A preliminary investigation into SPARQL query complexity and federation in Bio2RDF

Carlos Buil-Aranda[1], Martín Ugarte[1], Marcelo Arenas[1], and Michel Dumontier[2]

[1] Department of Computer Science, Pontificia Universidad Católica, Chile
{cbuil, marenas}@ing.puc.cl
martinugarte@puc.cl
[2] Stanford Center for Biomedical Informatics Research
Stanford University, Stanford, CA, USA
michel.dumontier@stanford.edu

Abstract When users query a SPARQL endpoint, they normally face an empty text box in which they have to write the desired queries. This obstructs the process of obtaining the data they want, since users rarely have any assistance in querying a (possibly huge) RDF database. In this paper we report a deep analysis of the server log files that record the queries that users send to the SPARQL endpoints, focusing in the Bio2RDF cluster. This log analysis reveals the large number of repeated queries that users submit, and how they pursue a trial and error process by adding and removing operations from the submitted queries to obtain the desired results. We also show how users try to connect to other RDF datasets in the Linked Open Data cloud. Our results offer insight into the interaction between users and a schema-light RDF dataset, and secondly, suggest improvements to SPARQL server optimizations in terms of optimization and results caching.

1 Introduction

Querying Semantic Web data is a difficult task. Normally the databases are publicly available and they can be accessed via a web service called SPARQL endpoint. This service is made available through a web application with a single text box, allowing users to enter arbitrary SPARQL queries. That text box generally contains an example query, which is the only assistance that users have for accessing the data stored in the RDF database. That default query may point them to some useful data but most probably the results obtained will be meaningless for specific tasks. This situation is more problematic if instead of a single RDF database, users wants to access a cluster of databases like in the Bio2RDF [9,3] project. Bio2RDF is an open source project that provides over 30 biomedical datasets as Linked Data. Each dataset is made available for download and is available for querying in a dataset-specific SPARQL endpoint. In this case users not only face the difficulty of accessing the data in every dataset separately, but they also face the difficulty of combining results from several databases.

In this paper we propose to analyze the server log files from the datasets in the Bio2RDF project. This log analysis shows that there is a large amount of repeated queries, and that users follow a trial and error process, varying the complexity of the queries for obtaining the results they want. We also show how users try to connect to

external RDF datasets in the Linked Open Data cloud, and we try to understand and explain the users' intentions when they query a SPARQL endpoint. The log analysis we show in this paper is driven by two goals: first to help users in obtaining useful results from a semi-unknown RDF database and second, improve the performance of SPARQL servers by looking in detail how users access them.

Related Work There have been several attempts to obtain useful research results from SPARQL endpoints query logs. Most of them have been published in the workshop series Usage Analysis and the Web of Data (USEWOD) [5,4,6,7], which is the leading initiative for encouraging research in SPARQL endpoints log analysis. These research works vary from analyzing the usage frequency of the main SPARQL operators [2], characterizing machine agents [15] or identifying browsing and query patterns by using Description Logics ontologies [10]. Further research works include a more in-detail analysis of the FILTER operator usage [1], a log analysis towards caching and prefetching SPARQL query results for improving performance [11], and a work that used the USEDOWD dataset to differentiate queries generated by software applications from those generated by users [16]. Other works outside the USEWOD workshop include an analysis of the SPARQL queries submitted to DBpedia [14], statistics about the access to RDF datasets in the Linked Data Cloud in 2010 [13] and a method to detect errors or weaknesses within ontologies used for Linked Data population based on statistics and network visualizations [12].

2 Log Processing

We analyzed the log files generated by the Bio2RDF servers maintained by the Dumontier Lab over an 18 month period (from May 12th 2013 until September 28th 2014). These logs included every valid HTTP GET/POST request that was received by each Bio2RDF endpoint. The total amount of requests received was 115,119,540. We first parsed these log files generating, for each request, a tuple containing the user's IP address (used as user ID), the time and date in which the HTTP request was received, the string in the HTTP request unquoted, the user agent which submitted the request, the Bio2RDF server targeted, the HTTP response code, and the size of that response. Out of the 115,119,540 valid HTTP requests received, 90,938,804 of them corresponded to SPARQL queries. This is natural since the studied servers also serve websites and further services. The queries were characterized as SELECT, ASK, CONSTRUCT and DESCRIBE queries. The next step was to remove duplicate queries. We parsed the tuples generated in the previous file and we obtained that (surprisingly) only 6,538,280 queries were unique, having thus a total of 84,400,524 repetitions. As our log study is intended to analyze the users' behavior, repetitions were only counted under the same user, meaning that the same query issued by two different users is not considered as a repetition. Next, we transformed the 6,538,280 queries into an algebraic representation, using the SPARQL Syntax Expressions[1] format from Apache Jena. This transformation facilitated a detailed analysis of the queries. For the generation of the SSE expressions

