Data Frame Augmentation of Free Form Queries for Constraint Based Document Filtering

Andrew Zitzelberger

A term project submitted to Professor Dennis Ng of
Brigham Young University
in partial fulfillment of the requirements for

CS 653

Department of Computer Science

Brigham Young University

December 2009

ABSTRACT

Data Frame Augmentation of Free Form Queries for Constraint Based Document Filtering

Andrew Zitzelberger

Department of Computer Science

CS 653

The ever expanding nature of the web makes finding relevant documents increasingly difficult. Traditional free form query methods, primarily keyword queries, are favored by a wide array of users but appear inadequate for the task of high precision information retrieval. Other search models such as form or graphed based models, or those using controlled vocabularies have been proposed but require the user to adapt to the model. These alternative models work well in domain specific searches or in semantic environments, but have been met with limited success when searching extremely heterogeneous collections such as the web.

We have constructed a prototype system that allows users to keep the ease and familiarity of keyword-like search while improving precision by using limited semantics to filter through documents. These limited semantics are extracted from queries in the form numerical constraints. Remaining keywords are then used in a traditional keyword search fashion to retrieve a result set on which filtering can be applied. This approach demonstrates a synergistic solution to improved search that lies between the extremes of bag of words and structured queries. Experimental results indicate that precision is improved for 56% of queries containing numerical constraints. Furthermore, precision for the prototype is 20% (62%) higher than the keyword query baseline measured at 42%.

Keywords: Data frame, keyword search, constraint filtering, high precision

Contents

Chapter 1

Introduction

As the number of pages on the web proliferates it becomes increasingly difficult for users to locate relevant documents. Despite the need to access relevant information quickly, most search engines do not offer the ability to specify semantics as part of a user's information need. Rather, semantic queries are generally issued through form based interfaces which access the deep web. The sheer size of the deep web (several orders of magnitude larger [6] than the estimated 11.5 billion indexable web pages [5]) indicates that users are interested in using semantics to locate relevant information.

Applying queries with semantics to the indexable web is a challenging problem because the web does not have the advantage of an overarching schema or completely structured data. This disallows the use of fully structured queries barring a complete extraction and integration of all web information. Furthermore, the success of companies such as Google and Yahoo! indicate that users are comfortable using keywords to search the indexable web. As such, we construct a synergistic solution that combines the simplicity of keyword search with the effectiveness of semantics. This type of solution allows users to take a step closer to semantic queries over the indexable web.

For semantics we focus on the extraction and application of numerical constraints. Numerical constraints are chosen because numerical comparisons apply universally, and because they are useful for a variety of common web based tasks such as searching for a car or planning a trip. We define a numerical constraint to be a relational operator containing one variable and one value. For example, if "car under $5000" was submitted as a query,

the extracted constraint would be ($Price < 5000$), where $Price$ is the variable and 5000 is the numerical value. The value of the variable cannot be determined by the query itself. Substitutions are made from extracted values in documents when determining relevancy to query, at which time the constraints can be evaluated.

In this work we present a prototype system that allows the user to input a free form query and retrieve relevant results with higher precision than a corresponding keyword query. This is done by extracting semantics from the query in the form of numerical constraints and enforcing that returned documents satisfy the extracted constraints. We choose the term free form query to indicate that a query can be any string and need not consist of only keywords. It is important to note, however, that any free form query can be processed as a keyword query and vice versa. The prototype system uses keywords to quickly narrow the search space and then applies the numerical constraints as a final filter.

Data frames [3] are the means by which the semantics are extracted from text. Each data frame consists of simple recognizers in the form of regular expressions as well as methods to compare extracted values. The means by which data frames are used for extraction and comparisons will be explained in Chapter 3. The synergistic approach to combining keyword search and the semantics provided by data frames is explained in Chapter 4.

Finally, initial experiments on two small test sets taken from the web indicate that when using queries with extractable constraints the prototype system improves precision over a keyword baseline for the top three results 56% of the time. Additionally, the data frame augmentation approach results in 62% precision over the test sets compared to 42% for the vanilla keyword search. The experiments and details of these results will be presented in Chapter 5.

