings.

More reasonable would be to include shared *inverse* features $\langle o, p, s_1 \rangle$ and $\langle o, p, s_2 \rangle$. This would account for inverse modelling in the RDF graph, for example using the predicate `is-manufacturer` instead of `manufactured-by`. Such inverse properties are rare in FactForge, but this would be worth further investigation.

5.6 Heuristic Performance

We are now in a position to measure how good the heuristic from Def. 2 is at selecting the correct answers. In analogy to our analysis of our human ratings above (see Section 5.4) we will first present these results for an example query, before accumulating results over all queries.

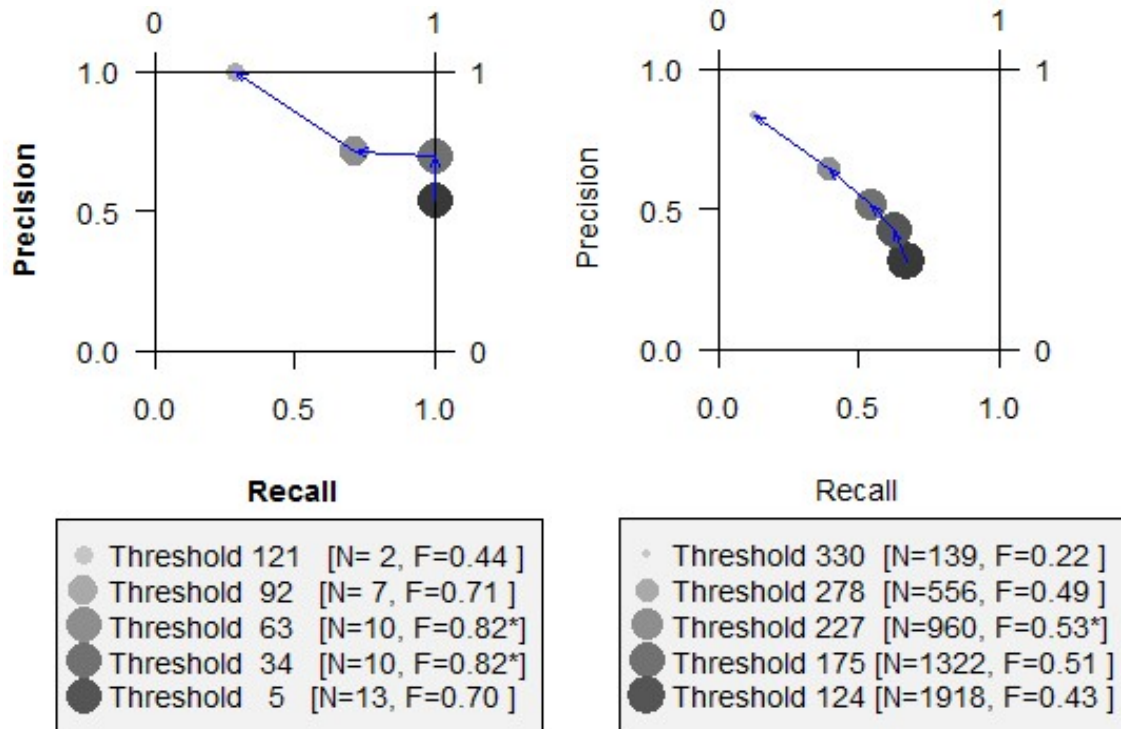


Figure 5.3: Performance of the similarity heuristic: (a) on the example query and (b) accumulated results.

Example query Figure 5.2(a) showed the recall and precision figures of our human subject for the “seven summits” query. Figure 5.3 presents the performance of our similarity based confidence estimate on the same query. To use the same evaluation procedure as for the human rater (in Section 5.4), we divide for each query Q the interval $[\min_{a \in A(Q)} C(a, Q, G), \max_{a \in A(Q)} C(a, Q, G)]$ uniformly into five equal steps.

Trivially, the heuristic performance at the lowest confidence level equals that of the human performance at the lowest confidence level, as both sets still include exactly the same exemplars. Excluding exemplars results in an inevitable decrease in recall and a corresponding increase in precision. However, the slope of this decrease seems a little shallower for the heuristic, i.e., a given decrease in recall does not result in the same increase in precision as for the human rater. At the maximal level of precision (1.0) the heuristic only retains 2 out of 7 correct exemplars (0.28), whereas the human rater still managed to achieve a recall of 0.57 (i.e., 4 out of 7 exemplars).

Accumulated results What happens when we pool our heuristic results over all queries? Figure 5.2(b) showed the recall and precision figures of the human subject accumulated over all 47 queries. Figure 5.3(b) shows the corresponding performance of the similarity based confidence estimate (i.e., its performance accumulated over all 47 queries).

5.7 Conclusion and Outlook

The conclusion from this comparison is mixed: On the one hand, the human recall-precision curve lies everywhere above the heuristic curve, which means that the human rater outperformed the heuristic on all levels of precision/recall. On the other hand,

the highest heuristic F-score ($F = 0.53$) is within 10% of the highest human F-score ($F = 0.60$), suggesting that the heuristic performs relatively well. This is all the more surprising since the heuristic — in stark contrast to the human rater — relies on no background knowledge whatsoever, but only counts the number of shared feature-value pairs between the members of the answer set. This lends some support to the conclusion that very simple “fast and frugal” heuristics that prune the set of retrieved results on the basis of structural similarity can achieve high performance levels.

In this chapter, we suggested that a simple cognitively inspired similarity heuristic could be used to select the correct answers from among the query results obtained from querying Linked Open Data sets. Our proposed heuristic is extremely simple and efficient when compared to those discussed in the literature. Consequently, we would count it as a success if—as we hypothesize—it yielded a similar quality of retrieval results as a human rater.