[1] https://jena.apache.org/documentation/notes/sse.html

we used the Ruby library Ruby-RDF[2], which was unable to parse 174,011 of the queries (possibly due to syntax errors). For 1,289,134 of the remaining 6,364,269, we were unable to generate the corresponding algebraic expression, in some cases due to syntax errors that were not captured by the SSE parser (e.g. not using <> for URIs). Finally this process generated a total of 4,901,124 unique queries for analysis.

3 Complexity Analysis

To gain insight into what users formulate against Bio2RDF SPARQL endpoints, we tabulated combinations of operators mentioned in the queries. We first decomposed each query into its operators, number of triple patterns, and expressions used in FILTER clauses.

Our results, summarized in Table 1, show that the most submitted query pattern is a SELECT query with a single triple pattern, scoring more than 820,000 submissions out of the 4,901,124 unique queries. The second most submitted query pattern is a CONSTRUCT query with a single triple pattern and a FILTER expression (representing more than 540,000 unique queries) followed by a DESCRIBE query (which represent about 440,000 unique queries). The three patterns mentioned above characterize a 36.8% of the analyzed queries. This shows that the general usage of SPARQL is very basic and the patterns in the body of the queries are rather simple, but users know about the language given they use distinct query forms (SELECT, CONSTRUCT and DESCRIBE). We can conclude that there is a fair understanding of the query language, but the datasets are not known by the users. Hence, they issue basic queries to gain some insight on how data is structured.

Figure 1 represents the previous situation in more detail, showing how the decreasing amount of queries submitted is directly related with the increasing amount of operators in the queries. The less common query patterns contained several operators (e.g. 3 OPTs, 3 joins and 7 triples), which is by no means surprising. We can also see how only the first 20 query patterns represent a 90% of the queries submitted, which can be an interesting fact to consider when optimizing a SPARQL endpoint. Figure 1 contains the first 50 query patterns, with labels for some of them. In total there are almost 1,000 different query pattern types. The data used to generate Figure 1 and a more detailed figure is available at https://plot.ly/%7Ecbuil/31.

4 Iterative Analysis

Next, we examined the behavior of users in terms of how they increased/decreased the amount of operators and triple patterns in SPARQL queries over time. We hypothesized that if an initial query returned a large set of results, users might then refine the query with additional SPARQL operators to reduce the result size. In contrast, if a user obtains too few results, she might generalize the query by removing some operations in order to increase the number of results. To evaluate this we defined a query complexity measure which assigns a weight of 1 to each operator and each triple pattern. For