Chapter 2

Related Work

To our knowledge this work is unique in using textual queries to combine keyword search with constraint based filtering to retrieve relevant documents. However, many similar attempts have been made to combine keyword queries with the semantics they represent. SPARK [14] is a system designed to take an arbitrary keyword query as input and return a structured query that represents the semantics. The system attempts to match the keywords to the closest concepts in underlying ontologies and then generates a set of ranked query graphs. Each query graph becomes a query and the user can select the best match to his/her query to run over the knowledge base. While the SPARK system also attempts to detect constraints in the keyword query it is fundamentally different than the prototype system in that it attempts to translate the keyword query into a structured query for use over a structured knowledge base.

MELISA [1] represents another system that integrates keywords with semantics for search. MELISA greatly resembles the prototype system in that it performs keyword search to narrow the search space and then applies filters to remove irrelevant documents. However, unlike the prototype system, the keyword search field is only one of the query fields required by MELISA. The user must enter the semantics of the query into specified form-like fields. As such, the simplicity of the keyword search model is mitigated. Additionally, MELISA has been designed to work specifically with the structure of MedLine sites for medically related searches and would need to be modified to work with other document types.

An approach similar to the prototype system is also presented in [11]. The authors present a system that again uses keyword search to the narrow the search space. However, this narrowed search space acts as the starting point for a spread activation algorithm through underlying domain ontologies. It is possible that results not in the set of documents retrieved by the keyword search may appear in the final results or vice versa. The system makes no mention of attempts to handle numerical constraints as defined in this work.

Perhaps the approach most similar to the prototype system is presented in [10]. This approach also uses keyword search to generate an initial document set and applies further filtering based on underlying ontological information. However, rather that using constraints to filter through pages the system re-ranks pages in the initial set using a semantic relatedness similarity metric.

Other systems which attempt to translate keyword queries directly into their semantics are presented in [13] and [12]. Each of these systems use underlying ontologies to translate the user's keyword query into a structured query that can be executed over a structured knowledge base. Such systems work well as long as the user query contains recognizable constants, but performance degrades sharply otherwise. The performance of the prototype system degrades more gracefully as keyword search is used if no constraints can be extracted.

The prototype system also bears some resemblance to question-answering systems. In such systems the users enters a fact based question and the answer is returned directly to the user, without requiring the user to search through pages that might contain the answer. Question-answering systems attempt to solve a difficult problem and many approaches (e.g. [8], [2], etc.) have and will be developed. These systems are similar in that they both enforce constraints based on extracted or inferred values. They differ in that question-answering systems attempt to understand the full semantics of the query and attempt to return the answer directly rather than relevant documents. The prototype system also differs in that it performs well even if the answer to a question cannot be found or does not exist.

Chapter 3

Data Frames

Data frames are the essential component for extracting semantics from queries and documents. A data frame is an abstract data type augmented with a name, instance recognizer for values, and operations to convert and compare those values [4]. These augmentations are represented in the forms of internal representations, external representations, units, canonicalization methods, and comparison methods. Each of these components work together to allow the data frame to extract and interact with semantics in a manner that simple keyword search cannot. Each of these components will be explained in more detail in the sections below and will refer to the example data frame in Figure 3.1.

3.1 Internal Representation

The internal representation of the data frame refers to how the information extracted by the data frame will be stored internally. These internal representations must correspond to data types or classes in the Java programming language. The partial data frame in Figure 3.1 uses Double as the internal representation in order to represent all possible prices.

3.2 External Representation

The data frame's external representation specifies how relevant values can be found in text. The external representation must be expressed in Perl 5 regular expression syntax. A data frame may also provide multiple external representations. Before an extraction step, each of these external representations will be combined into a larger regular expression with each

external representation separated by a logical OR. This regular expression is then used to find matches in the content of a document. These matches become the extracted values. The partial Price data frame in Figure 3.1 shows a common ways a price might appear in text. The "..." indicates that many other external representations must be provided in order to create a more complete Price data frame.

3.3 Units

A data frame may express units that are likely to accompany the external representation. Units are separated into left units and right units indicating where they are likely to occur relative to the extracted value. For the Price data frame in Figure 3.1 right units consists of strings such as "dollars" and "cents", as well as common abbreviations such as "K". Note that "K" is included as a separate unit in the regular expression because it can co-occur with other units.