Our efforts towards this goal differ from previous work in the following important ways:

- Firstly, we do not assume any expensive prior pagerank-style analysis of the structure of the entire data-space we are querying. Instead, we do a simple count over information local to the URI’s that are returned as query results. In the worst case, the cost of our analysis is limited to retrieving all properties of all elements in the answer set, a cost which is negligible when compared to the large scale network analysis needed for most ranking approaches. Also, any such *a priori* large scale link-analysis is likely to be outdated when it is needed for querying. The information that our proposed heuristic needs is so simple that it can be retrieved at query time, and is hence always up-to-date.
- Secondly, we do not require any background knowledge or inferencing. No reference is made to any ontological background knowledge, it is not necessary to relate any answers to a shared ontology, and no inference of any kind is performed by our simple heuristic. All we require is to retrieve the properties of the elements in the answer set. These are simple atomic queries of the form $\langle s, -, - \rangle$, that are efficiently supported by the index-structures of any triple-store.

An important secondary contribution of this work will be the construction of a benchmark of general knowledge queries (expressed in natural language and SPARQL) with corresponding human ratings that can serve as Gold Standard answers. Almost 2,000 answers to such queries have been manually ranked on their quality. This collection will be made available at <http://www.larkc.eu/resources/published-data-sources> for other researchers as an important tool in benchmarking their query strategies over the Web of Data.

6 Conclusions

This document reports the results of several efforts to evaluate data retrieval and selection methods within the LarKC project.

A continuous thread between the current efforts and previous deliverables (e.g., D2.3.3 Quesada et al., 2011) is the question whether a large data set should first be reduced (e.g., by subsetting) or, alternatively, whether we should retrieve large amounts of data and then iteratively select the most relevant parts of it. The former question is the one discussed in Chapter 3, while the latter is discussed in Chapter 5 and also in the previous version of this deliverable D2.7.2. The basic question behind the idea of subsetting is the following: When using the full graph results as a benchmark, can we identify subsets that allow retrieval of the similar results on the basis of reduced computations?

Chapter 3 reported how selection methods can be used to answer this question. We varied the relevant parameters of the three methods (baseline, Random Indexing and Spreading Activation) in order to see how they influence the results and conclude whether selection can be used to reduce the problem space for reasoning.

For both baseline and Random Indexing described in Section 3.2.1, the increase of parameter *depth* which affects the size of the virtual document, resulted in increased recall for the two datasets of different type. While the overall execution time for the evaluation of the SPARQL queries was reduced, it is questionable whether, generally speaking, the same method would be practical for billions of triples repositories. The reason is that generating the index and semantic spaces for Random Indexing is quite expensive, and the search that is required to be performed in order to locate the specific relevant subset is also such that despite its parallelisation options (Damjanovic et al., 2011, discussed in D2.5.3.), the time reduction becomes negligible. This conclusion can not be generalised, as for a specific kind of the SPARQL queries the selection can be effective, e.g., for the queries which have URIs at the subject position.

With regard to Spreading Activation the results reported in Section 3.2.2 have shown that any desired level of recall can be reached by increasing the subset, but not without a corresponding cost of speed. In other words, for the specific kinds of SPARQL queries the selection methods could produce the relevant subset quite quickly. Similarly to the RI and baseline, these are the queries which have an URI at the subject position. If the query has a variable at the subject position, none of the selection methods could produce a relevant subset quickly, if at all.

In addition, what was necessary for producing results with the RI and baseline methods when the query required reasoning over triples (OWLIM *ruleset = rdfs*), is that the TBox had to be loaded together with the subset. Increasing parameter depth from 1 to 2, and including TBox significantly increased the overall recall for the dataset that required *rdfs* reasoning (DBpedia) from 23.8% to 63.8%.

Another selection method based on statistical semantics was explored in the LarKC deliverable 2.3.3 (Quesada et al., 2011). In the research reported there, we assigned weights to a large RDF graph (by representing the keywords of SPARQL queries as a vector) and defined a subset by a threshold value on those weights. In experiments using FactForge we were unable to improve performance through subsetting, irrespective of the effectiveness of graph weighting. Subsetting proved to be impractical regardless of whether it was applied before or after materialization (see D2.3.3, Part 2, for details). This failure encouraged us to explore ways of using statistical semantics to

increase the quality of retrieved results, rather than to reduce computational efforts (and results of this work were reported in Chapter 5).

In Chapter 4, LarKC partner WICI reformulated their theoretic framework of interests-based selection based on previous work in D2.3.1 and D2.3.2 (Quesada et al., 2009, 2010). In the current document, we provided a quantitative evaluation of interests ranking and interests-based selection on a single interest type. This method and relevant algorithms are suitable when interests are of the same type, such as research interests from publications-related semantic data. For complex scenarios, which are related to multiple interests types, we developed a qualitative evaluation approach among different types of interests. We investigated the proposed methods in the scenario of active academic visit recommendation system based on real time semantic data from the Semantic Web Dog Food and Twitter (through transformation). Evaluations of interests-based selection through concrete usecases show that the proposed method is user-centric and could be very useful when the background information of a user is available. In addition, it can help users get what they really need within a comparatively less time when compared to traditional semantic search methods.

In Chapter 5, LarKC partners VUA and MPG explored a different solution to the data selection problem. Rather than restricting the selection set *a priori* to its most relevant parts, their heuristic method ranks retrieval results and prunes the retrieval set by a measure of semantic similarity. As dissimilar exemplars are dropped a reduced recall corresponds to an increase in precision that mimics the performance of a human rater. Although the resulting trade-off curve of the heuristic does not outperform the human it is remarkable how well the method performs in the absence of any sophisticated background knowledge.

In conclusion, it is no surprise that the analyses reported in this document cannot eschew the inevitable trade-offs in information retrieval, e.g., between precision and recall or that an increase in speed may come at a cost to recall. However, our analysis has shown that for a specific kind of SPARQL queries, this trade-off can fade away and that our selection methods can yield 100% recall while at the same time reducing the cost for retrieval. In addition, LarKC has proven to be a valuable platform to explore and negotiate these trade-offs. Finally, our methods for ranking retrieved items based on user-interests or structural similarity is proven to be a promising selection strategy.