[2] https://github.com/ruby-rdf/sparql

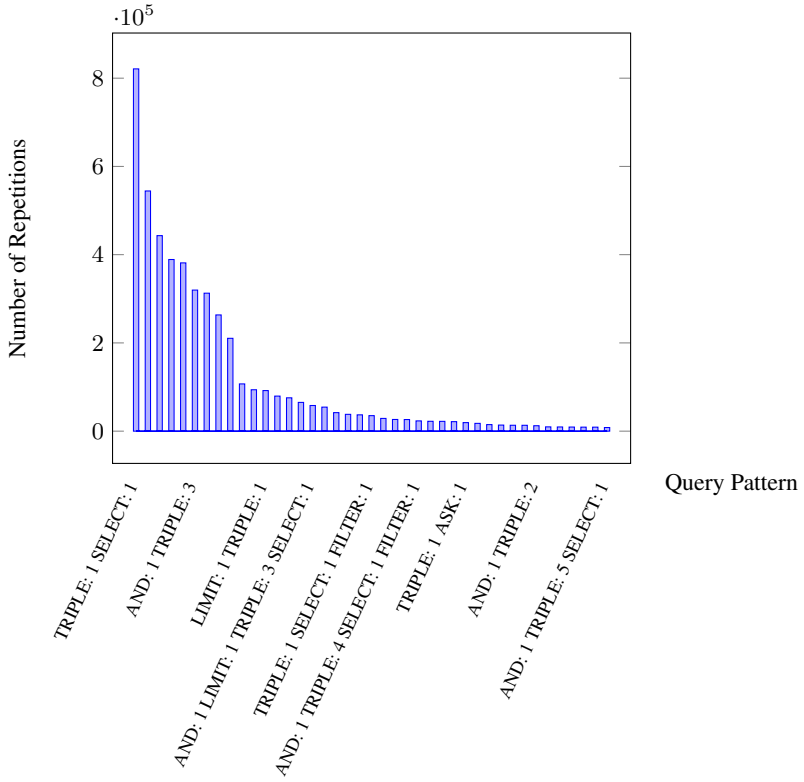

Figure 1. Query patterns ordered by number of repetitions. The most common pattern is SELECT v WHERE t, being v a set of variables (or the symbol $*$) and t a triple pattern.

instance, a query of the form DESCRIBE u (where u is a URI) has complexity 1 because of the DESCRIBE operator, while a SELECT query joining 3 triple patterns has a complexity of 5; 1 for the SELECT operator, 1 for the join (bgp) and 1 for each triple pattern.

We measured the HTTP request response size as a proxy for the size of the result set returned to the user. Then, we computed the number of consecutive complexity increases/decreases (referred to as a streak) for each user. Table 4 shows the amount of complexity-increasing streaks we found in the Bio2RDF log files. The first column shows the length of each streak. For example length 2 means that a user issued three queries, being the second more complex than the first and the third more complex than de second. The second column shows the times we found streaks of that length. The third column show how many of the streaks stopped when the result size of the last query was larger than the result size of the second last query. Conversely, the fourth column shows the same but when the result size of the last query was smaller than the result size of the second last query. The last two columns are intended to understand the intention of a user when he issues a streak of increasing queries: did the user stop

Table 1. Query Pattern Repetitions. The most repeated query pattern has a SELECT and a single triple, followed by a CONSTRUCT query and a DESCRIBE query.

Number of Query Pattern Repetitions	Query Pattern
821046	TRIPLE PATTERNS: 1, SELECT: 1
544341	TRIPLE PATTERNS: 1, CONSTRUCT: 1,FILTER: 1
443125	DESCRIBE 1
389011	LIMIT: 1, TRIPLE PATTERNS: 1, SELECT: 1
381351	TRIPLE PATTERNS: 1
319708	AND: 1, TRIPLE PATTERNS: 3

adding operators when he got less results? or was it when he got more results than before? Conversely, Table 4 shows the same statistics for decreasing streaks.

As opposed to what we originally hypothesis, the results show that users who add new operators in their query workflow will generally obtain more results than in their previous query, as depicted in Table 4. Similarly, users who remove operators stop removing them generally when the result size is smaller than that of the previous query (4). Our interpretation of the statistics is that users will add operators once they understand the dataset structure, and hence they will issue correct queries that will return more results. On the other side, when users issue a query with less operators they might be looking to understand new portions of the data, but in general they obtain less results due to a limited knowledge of the data structure. The only situation in which this is not the case is when users issue decreasing streaks of size 3. Here users stop removing operators once they get larger result. A preliminary interpretation of this could be that users who know the dataset are obtaining to few results, and hence they start removing restrictive parts of the query (like joins or filters) in order to obtain more information. We believe this is something worth investigating in more detail. In summary, the results show that when users add operations they generally are obtaining more results, and when users remove operations they are obtaining less results. This might have an interesting impact in terms of server optimization, as a static analysis on two consecutive queries and the answer to the first of them could already give insight on what will be the size of the result to the second query. Of course this requires a more refined definition of increasing/decreasing streaks, which is left as future work.

Table 2. Amount of *increasing* streaks, streak sizes and relation with amount of results from the previous query results.

Streak length	# of increasing streaks	Ended with larger result	Ended with smaller result
1	286,684	259,148	26,083
2	21,903	21,334	464
3	157	60	88

Table 3. Amount of *decreasing* streaks, streak sizes and relation with amount of results from the previous query results.