3.4 Canonicalization Methods

Canonicalization methods associated with a data frame take the extracted value and the extracted unit and convert them into a single value with a standard format. For example, a canonicalization might take the extracted value $15K and store it internally as 15000 for use in future comparisons. Before a comparison is made the extracted value used to replace the variable in the constraint is also canonicalized. This conversion process allows users to ask queries with constraints on the car price using a string like "<15K" and pick up a relevant document even though the price is expressed as "13 hundred" in that document. Each data frame contains a pointer to the name of the canonicalization method which is written in the Java programming language. The partial data frame in Figure 3.1 has the method "toUSDollars" as its canonicalization method.

Price
internal representation: Double
external representation: \$[1-9]\d{0,2}(,\d{3})* | ...
...
right units:(K)?\s*(cents|dollars|[Gg]rand|...)
canonicalization method: toUSDollars
comparison methods:
LessThan(p1: Price, p2: Price) **returns** (Boolean)
external representation: (less than | < | under | ...)\s*{p2} | ...
...
...
end

Figure 3.1: Sample Data Frame for Price.

3.5 Comparison Methods

The final component of interest in the data frame is the comparison methods. These comparison methods are an integral part of this work as they are used for both extracting constraints from the query and for applying filtering constraints to documents. Comparison methods include an external representation of how the method might appear in text to facilitate extraction from the query. The LessThan comparison method in Figure 3.1 looks for text such as "less than" or "under" to recognize the operation. The operation is extracted if and only if the parameter $p2$ can be found with the operator. The method declares $p2$ to be of type Price, thus when looking for $p2$ the external representation of Price will be used. Once this information is extracted the constraints are created. Note that for this work constraint extraction is only applied to the query.

Chapter 4

Data Frame Augmentation

The principal contribution of this work is the process of augmenting keyword search with constraint based filtering using semantics as interpreted by data frames. In this chapter we explain the process by which the queries are processed and documents are filtered using this synergistic approach.

4.1 Data Frame Library

The first essential component to provide data frame augmentation is to have the data frames themselves. We refer to the data frames used by the system as the data frame library. The data frame library can consist of data frames for any type of values and can provide any type of comparison. For this work we construct numerical data frames with universally understood comparison methods. The specific data frames chosen for the prototype system are Number, Distance, Price, and Year.

Note that the latter three data frames are all specializations of Number. While the Number data frame will recognize any number, the specializations can each recognize a more specific type of number. In the event that there are overlaps in extracted values between number and any of the other data frames, the specialized data frame will claim the value. The intuition is that a more specialized data frame will provide a more precise context than a more general one. Note also that the latter three data frames can apply in multiple contexts. Distance can refer to the mileage on car, the length of a trip, or the elevation of a mountain. Price and year can apply to a large subset of purchasable items.

4.2 Constraint Extraction

After ensuring that a data frame library exists, the first step in query processing is to extract the constraints from the user provided free form query. The free form query is processed by each data frame in the library. This is done using by combining the external representations into a single regular expression as described in Section 3.2. This regular expression then replaces any parameters in the external representations for each comparison method to create a final regular expression for each comparison method in each data frame.

These regular expressions are run against the query and any matching values are removed from the query. The purpose of the constraint text removal is to remove noise from the query. The words that make up the condition may are unlikely to occur in a large number of relevant documents. For example the string "less than $15K" will not show up in a relevant document containing an item with a cost of $13000. After the constraint text is removed, the query is referred to as the reduced free form query. Note that if no constraints are extracted from the original query, the reduced free form query will be equivalent to the original query.

4.3 Keyword Search

After the constraint extraction step, keyword search will be performed using the reduced free form query. The model for keyword search will perform stemming and remove stopwords before employing a vector space model type search. If after removing stopwords the query is the empty string (because the constraint extraction removed everything except for stopwords), the reduced free form query will be replaced with the original query to ensure that some results are obtained. The Lucene engine will be used for indexing all test documents and for performing the keyword search.