References

- Alani, H., Brewster, C., & Shadbolt, N. (2006). Ranking ontologies with aktiverank. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, & L. Aroyo (Eds.), *The semantic web - iswc 2006* (Vol. 4273, p. 1-15). Springer Berlin / Heidelberg.
- Anyanwu, K., Maduko, A., & Sheth, A. (2005). Semrank: ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on world wide web* (pp. 117–127). New York, NY, USA: ACM.
- Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., & Velkov, R. (2010a). FactForge: A fast track to the web of data. *Semantic Web Journal*.
- Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., & Velkov, R. (2010b). OWLIM: A family of scalable semantic repositories. *Semantic Web Journal*.
- Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009). DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3), 154–165.
- Bizer, C., & Schultz, A. (2009). The Berlin SPARQL benchmark. *International Journal On Semantic Web and Information Systems*, 5(1), 1-24.
- Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. 1247–1250.
- Buikstra, A., Neth, H., Schooler, L., Teije, A. ten, & Harmelen, F. van. (2011, July 16). Ranking query results from linked open data using a simple cognitive heuristic. In *Workshop on discovering meaning on the go in large heterogeneous data 2011 (LHD-11)*. Barcelona, Spain: Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11). (Held at the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11) July 16, 2011 Barcelona, Spain)
- Croft, B., Metzler, D., & Strohman, T. (2009). *Search Engines: Information Retrieval in Practice* (1 ed.). Addison Wesley.
- Cunningham, H., Roberts, A., Li, Y., Aswani, N., Momchev, V., Kiryakov, A., Quesada, J., & Schooler, L. J. (2008). *Selection: Experimental apparatus. LarKC project deliverable 2.1.2* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Cunningham, H., Roberts, A., Li, Y., Kiryakov, A., Schooler, L. J., Quesada, J., Neth, H., Valle, E. D., Braga, D., & Zhong, N. (2008). *Selection and retrieval methods. LarKC project deliverable 2.1.1* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Czerlinski, J., Gigerenzer, G., & Goldstein, D. (1999). How good are simple heuristics. In G. Gigerenzer, P. M. Todd, & the ABC research group (Eds.), *Simple heuristics that make us smart* (pp. 97–118). New York, NY, USA: Oxford University Press.
- Damljanovic, D., Lupu, M., Peikov, I., Assel, M., Cheptsov, A., Quesada, J., Carlsson, M., Engstrom, G., & Andersson, B. (2011). *Geometrical semantics components (v3). LarKC project deliverable 2.5.3* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., & Sachs, J. (2004). Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the thirteenth acm international conference on information and*

- knowledge management* (pp. 652–659). New York, NY, USA: ACM.
- Fensel, D., & Harmelen, F. van. (2007). Unifying reasoning and search to web scale. *IEEE Internet Computing*, 94–96.
- Fensel, D., Harmelen, F. van, Andersson, B., Brennan, P., Cunningham, H., Valle, E. D., Fischer, F., Huang, Z., Kiryakov, A., Lee, T. K., Schooler, L. J., Tresp, V., Wesner, S., Witbrock, M., & Zhong, N. (2008). Towards LarKC: A platform for Web-scale reasoning. In *Proceedings of the IEEE international conference on semantic computing (ICSC 2008), August 4-7, 2008, santa clara, ca, usa*. Los Alamitos, CA, USA: IEEE Computer Society Press.
- Frakes, W., & Baeza-Yates, R. (1992). Information retrieval: Data structures and algorithms.
- Gigerenzer, G. (2000). *Adaptive thinking: Rationality in the real world*. New York, NY: Oxford University Press.
- Gigerenzer, G. (2008). Why heuristics work. *Perspectives on Psychological Science*, 3(1), 20–29.
- Gigerenzer, G., & Brighton, H. (2009). Homo heuristicus: Why biased minds make better inferences. *Topics in Cognitive Science*, 1(1), 107–143.
- Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103, 650–669.
- Gigerenzer, G., Todd, P. M., & the ABC research group. (1999). *Simple heuristics that make us smart*. New York, NY: Oxford University Press.
- Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. *Advances in Information Retrieval*, 345–359.
- Grinberg, M., Stefanov, H., Stefanov, K., & Peikov, I. (2011). *Spreading activation components (v3). LarKC project deliverable 2.4.3* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Guo, Y., Pan, Z., & Heflin, J. (2005). LUBM: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3), 158 - 182. (Selected Papers from the International Semantic Web Conference, 2004 - ISWC, 2004)
- He, X., & Baker, M. (2010, November). xhRank: Ranking entities on the Semantic Web. In *9th international semantic web conference (ISWC2010)*.
- Hurtado, C., Poulouvasilis, A., & Wood, P. (2009). Ranking approximate answers to semantic web queries. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvnen, R. Mizoguchi, E. Oren, M. Sabou, & E. Simperl (Eds.), *The semantic web: Research and applications* (Vol. 5554, p. 263-277). Springer Berlin / Heidelberg. (10.1007/978-3-642-02121-3_22)
- Klyne, G., & Carrol, J. J. (2004, 10 February). *Resource description framework (RDF): Concepts and abstract syntax* (Tech. Rep.). W3C Recommendation. (Available at <http://www.w3.org/TR/rdf-concepts/>.)
- Lopez, V., Nikolov, A., Fernandez, M., Sabou, M., Uren, V., & Motta, E. (2009). Merging and ranking answers in the semantic web: The wisdom of crowds. In A. Gomez-Prez, Y. Yu, & Y. Ding (Eds.), *The semantic web* (Vol. 5926, pp. 135–152). Springer Berlin / Heidelberg. (10.1007/978-3-642-10871-6_10)
- Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., & Liu, S. (2006). Towards a complete owl ontology benchmark. In Y. Sure & J. Domingue (Eds.), *The semantic web: Research and applications* (Vol. 4011, p. 125-139). Springer Berlin / Heidelberg.