Streak length	# of decreasing streaks	Ended with larger result	Ended with smaller result
1	283,474	23,007	258,655
2	6,905	1,602	5,187
3	10,005	9,854	129

5 Dataset Federation Analysis

Finally, we used our analysis platform to examine which datasets were being queried both inside and outside the Bio2RDF network. To do so, we first tabulated the queries sent to specific Bio2RDF endpoints (Figure 2), as well as the queries that used the SERVICE keyword to query SPARQL endpoints that were outside of the Bio2RDF network (Figure 3). Our results show that i) the top 5 Bio2RDF datasets are (in decreasing number of queries posed): Pubmed (with more than 11,000 SERVICE calls), Gene (with almost 1,000 SERVICE calls), Pharmgkb, Drugbank and Bioportal (recently added to the Bio2RDF network); and ii) the top 5 SPARQL endpoints used to complement Bio2RDF queries are: the Gene Expression Atlas (with more than 500 SERVICE requests), Beta Uniprot (a development version of the Uniprot dataset), DBpedia, the Chemical Biology Group and Reactome: a knowledge base of biologic pathways and processes. It is important to notice that 4 of these datasets are funded by the European Bioinformatics Institute. More detailed figures and the data used to generate Figures 2 and 3 can be found at https://plot.ly/%7Ecbuil/77 and https://plot.ly/%7Ecbuil/78 respectively. The results we present in this section show that SPARQL 1.1 federation features are being used to connect to a surprisingly large number of endpoints. In total, there were 5,470 SERVICE calls in the final log files processed, 4,462 of them were directed to Bio2RDF server while 1008 were directed to other endpoints in the LOD Cloud.

6 Conclusions

In this paper we performed an analysis of the queries received at the Bio2RDF servers. This analysis showed first that the amount of repeated SPARQL queries received by these servers is huge (about a 7% of queries are unique), which of course can be used to optimize how servers are caching previously computed answers. Once the duplicate queries were removed, we found that around 50% of the queries contain just a single triple pattern, and that triple pattern is generally under of a SELECT or a CONSTRUCT. Furthermore, 20 query patterns represent 90% of the queries received by the servers, and this can have a high impact on terms of how servers should be optimized to answer queries. It is also interesting to notice that the second most used SPARQL query is DESCRIBE, which is marked as an *"Informative"* query form in the SPARQL 1.1 recommendation document (i.e. its implementation is not mandatory). This indicates that a large portion of the users are trying to understand the shape of the data before querying for information.

A more in detail analysis of the query patterns showed that a significant number of users add operators once they understand the data structure, and hence they obtain more

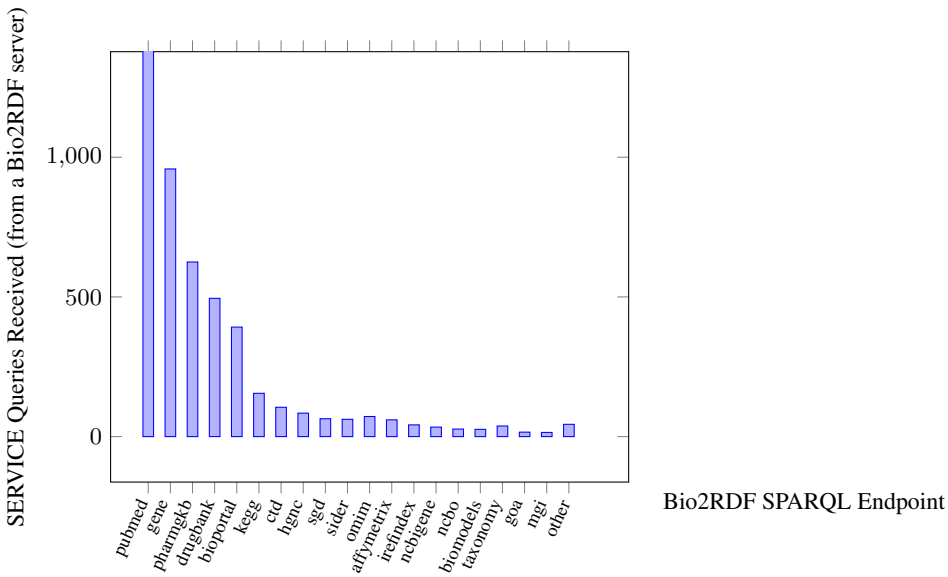

Figure 2. Bio2RDF servers queried using SPARQL Federation. The most Bio2RDF endpoint queried, by large, is Pubmed (http://pubmed.bio2rdf.org/sparql).