4.4 Data Frame Filter

Documents retrieved after running the reduced free form query will be subjected to filtering based on the extracted constraints. First each document will be subjected to preprocessing in order to remove any meta-data or markup. For the web pages used in this project the pages were stripped of HTML tags. Then the data frames that successfully extracted constraints from the query will be run against each document. In this case the external representation of the data frame will be compiled as a regular expression and the contents of document will be tested for matches. In the event that a match is found, the data frame canonicalization method will be used to convert it to a standard format and this value will be used to test against the constraint. The filtering step requires that each constraint extracted from the query be explicitly passed by at least one extracted value from the document.

4.5 Result Ranking

The initial ranking of results is provided by keyword search using the reduced free form query. This order is maintained for the final ranking with the exception of that documents that are deemed non-relevant because they do not pass constraints are removed.

4.6 Output

The output web page displayed to the user provides links to a highlighting script with the URL to the original document, the conditions extracted from the query, and extracted text satisfying those conditions passed as arguments. This script highlights the extracted values enabling the user to quickly determine which text was used to satisfy constraints. In the event that more than one extracted value satisfies the query, each value will be highlighted in the text. This might occur, for example, if an advertisement lists prices for multiple items. The user also has the ability to mouse over that text to see which condition the highlighted text satisfied. Figure 4.1 shows an example of a highlighted page generated by the script.

White Pontiac Grand Am - $2000 (Provo UTah)

miscategorized
prohibited
spam/overpost
best of craigslist

Date: 2009-12-02, 7:02PM MST
Reply to: sale-5y9zn-1492330154@craigslist.org [Errors when replying to ads?]

I'm selling my white 1998 Grand Am Pontiac. I need to get it sold before Christmas because I will heading home to the Midwest and then will continue on to serve a mission for the Church of Jesus Christ of Latter-day Saints. It's a great car, a little older but still runs fantastic! It has not required much maintenance other than the usual car up-keep. It has new tires and the oil was changed the week before Thanksgiving. Shocks, fluids, and everything looks great. It has 104632 miles on it, power steering, cruise control, ABS breaks, power door locks, AM/FM Stereo, tilt wheel, dual front air bags, CD player, and an Ipod/MP3 adapter plug. It bluebooked for more than $4,000 but I don't have the time or desire to squeeze that much out of anyone, so I'm selling it for half that! I would be happy to show the car if you're interested.

- Location: Provo UTah
- it's NOT ok to contact this poster with services or other commercial interests

PostingID: 1492330154

Figure 4.1: Example Highlighted Page.

Test Queries

1) Find me a Wii game.
2) Find me a Honda for under 15 thousand dollars.
3) Roller Coaster more than 150 ft high
4) mountains at least 15K ft
5) games under $25
6) mountains less than 4 km
7) ps games $<$ 40 bucks
8) coasters longer than 1000 feet
9) car for under 5 grand newer than 1990 with less than 115K miles
10) more than 15K miles under 5 grand newer than 2004

Figure 4.2: Test Queries.

Chapter 5

Experimental Results

In evaluating the prototype system we seek to test the claim of increased precision. Due to this claim and studies that indicate that a typical user only looks at approximately the first three documents [7] we record precision over the top three results for a variety of experiments.

5.1 Experimental Setup

The prototype system was tested using two small test sets created from crawling actual documents on the web. Each test set contains 50 advertisements each from the car+trucks and video gaming sections of http://provo.craigslist.org, as well as 25 documents each from a list of roller coasters and a list of mountains on http://en.wikipedia.org.

The first test set is a development-test set that experimenters had access to while creating the data frames. The regular expressions for the data frames were tested against these documents to ensure values were extracted properly. To ensure that the system also performs well against unseen pages the second test set was withheld as a blind test set. After data frames were deemed to be sufficiently robust, experiments were run against the blind test set without allowing experimenters to see the documents before hand.

A set of ten predetermined queries was used to test the system (see Figure 4.2). Note that the first and last query should both be processed as keyword queries with no increase in precision. The first because there are no constraints, and the last because there are no

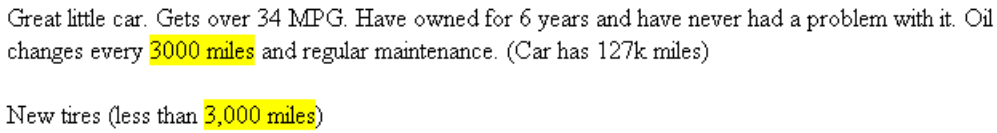

Figure 5.1: Example of Noise in Documents.

keywords remaining after constraint extraction. This leaves eight queries on which a possible precision increase may occur.