- Neth, H., Quesada, J., Schooler, L. J., Wong, J. T., Buikstra, A., Teije, A. ten, Harmelen, F. van, Damljanovic, D., Peikov, I., & Zeng, Y. (2011). *Experimental data analysis, combinatorics plan. LarKC project deliverable 2.7.2* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Neth, H., Schooler, L. J., Quesada, J., & Rieskamp, J. (2009). *Analysis of human search strategies. LarKC project deliverable 4.2.2* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Peikov, I., Grinberg, M., Haltakov, V., Stefanov, H., Kiryakov, A., Ognyanoff, D., & Velkov, R. (2010). *Spreading activation components (v2). LarKC project deliverable 2.4.2* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Prud'hommeaux, E., & Seaborne, A. (2008, 15 January). *SPARQL query language for RDF* (Tech. Rep.). W3C Recommendation. (Available at <http://www.w3.org/TR/rdf-sparql-query/>.)
- Quesada, J., Otte, S., Brandao-Vidal, R., Schooler, L., Wong, J. T., Neth, H., Damljanovic, D., & Peikov, I. (2011). *Cognitive memories components (v3): Subsetting by statistical semantics and user interests. LarKC project deliverable 2.3.3* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Quesada, J., Zeng, Y., Brandao, R., Schooler, L. J., Otte, S., Zeng, Y., Wang, Y., Huang, Z., Zhong, N., & Damljanovic, D. (2010). *Cognitive memories components (v2): Subsetting by statistical semantics and user interests. LarKC project deliverable 2.3.2* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Quesada, J., Zeng, Y., Schooler, L. J., Zhou, H., Zhong, N., Qin, Y., Lu, S., Yao, Y., & Gao, Y. (2009). *Cognitive memories components (v1). LarKC project deliverable 2.3.1* (Tech. Rep.). The Large Knowledge Collider (LarKC).
- Shepard, R. N. (1957). Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space. *Psychometrika*, 22(4), 325-345.
- Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, 63(2), 129–138.
- Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: The MIT Press.
- Simon, H. A. (1990). Invariants of human behavior. *Annual Reviews in Psychology*, 41(1), 1–20.
- Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge, MA: The MIT Press.
- Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4), 35–43.
- Stojanovic, N., Studer, R., & Stojanovic, L. (2003). An approach for the ranking of query results in the semantic web. In *The semanticweb - iswc 2003* (Vol. 2870, p. 500-516). Springer Berlin / Heidelberg. (10.1007/978-3-540-39718-2_32)
- Tartir, S., & Budak Arpinar, I. (2007, September). Ontology evaluation and ranking using OntoQA. In *International conference on semantic computing (ICSC)* (pp. 185–192).
- Thomas, E., Alani, H., Sleeman, D., & Brewster, C. (2005). Searching and ranking ontologies on the semantic web. In *Workshop on ontology management: Searching, selection, ranking, and segmentation. 3rd k-cap 2005* (pp. 57–60).
- Todorova, P., Kiryakov, A., Ognyanoff, D., Peikov, I., Velkov, R., & Tashev, Z. (2009). *Spreading activation components (v1). LarKC project deliverable 2.4.1* (Tech. Rep.). The Large Knowledge Collider (LarKC).

- Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4), 327-352.
- Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157), 1124-31.
- Vu, L.-H., Hauswirth, M., & Aberer, K. (2005). QoS-based service selection and ranking with trust and reputation management. In *OTM Conferences* (Vol. 3760, pp. 466–483). Springer.
- Zeng, Y., Zhong, N., Wang, Y., Qin, Y., Huang, Z., Zhou, H., Yao, Y., & Harmelen, F. van. (2011a). User-centric query refinement and processing using granularity based strategies. *Knowledge and Information Systems*, 27(3).
- Zeng, Y., Zhou, E., Wang, Y., Ren, X., Qin, Y., Huang, Z., & Zhong, N. (2011b). Research interests : Their dynamics, structures and applications in unifying search and reasoning. *Journal of Intelligent Information Systems*, 37(1), 65–88.

A Queries used in Evaluation of Data Selection Methods

A.1 MusicBrainz queries

This appendix lists the MusicBrainz queries discussed in the evaluation of selection methods (see Section 3.1).

Query 1: Give me all soundtracks composed by John Williams.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX ar: <http://musicbrainz.org/ar/ar-1.0#>
SELECT DISTINCT ?album ?albumname
WHERE {
  ?album mm:releaseType mm:TypeSoundtrack .
  ?album ar:composer ?artist .
  ?artist dc:title 'John Williams' .
  ?album dc:title ?albumname .
}
```

Query 2: Which bands was Robbie Williams a member of?

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ar: <http://musicbrainz.org/ar/ar-1.0#>
SELECT DISTINCT ?band ?bandname
WHERE {
  ?artist dc:title 'Robbie Williams' .
  ?artist ar:memberOfBand ?bandinstance .
  ?bandinstance ar:toArtist ?band .
  ?band dc:title ?bandname .
}
```

Query 3: Which artists performed the song Over the Rainbow?

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
SELECT DISTINCT ?artist ?artistname
WHERE {
  ?track rdf:type mm:Track .
  ?track dc:title 'Over the Rainbow' .
  ?track dc:creator ?artist .
  ?artist dc:title ?artistname .
}
```

Query 4: Give me all siblings of Michael Jackson.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ar: <http://musicbrainz.org/ar/ar-1.0#>
SELECT DISTINCT ?artist1 ?artistname {
```

```
?artist1 dc:title ?artistname .
?artist2 dc:title 'Michael Jackson' .
?artist1 ar:isSibling ?artistinstance .
?artistinstance ar:toArtist ?artist2 .
}
```

Query 5: List live albums by the Beatles.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select *
WHERE {
  ?album rdf:type mm:Album .
  ?album mm:releaseType mm:TypeLive .
  ?album dc:creator ?artist .
  ?artist dc:title 'The Beatles' .
}
```

Query 6: List all jazz compilations.

```
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ct: <http://www.sc.cit-ec.uni-bielefeld.de/ontologies/musicbrainz#>
SELECT *
WHERE {
  ?album rdf:type mm:Album .
  ?album mm:releaseType mm:TypeCompilation .
  ?album ct:tag 'jazz' .
}
```

Query 7: What is the longest song by John Cage?