results. On the contrary, users remove operators looking for new portions of the data, and their lack of knowledge on the dataset leads to less results. Finally, a brief analysis of the SPARQL Federation queries showed a surprising amount of users that try to link the Bio2RDF data in one endpoint to other Bio2RDF datasets or to other datasets in the LOD cloud using the SERVICE keyword.

Limitations of the experiments. The experiments performed in this work present several limitations. First of all, we did not analyze the URIs in the SPARQL queries and the result sizes may be related to these URIs. It is important no notice that URIs in the queries may exist in the dataset or not, affecting positively or negatively to the queries result sizes and thus to our results. Similarly we did not analyze the effect of the LIMIT solution modifier not the effect of FILTERs, affecting as well to the query's result sizes. However, our results can still provide a useful insight of what Bio2RDF users want to obtain when querying the endopints.

Future work. This work is just a starting point for a more complete and detailed analysis of the Bio2RDF users and queries. A first next step is to overcome the experiments limitations to produce more accurate statistics about the use of Bio2RDF datasets. Once we overcome the limitations, our results can also provide a base line in which we can assess the capability of users to generate more complex queries with guided query tools (e.g. SPARQLED[3], YASGUI[4], etc). The statistics gathered about query patterns and

[3] http://sindice.com/sparqled/
[4] http://yasgui.org/

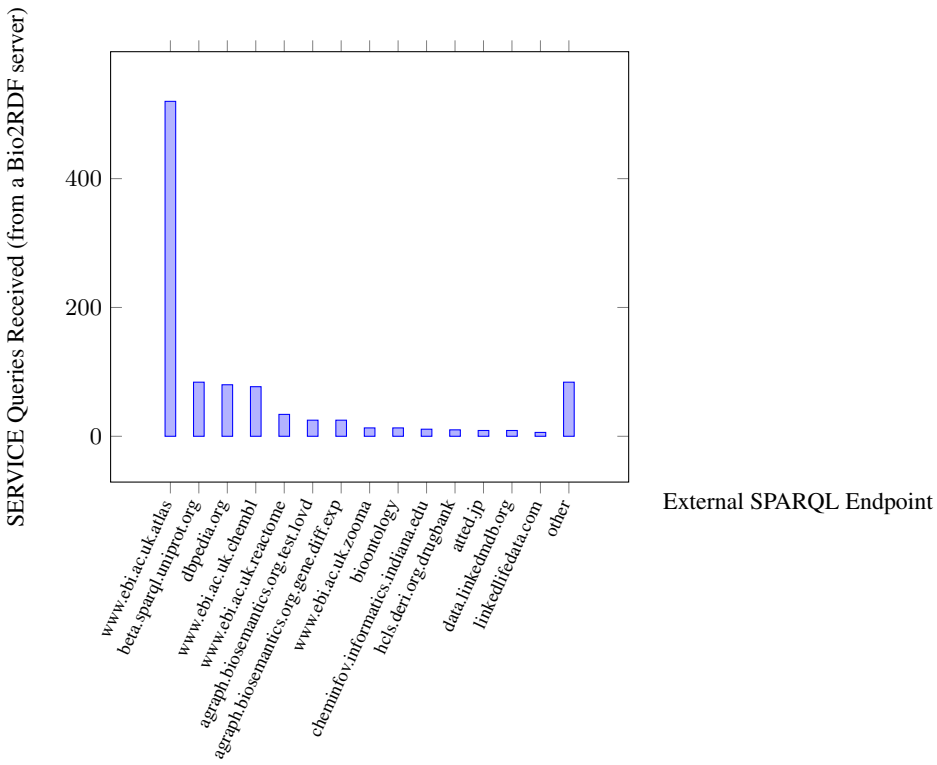

Figure 3. External servers to Bio2RDF queried, using SPARQL Federation. The most queried external endpoint are those from the European Bioinformatics Institute. That suggest that the researchers from that institute try to link the Bio2RDF data with theirs. Endpoints with less than 6 SERVICE queries are grouped in the "Other".