Each query is processed in three distinct manners. First the entire free form query is processed as if a keyword query. Second, constraints are removed from the query and the reduced free form query is used as a keyword query. Finally, the entire system with data frame augmentation is employed. For each of these methods the precision for the top three documents is reported.

5.2 Results

The results for the development test set can be seen in Table 5.1. As expected, queries one and ten showed no difference in precision. For the eight queries containing constraints, removing the constraints and using keyword search improved the precision in four of the eight cases (50%). Precision remained the same in two cases and lowered in two cases.

Running the full system with data frame augmentation resulted in improved precision over keyword search in five of the eight (62.5%) queries containing constraints. In the remaining three cases the data frame augmentation performed equally well as keyword search. However, the full system only produced 100% precision for three of the eight (37.5%) of the time.

The results for the blind-test set can be seen in Table 5.2. For the blind-test set removing constraints from the original query made virtually no difference in precision results as six of the eight (75%) of the queries showed no difference in precision. For the two remaining queries one improved and one decreased. The full system improved for four of the

Query	Keyword Query	Reduced Query	Data Frame Augmented Query
1	0.33	0.33	0.33
2	0.67	1.00	1.00
3	0.33	0.33	0.67
4	1.00	0.67	1.00
5	0.00	0.33	0.67
6	0.00	0.00	0.33
7	0.33	0.00	0.33
8	0.33	1.00	1.00
9	0.33	0.33	0.67
10	0.00	0.00	0.00

Table 5.1: Precision@3 for development-test set.

Query	Keyword Query	Reduced Query	Data Frame Augmented Query
1	0.67	0.67	0.67
2	0.67	1.00	1.00
3	0.67	0.67	0.67
4	0.33	0.33	0.67
5	0.67	0.67	1.00
6	0.00	0.00	0.00
7	0.33	0.33	0.33
8	0.67	0.67	1.00
9	0.67	0.00	0.67
10	0.33	0.33	0.33

Table 5.2: Precision@3 for blind test set.

eight queries (50%) and remained the same for other four. The system only obtained 100% precision for three of the queries.

Overall the data frame augmentation approach performed significantly better than the keyword search baseline (see Figure 5.3). The keyword query approach achieved about 42% precision while the prototype system obtained about 62% precision. Studies indicate that users expect about 50% precision from keyword search in the top three results [7]. The 12% increase over this expectation thus represents significant potential for this approach. Furthermore, the prototype results in increased precision for 56% of the queries.

Precision@3/Query Type	Keyword Query	Reduced Query	Data Frame Augmented Query
Dev-Test Queries	33%	40%	60%
Blind-Test Queries	50%	46%	63%
Overall	42%	43%	62%

Table 5.3: Summary Precision Comparison.

5.3 Issues

In evaluating the precision of the prototype system two issues were discovered that lead to less than perfect precision. The first issue is that the keyword search does not rank results adequately or does not retrieve all of the relevant documents for filtering. This is most noticeable for the query "ps games under $25." A domain specific search engine or a user with knowledge would recognize "ps" as refering to playstation. However, a simple keyword search engine does not recognize that this is referring to same concept. As such, the filtering begins with documents that do not necessarily meet the playstation requirement but pass the filter because they contain a price under $25.

The second reason for receiving less than perfect results is the presence of other numbers on the page that strictly satisfy the constraint but do not satisfy what the user is actually looking for. For example, one document stated that there was less than 3000 miles on the tires (see Figure 5.1). The filter picked up the 3000 miles as satisfying the constraint of being under 115K miles and allowed the page to pass despite the fact that elsewhere on the page it says the car has 127K miles on it. This problem was more prevalent when dealing with the roller coaster documents. These documents often contained distance information about the length, height, drop, and even elevation of the amusement park in which the reside, not to mention historical information. As such, it was often possible that some of these distance measures could satisfy the constraint even though it was not relevant to the user.