```
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?song ?songtitle
WHERE {
  ?song mm:duration ?duration .
  ?song dc:title ?songtitle .
  ?song dc:creator ?artist .
  ?artist dc:title 'John Cage' .
}
ORDER BY DESC(?duration)
LIMIT 1
```

Query 8: Which compilations does the song Waterloo by ABBA appear on?

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
PREFIX ar: <http://musicbrainz.org/ar/ar-1.0#>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
SELECT DISTINCT ?album ?albumtitle
WHERE {
  ?album dc:title ?albumtitle .
  ?album mm:releaseType mm:TypeCompilation .
  ?album dc:creator ?artist .
  ?artist dc:title 'ABBA' .
  ?track dc:title 'Waterloo' .
  ?album mm:trackList ?tracklist .
  ?tracklist ?seq_index ?track .
}
```

Query 9: When was Ludwig van Beethoven born?

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
SELECT ?birthday
WHERE {
  ?artist dc:title 'Ludwig van Beethoven' .
  ?artist mm:beginDate ?birthday .
}
```

Query 10: Give me all songs on the album Abbey Road.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?track ?title {
  ?album rdf:type mm:Album .
  ?album mm:releaseType mm:TypeAlbum .
  ?album dc:title 'Abbey Road' .
  ?album mm:trackList ?tracklist .
  ?tracklist ?seq_index ?track .
  ?track dc:title ?title .
}
```

Query 11: Which bands are called Queen?

```
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT COUNT(DISTINCT ?artist) {
  ?artist mm:artistType mm:TypeGroup .
  ?artist dc:title 'Queen' .
}
```

Query 12: Are there bootleg recordings by Pink Floyd?

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mm: <http://musicbrainz.org/mm/mm-2.1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select *
WHERE {
  ?album rdf:type mm:Album .
  ?album mm:releaseStatus mm:StatusBootleg .
  ?album dc:creator ?artist .
  ?artist dc:title 'Pink Floyd' .
}
```

A.2 Dbpedia queries

This appendix lists the Dbpedia queries discussed in the evaluation of selection methods (see Section 3.1).

Query 1: Who is the daughter of Bill Clinton married to?

```
PREFIX res: <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?uri
WHERE
{
  res:Bill_Clinton onto:child ?child .
  ?child dbpedia2:spouse ?uri .
}
```

Query 2: Find buildings located in Belgrade.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT distinct ?building
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
  ?building rdf:type onto:Building .
  ?building onto:location ?uri .
  filter (?uri=res:Belgrade)
}
```

Query 3: Find buildings located in Serbia.

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT distinct ?building
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
    ?building rdf:type onto:Building .
    ?building onto:location ?uri .
    filter (?uri=res:Serbia)
}
```

Query 4: Is Proinsulin a chemical compound?

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT * WHERE
{
    ?v rdf:type <http://dbpedia.org/class/yago/Compound114818238> .
    FILTER(?v = res:Proinsulin)
}
```

Query 5: List amides.

```
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX onto: <http://dbpedia.org/ontology/>
select ?x
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
    ?x rdf:type <http://dbpedia.org/class/yago/Amide114724264> .
}
```

Query 6: Who created Goofy?

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT DISTINCT ?uri
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
    res:Goofy prop:creator ?string .
    ?uri prop:name ?string .
    ?uri a ?type.
}
```

```
filter (?type=onto:Person)
}
```

Query 7: Who is the author of WikiLeaks?

```
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX onto: <http://dbpedia.org/ontology/>
SELECT ?uri
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
    res:WikiLeaks onto:author ?uri . ?uri a
    <http://dbpedia.org/class/yago/Writer110794014>
}
```

Query 8: Who is the owner of Universal Studios?

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?uri
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
    res:Universal_Studios onto:owner ?uri .
    ?uri a <http://dbpedia.org/class/yago/YagoLegalActor>
}
```

Query 9: Which monarchs of the United Kingdom were married to a German?

```
PREFIX yago: <http://dbpedia.org/class/yago/>
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX res: <http://dbpedia.org/resource/>
SELECT DISTINCT ?uri
from <http://www.ontotext.com/disable-sameAs>
WHERE{
?uri rdf:type <http://dbpedia.org/class/yago/Prince110472799> .
{ ?uri onto:spouse ?spouse . } UNION { ?spouse onto:spouse ?uri . }
?spouse onto:birthPlace res:Germany.}
```

Query 10: What are the official languages of the Philippines?

```
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT DISTINCT ?uri
from <http://www.ontotext.com/disable-sameAs>
WHERE
{
    res:Philippines onto:officialLanguage ?uri .
    ?uri a <http://dbpedia.org/class/yago/Language106282651> .
}
```

B Questions and Queries for Similarity-Based Data Selection

This appendix complements the methodology and results of our selection heuristic based on semantic similarity filters (Section 5.1).

Section B.1 lists the general knowledge questions that asked for the enumerations of objects of a given type, or with a given property (see Neth et al., 2009, for details).

Section B.2 lists the 47 SPARQL queries which were made to resemble these questions.

B.1 General Knowledge Questions

The following 47 questions are a subset of the 60 questions listed in Appendix A1 of LarKC deliverable 4.2.2 (Neth et al., 2009). (Note that a few questions in the original set were omitted, as those were not translatable into SPARQL queries of decent quality.)