complexity may be of great help for these type of applications in order to help formulate effective SPARQL queries. Another well know problem in the Linked Data community is that SPARQL endpoints suffer from a performance problem due to the many requests received [8]. As mentioned before, our results be largely used to address this problem by guiding the optimization of software and result caching. Finally, our analysis only targeted the Bio2RDF endpoints. Our final goal would be to generalize our results to more SPARQL endpoints in the LOD Cloud. It is worth noting that our framework is not particular to Bio2RDF and can be applied to arbitrary (clusters of) SPARQL endpoints.

7 Acknowledgments

Carlos Buil-Aranda has been supported by the CONICYT/FONDECYT project 3130617 and by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004. Marcelo Arenas and Martín Ugarte have been supported also by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004

References

1. S. Aljaloud, M. Luczak-Rösch, T. Chown, and N. Gibbins. Get All, Filter Details - On the Use of Regular Expressions in SPARQL queries. In *Proceedings of the ESWC2014 workshop on Usage Analysis and the Web of Data (USEWOD 2014)*, 2014.
2. M. Arias, J. D. Fernández, M. A. Martínez-Prieto, and P. de la Fuente. An empirical study of real-world sparql queries. *CoRR*, abs/1103.5043, 2011.
3. F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *J. of Biomedical Informatics*, 41(5):706–716, Oct. 2008.
4. B. Berendt, V. Hollink, K. M. Markus Luczak-Rösch, and D. Vallet. 2nd international workshop on usage analysis and the web of data. in 21st eswc. In *In 21st International World Wide Web Conference (WWW2012)*.
5. B. Berendt, V. Hollink, K. M. Markus Luczak-Rösch, and D. Vallet. 1st international workshop on usage analysis and the web of data. In *In 20th International World Wide Web Conference (WWW2011)*, 2011.
6. B. Berendt, V. Hollink, K. M. Markus Luczak-Rösch, and D. Vallet. 3rd international workshop on usage analysis and the web of data. in 10th eswc. In *In 10th ESWC Semantics and Big Data, Montpellier, France.*, 2013.
7. B. Berendt, V. Hollink, K. M. Markus Luczak-Rösch, and D. Vallet. 4th international workshop on usage analysis and the web of data. in 11th eswc. In *In 11th ESWC Semantics and Big Data.*, 2014.
8. C. Buil-Aranda, A. Hogan, J. Umbrich, and P. Vandenbussche. SPARQL web-querying infrastructure: Ready for action? In *ISWC*, pages 277–293, 2013.
9. A. Callahan, J. Cruz-Toledo, P. Ansell, and M. Dumontier. Bio2rdf release 2: Improved coverage, interoperability and provenance of life science linked data. In *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, pages 200–212, 2013.
10. J. Hoxha, M. Junghans, and S. Agarwal. Enabling semantic analysis of user browsing patterns in the web of data. *CoRR*, abs/1204.2713, 2012.
11. J. Lorey and F. Naumann. Caching and prefetching strategies for sparql queries. In *Proceedings of the 3rd International Workshop on Usage Analysis and the Web of Data (USEWOD)*, Montpellier, France, 0 2013. Best Workshop Paper.
12. M. Luczak-Rösch and M. Bischoff. Statistical analysis of web of data usage. In *Joint Workshop on Knowledge Evolution and Ontology Dynamics (EvoDyn2011), CEUR WS*, 2014.
13. K. Mller, M. Hausenblas, R. Cyganiak, and S. Handschuh. Learning from linked open data usage: Patterns and metrics. In *Web Science*, 2010.
14. F. Picalausa and S. Vansummeren. What are real sparql queries like? In *Proceedings of the International Workshop on Semantic Web Information Management*, SWIM '11, pages 7:1–7:6, New York, NY, USA, 2011. ACM.
15. M. Raghuveer. Characterizing machine agent behavior through sparql query mining. *CoRR*.
16. L. Rietveld and R. Hoekstra. Man vs. Machine Dierences in SPARQL Queries. In *Proceedings of the ESWC2014 workshop on Usage Analysis and the Web of Data (USEWOD 2014)*, 2014.