It is also worth noting that the reduced keyword query occasionally performs worse than the original keyword query. The lowering of precision usually occurred in cases where

the noise from the constraint text actually helped find relevant documents. For example, for the query "games $<$ 40 bucks", one of the relevant documents contained the word "bucks" and was picked up by the original free form query but not by the reduced free form query. It is not clear whether using the reduced query is actually beneficial, although this opens avenues for further study.

Chapter 6

Conclusions and Future Work

We have developed a prototype system that combines traditional keyword search and limited semantics in order to obtain higher precision results for the class of queries that contain numerical constraints. This system works by extracting constraints from the query through the use of data frames, performing a keyword search using non-extracted words, and filtering the returned document set using the numerical constraints. Experimental results indicated that for queries with numerical conditions this approach increases precision for 56% of the queries. Furthermore, there is a 20% difference in the precision between the keyword baseline and the data frame augmented search (42% and 62% respectively). Additionally, this system causes no degradation in performance to queries with no numerical conditions.

6.1 Future Work

While this shows great promise there is still much work yet to do. This work demonstrates that there is value for filtering on simple numeric constraints but there are many other types of constraint of interest to a user. Further research is needed on how well other constraints can perform in combination with keyword search. Additionally, the trade offs between precision and recall are ignored in this work but offer ample research opportunities, especially as more sophisticated filters can be developed.

Furthermore, there are other issues to consider in actually employing such a system. Each data frame must be run against the query. As the number of data frames increases overhead is added that will slow down the processing time. Further research is needed

on method to reduce this overhead by only running data frames that are likely to extract constraints rather than the full set.

Additionally, the prototype performs the extraction required for filtering documents at query execution time. This type of filtering takes a few seconds for short documents and the processing time grows linearly with the length of the document. This delay in displaying results in an unreasonable wait time for users. Further research is required into proper storage/indexing of terms in relation to data frames and how such a system could be brought up to scale for large document collections or even the world wide web.

This type of system also sets up motivation for a pay-as-you-go [9] framework for information retrieval. In such a framework, a user always has the baseline of keyword search but can add semantics to the system through components such as data frames to achieve higher precision results. The user must pay the price to get higher precision by taking the time to build the data frame. Or, one could imagine data frame libraries on the web from which a user can pick and choose to build specialized searches of interest to him/her and change the way people search for hard to find information.

References

[1] Jose Maria Abasolo and Mario Gomez. Melisa. an ontology-based agent for information retrieval in medicine. In *Proceedings of the First International Workshop on the Semantic Web (SemWeb2000)*, pages 73–82, 2000.

[2] Eugene Agichtein, Chris Burges, and Eric Brill. Question answering over implicitly structured web content. In *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 18–25, Washington, DC, USA, 2007. IEEE Computer Society.

[3] D.W. Embley. Programming with data frames for everyday data items. In *AFIPS National Computer Conference (NCC'80)*, pages 301–305, Anaheim, California, May 1980.

[4] D.W. Embley and A. Zitzelberger. Theoretical foundations for enabling a web of knowledge. In *Foundations of Information and Knowledge Systems, Sixth International Symposium (FoIKS 2010) (accepted paper)*, 2010.

[5] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902–903, New York, NY, USA, 2005. ACM.

[6] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, 2007.

[7] B. Jansen and A. Spink. An analysis of web documents retrieved and viewed. In *International Conference on Internet Computing*, June 2003.

[8] Jimmy Lin and Boris Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 116–123, New York, NY, USA, 2003. ACM.

[9] Jayant Madhavan, Shirley Cohen, Xin L. Dong, Alon Y. Halevy, Shawn R. Jeffery, David Ko, and Cong Yu. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[10] Antonio M. Rinaldi. An ontology-driven approach for semantic information retrieval on the web. *ACM Trans. Internet Technol.*, 9(3):1–24, 2009.

[11] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi Aragao. A hybrid approach for searching in the semantic web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 374–383, New York, NY, USA, 2004. ACM.

[12] Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. Ontology-based interpretation of keywords for semantic search. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 523–536. Springer, 2007.

[13] M. Vickers. Ontology-based free-form query processing for the semantic web. Master's thesis, Brigham Young University, Provo, Utah, June 2006.

[14] Qi Zhou, Chong Wang, Miao Xiong, Haofen Wang, and Yong Yu. Spark: Adapting keyword query to semantic search. In *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, pages 687–700, November 2007.