1. Name the signs of the (Western) *zodiac*
2. Name the *planets* of our solar system
3. Name types of *grains* or *cereals* (genera)
4. Name species of *cats* (felidae)
5. Name genera of *pine trees* (Pinaceae)
6. Name *beer brands* with over \$1 billion in sales (2008)
7. Name *technology brands* with a value exceeding \$5 billion (2009)
8. Name *oil companies* with a value exceeding \$700 million (2009)
9. Name products in the *Microsoft Office Suite*
10. Name *Apple* products (hard- or software) (2009)
11. Name *US-American car manufacturers* (current)
12. Name *Japanese car manufacturers* (current)
13. Name *Italian pasta varieties*
14. Name *American countries* (nations)
15. Name *U.S.-American states* (current)
16. Name *capital cities* of U.S.-American states (current)
17. Name *continents* on earth
18. Name the *highest mountain* (peaks) of each continent
19. Name *countries* with a population exceeding 80 million (as of 2009)

20. Name the *German Bundesländer* (Federal States of the current Federal Republic of Germany)
21. Name the *capital cities* of German federal states (current)
22. Name *Nobel laureates in literature* (since 1945)
23. Name the feature *films by David Lynch*
24. Name the main characters of the first *Star Wars* movie (1977)
25. Name *James Bond movies* (titles)
26. Name the members of the pop band *The Beatles*
27. Name the musical *instruments of a symphonic orchestra*
28. Name the members of the pop band *ABBA*
29. Name the Olympian gods of the *Greek* mythology
30. Name the Olympian gods of the *Roman* mythology
31. Name the German chancellors (Federal Republic of Germany since 1949)
32. Name the US-American presidents (since 1945)
33. Name the foreign ministers of Germany (Federal Republic since 1951)
34. Name the current member states of the UN Security Council (2009)
35. Name Nobel Peace Prize laureates (since 1975)
36. Name Wimbledon winners for women's or men's singles (since 1980)
37. Name the teams of the 1st German football league *Bundesliga* (2008/09)
38. Name the sites of the modern Olympic Winter games (so far)
39. Name the sites of the modern Olympic Summer games (so far)
40. Name the teams of the *NBA* (National Basketball Association in 2009)
41. Name the teams of the 2nd German football league *2. Bundesliga* (2008/09)
42. Name the types of chess pieces
43. Name the genera of European broadleaf trees
44. Name meat products offered by McDonalds
45. Name the operas by Wolfgang Amadeus Mozart
46. Name existing *brands* worth over \$20 billion (in 2009)
47. Name existing *fashion brands* worth over \$1 billion (in 2009)

B.2 SPARQL Queries

B.2.1 Namespaces

All SPARQL queries below use the following collection of prefixes:

```

PREFIX dbp-cat: <http://dbpedia.org/resource/Category>
PREFIX dbp-ont: <http://dbpedia.org/ontology/>
PREFIX dbp-prop: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX fb: <http://rdf.freebase.com/ns/>
PREFIX ff: <http://factforge.net/>
PREFIX fu: <http://www4.wiwiss.fu-berlin.de/factbook/resource/>
PREFIX fuf: <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
PREFIX geo-ont: <http://www.geonames.org/ontology#>
PREFIX opencyc-en: <http://sw.opencyc.org/2008/06/10/concept/en/>
PREFIX owl: <http://www.ontotext.com/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX yago: <http://mpii.de/yago/resource/>

```

B.2.2 Queries

The following queries were constructed to correspond to the general knowledge questions listed in Appendix B.1.

Query: 1_WESTERN_ZODIAC

```

SELECT DISTINCT ?Constellation
WHERE {
    ?Constellation dbp-prop:family dbpedia:Zodiac .
    ?Constellation rdfs:label ?label .

    FILTER (lang(?label) = "en")
}

```

QUERY: 4_PLANETS

```

SELECT DISTINCT ?Planet ?preferredlabel
WHERE {
    ?Planet rdf:type dbpedia:Interplanetary .
    ?Planet fb:astronomy.star_system_body.star_system dbpedia:Solar_System .
    ?Planet ff:preferredLabel ?preferredlabel.
}

```

QUERY: 5_CEREALS

```
SELECT DISTINCT ?grainOrCereal
WHERE {
  ?grainOrCereal fb:type.object.type
  fb:user.thadguidry.default_domain.cereal_grain .
}
```

QUERY: 6_CATS

```
SELECT DISTINCT ?cat ?label
WHERE {
  ?cat rdfs:label ?label .
  ?cat rdf:type dbpedia:Cat .
}
```

```
FILTER(LANG(?label) = "en")
}
```

QUERY: 7_TREES

```
SELECT DISTINCT ?genus
WHERE {
  ?tree dbp-ont:family dbpedia:Pinaceae.
  ?tree dbp-ont:genus ?genus
}
```

QUERY: 8_BEER

```
SELECT DISTINCT ?company
WHERE {
  ?company rdf:type dbp-ont:Company .
  ?company dbp-ont:netIncome ?income .
  ?company ?somerelation ?beer .
}
```

```
FILTER regex(?beer, "beer", "i") .
FILTER (xsd:double(str(?income)) > 1000000000) .
}
```

QUERY: 9_TECHNOLOGY

```
SELECT DISTINCT ?label
WHERE {
  ?Company ff:preferredLabel ?label .
}
```

```
?Company rdf:type opencyc-en:Business .
?Company dbp-prop:products ?Product .
?Product rdf:type umbel-sc:ComputerCode .
?Company dbp-ont:netIncome ?income .
```

```
FILTER (xsd:double(str(?income)) > 5000000000) .
}
```

QUERY: 10_OIL

```
SELECT DISTINCT ?label
WHERE {
  ?company rdfs:label ?label .
  ?company rdf:type dbp-ont:Company .
  ?company dbp-ont:equity ?eq .
  ?company skos:subject ?cat .
  ?cat skos:broader dbp-cat:Oil_companies .
```

```
FILTER (xsd:double(str(?eq)) > 700000000) .
FILTER ( lang(?label) = "en") .
}
```

QUERY: 13_OFFICE

```
SELECT DISTINCT ?Product
WHERE {
  ?Product skos:subject ?MSO .
  ?Product rdfs:label ?label .
  ?MSO skos:prefLabel "Microsoft Office"@en
}
```

QUERY: 14_APPLE

```
SELECT DISTINCT ?product
WHERE {
  {
    dbpedia:HPod dbp-prop:manufacturer ?Apple .
    ?product dbp-prop:manufacturer ?Apple .
  } UNION {
    dbpedia:IMix dbp-ont:developer ?Apple .
    ?product dbp-ont:developer ?Apple .
  }
  ?product owl:hasPageRank ?pagerank .
  ?product rdfs:label ?label.
```

```
FILTER(LANG(?label) = "en")
}
```

```
QUERY: 15_US_CARS
```

```
SELECT DISTINCT ?label
WHERE {
  { ?company dbp-ont:foundationPlace ?place }
  UNION
  { ?company dbp-ont:location ?place }
  UNION
  { ?company dbp-ont:locationCity ?place }

  ?place geo-ont:parentFeature dbpedia:United_States .

  { ?company dbp-ont:product dbpedia:Automobile }
  UNION
  { ?company dbp-ont:product dbpedia:Car }
  UNION
  { ?company skos:subject dbp-cat:Motor_vehicle_companies }

  ?company rdfs:label ?label .
  Filter(lang(?label) = "en")
}
```

```
QUERY: 17_JAPANESE_CARS
```

```
SELECT DISTINCT ?company
WHERE {
  { ?company dbp-ont:foundationPlace ?place }
  UNION
  { ?company dbp-ont:location ?place }
  UNION
  { ?company dbp-ont:locationCity ?place }

  ?place geo-ont:parentFeature dbpedia:Japan .

  { ?company dbp-ont:product dbpedia:Automobile }
  UNION { ?company dbp-ont:product dbpedia:Car }
}
```

```
QUERY: 20_PASTA
```

```
PREFIX umbel-sc: <http://umbel.org/umbel/sc/>
```

```
SELECT DISTINCT ?pasta
WHERE {
  ?pastas rdfs:label "Pasta"@en .
        ?pasta skos:subject ?pastas .
}
```

QUERY: 24_AMERICAN_COUNTRIES

```
SELECT DISTINCT ?label
WHERE {
  ?country rdf:type dbp-ont:Country ; rdfs:label ?label .
  { ?narrower skos:broader dbp-cat:American_countries }
UNION
  { [?narrower skos:broader] skos:broader dbp-cat:American_countries }

  ?country skos:subject ?narrower.

FILTER(LANG(?label) = "en")
}
```

QUERY: 28_US_STATES

```
SELECT DISTINCT ?state ?label
WHERE {
  ?state rdfs:label ?label .
  ?state a dbpedia:U.S._state .

FILTER(lang(?label) = "en" || lang(?label) = "") .
}
```

QUERY: 29_US_STATE_CAPITALS

```
SELECT DISTINCT ?capital ?label
WHERE {
  ?capital rdfs:label ?label .
  ?state a dbpedia:U.S._state .
  ?state dbp-ont:capital ?capital .

FILTER(lang(?label) = "en" || lang(?label) = "") .
}
```

QUERY: 30_CONTINENTS

```
SELECT DISTINCT ?continent ?label
```

```
WHERE {
  ?continent rdfs:label ?label .
  ?continent skos:subject dbp-cat:Continents .

FILTER(lang(?label) = "en" || lang(?label) = "") .
}
```

QUERY: 31_MOUNTAINS

```
SELECT DISTINCT ?continent ?label
WHERE {
  ?continent rdfs:label ?label .
  ?continent skos:subject dbp-cat:Seven_Summits.

FILTER(lang(?label) = "en" || lang(?label) = "") .
}
```

QUERY: 32_POPULOUS_COUNTRIES

```
SELECT DISTINCT ?country ?label
WHERE {
  ?country rdfs:label ?label .
  ?country dbp-prop:populationCensus ?pop.
  FILTER (xsd:double(?pop) > 80000000).

FILTER(lang(?label) = "en" ) .
}
```

Query: 33_bundeslander

```
SELECT DISTINCT ?bundesland ?label
WHERE {
  ?bundesland rdfs:label ?label .
  ?bundesland skos:subject dbp-cat:States_of_Germany .

FILTER(lang(?label) = "de" ) .
}
```

QUERY: 34_GERMAN_CAPITALS

```
SELECT DISTINCT ?capital ?label
WHERE {
  ?bundesland skos:subject dbp-cat:States_of_Germany .
  ?bundesland dbp-prop:capital ?capital .
}
```

```
?capital rdfs:label ?label .
```

```
FILTER(lang(?label) = "de").  
}
```

```
QUERY: 35_LITERATURE
```

```
SELECT DISTINCT ?laureate ?label  
WHERE {  
    ?laureate skos:subject dbp-cat:Nobel_laureates_in_Literature .  
    ?laureate rdfs:label ?label .
```

```
FILTER(lang(?label) = "en").  
}
```

```
QUERY: 36_DAVID_LYNCH
```

```
SELECT DISTINCT ?film ?label  
WHERE {  
    ?film dbp-ont:director dbpedia:David_Lynch .  
    ?film rdfs:label ?label .
```

```
FILTER(lang(?label) = "en").  
}
```

```
QUERY: 37_STAR_WARS
```

```
SELECT DISTINCT ?character ?label  
WHERE {  
    ?character skos:subject dbp-cat:Star_Wars_characters .  
    ?character rdfs:label ?label .
```

```
FILTER(lang(?label) = "en").  
}
```

```
QUERY: 38_JAMES_BOND
```

```
SELECT DISTINCT ?film ?label  
WHERE {  
    ?film skos:subject dbp-cat:James_Bond_films .  
    ?film rdfs:label ?label .
```

```
FILTER(lang(?label) = "en").  
}
```

QUERY: 39_BEATLES

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:The_Beatles_members .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 40_SYMPHONIC_ORCHESTRA

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member dbp-prop:wikilink dbpedia:Orchestra .
    ?member rdfs:subClassOf dbpedia:Musical_instrument .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 41_ABBA

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:ABBA_members .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 42_GREEK_GODS

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:Greek_gods .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 43_ROMAN_GODS

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:Roman_gods .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 44_GERMAN_CHANCELLORS

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:Chancellors_of_Germany .
    ?member dbp-ont:activeYearsStartDate ?date .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
FILTER (?date > "1949-01-01"^^xsd:date)
}
```

QUERY: 45_US_PRESIDENTS_1945

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:Presidents_of_the_United_States .
    ?member dbp-ont:activeYearsStartDate ?date .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
FILTER (?date > "1945-01-01"^^xsd:date)
}
```

Query: 46_german_foreign_ministers

```
SELECT DISTINCT ?member ?label
WHERE {
    ?member skos:subject dbp-cat:Foreign_Ministers_of_Germany .
    ?member dbp-ont:activeYearsStartDate ?date .
    ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
FILTER (?date > "1951-01-01"^^xsd:date)
}
```

QUERY: 47_UN_SECURITY_COUNCIL

```
SELECT DISTINCT ?member ?label
WHERE {dbpedia:Permanent_5
      fb:base.unitednations.united_nations_body.members ?overmember .
      ?overmember
      fb:base.unitednations.united_nations_body_membership.member ?member .
      ?member rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 48_PEACE

```
SELECT DISTINCT ?laureate ?label
WHERE {
      ?laureate skos:subject dbp-cat:Nobel_Peace_Prize_laureates.
      ?laureate rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 49_WIMBLEDON

```
SELECT DISTINCT ?player ?label
WHERE {
      ?player skos:subject dbp-cat:Wimbledon_champions .
      ?player fb:tennis.tennis_tournament_champion.tennis_titles ?title .
      ?title fb:tennis.tennis_tournament_championship.tournament
            dbpedia:Lawn_tennis_championships.
      ?title fb:tennis.tennis_tournament_championship.event_type ?event_type .
      ?event_type fb:type.object.name ?event_type_name.
      ?title fb:tennis.tennis_tournament_championship.year ?year .
      ?player rdfs:label ?label .

FILTER regex(str(?event_type_name), "ingle") .
FILTER (xsd:double(?year) > 1979) .
FILTER(lang(?label) = "en").

}
```

QUERY: 50_BUNDESLIGA

```
SELECT DISTINCT ?team ?label
```

```
WHERE {
    ?team dbp-ont:league ?league .
    ?league rdfs:label ?league_label .
    ?team rdfs:label ?label .

FILTER regex(str(?league_label), "bundesliga","i") .
FILTER regex(str(?league_label), "1","i") .
FILTER(lang(?label) = "en") .
}
```

QUERY: 51_OLYMPIC_WINTER

```
SELECT DISTINCT ?city ?label
WHERE {
    ?city fb:olympics.olympic_host_city.olympics_hosted ?olympics .
    ?city rdfs:label ?label .
    ?olympics rdfs:label ?olympics_label .

FILTER(lang(?label) = "en") .
FILTER regex(str(?olympics_label), "winter", "i")
}
```

QUERY: 52_OLYMPIC_SUMMER

```
SELECT DISTINCT ?city ?label
WHERE {
    ?city fb:olympics.olympic_host_city.olympics_hosted ?olympics .
    ?city rdfs:label ?label .
    ?olympics rdfs:label ?olympics_label .

FILTER(lang(?label) = "en") .
FILTER regex(str(?olympics_label), "summer", "i")
}
```

QUERY: 53_NBA_TEAMS

```
SELECT DISTINCT ?team ?label
WHERE {
    ?team skos:subject dbp-cat:National_Basketball_Association_teams .
    ?team rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 54_BUNDESLIGA2

```
SELECT DISTINCT ?team ?label
WHERE {
    ?team dbp-ont:league ?league .
    ?league rdfs:label ?league_label .
    ?team rdfs:label ?label .

FILTER regex(str(?league_label), "bundesliga","i") .
FILTER regex(str(?league_label), "2","i") .
FILTER(lang(?label) = "en") .
}
```

QUERY: 55_CHESS_PIECES

```
SELECT DISTINCT ?piece ?label
WHERE {
    ?piece skos:subject dbp-cat:Chess_pieces.
    ?piece rdfs:label ?label .

FILTER(lang(?label) = "en").
}
```

QUERY: 56_BROADLEAVES

```
SELECT DISTINCT ?tree ?label
WHERE {
    ?tree a dbp-ont:Eukaryote .
    ?tree rdfs:subClassOf dbpedia:Green_plant .
    ?tree rdfs:subClassOf opencyc-en:Tree_ThePlant .

    ?tree rdfs:label ?label .
FILTER(lang(?label) = "en")
}
```

QUERY: 57_MCDONALDS

```
Select ?meat ?label
WHERE {
    ?meat skos:subject
        <http://dbpedia.org/resource/Category:McDonald%27s_foods>.
    ?meat rdfs:label ?label.
FILTER(Lang(?label) = "en")
}
```

QUERY: 58_MOZART_OPERAS

```
SELECT DISTINCT ?opera ?label
WHERE {
    ?opera skos:subject dbp-cat:Operas_by_Wolfgang_Amadeus_Mozart .
    ?opera rdfs:label ?label .

FILTER(lang(?label) = "en")
}
```

QUERY: 59_20B_BRANDS

```
SELECT DISTINCT ?comp ?label
WHERE {
    ?comp dbp-ont:equity ?equity .
    ?comp rdfs:label ?label .
FILTER(lang(?label) = "en") .
FILTER (xsd:double(str(?equity )) > 2e10) .
}
```

QUERY: 60_1M_CLOTHES

```
SELECT DISTINCT ?comp ?label
WHERE {
    ?comp skos:subject dbp-cat:Clothing_brands .
    ?comp dbp-prop:revenue ?revenue.
    ?comp rdfs:label ?label .
FILTER(lang(?label) = "en") .
FILTER (xsd:double(str(?revenue)) > 1e9) .
}
```

C List of Abbreviations

HTTP	Hyper Text Transfer Protocol
IR	Information Retrieval
LarKC	Large Knowledge Collider
LDSR	Linked Data Semantic Repository
LLD	Linked Life Data
LOD	Linked Open Data
LSA	Latent Semantic Analysis
MAP	Mean Average Precision
N3	Notation 3
RDF	Resource Description Framework
RI	Random Indexing
SA	Spreading Activation
SPARQL	SPARQL Protocol And RDF Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WP	Work Package (within LarKC)