INSTITUTE OF SYSTEMS SCIENCE, NATIONAL UNIVERSITY OF SINGAPORE

Question-Answering Machine based on Deep Linguistic Parsing

A dissertation

Submitted to Institute of Systems Science
in partial fulfilment of the requirements
for the
Master of Technology in Knowledge Engineering

Students
Tuan Anh, Le
Ying, Sun

Supervisors
Professor Francis Bond
Ms Fan Zhenzhen

4/5/2013

By completed this project, we have achieved two important results. First, we contributed to the development of computational linguistics with our DMRS toolset. It provides a much easier way to do grammar comparison and a mechanism to store and search over a large number of DMRS graphs. Second, we have demonstrated that question-answering is practically possible with our DMRS persistent framework and DMRSQL.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
DL	Deep linguistic
DMRS	Dependency Minimal Recursion Semantics
DMRSQL	DMRS Query Language
HPSG	Head-driven Phrase Structure Grammar
LinGO ERG	Linguistic Grammars Online English Resource Grammar
MRS	Minimal Recursion Semantics
RDBMS	Relational database management system
QA	Question-Answering
QAM	Question-Answering Machine

1 Introduction

1.1 Purpose

Developing strong artificial intelligence systems is an open problem which a lot of researchers are trying to solve. One of the sub problems of strong artificial intelligence is question-answering machine. If we are able to build a machine which understands questions in human nature language and provides answers, we are one step closer to the strong AI solution. Recently, there are a lot of developments in computational linguistics and those are forming the foundation for further development of software solutions with deep-linguistic analysis capability. However, there are still very few knowledge-based solutions built on deep linguistic techniques in the market. In this project, we propose a novelty approach for building a question-answering machine which is practical enough for real world applications.

1.2 Scope

Due to the limitations of time and resources, we only built a prototype as a proof of concept for the approach we are proposing in this paper. Our focus at this early milestone of the project is providing the core functions of the QA framework as well as referring to the important literatures, resources and research directions for further developments.

Most of the efforts have been spent on crafting the linguistic research toolset: DMRS persistent mechanism, DMRS visualisation and DMRS indexing & retrieval methods.

1.3 Methods

The key theories and methods have been used in this paper include, but not limited to, English Resource Grammar, WordNet, DMRS, relational database, thick web-client, graph search, parallel and distributed processing.

1.4 Limitations

- The proposed DMRS Query Language is not fully implemented. We implemented the DMRSQL just to proof that the indexing are working fine.
- The cluster search function is only accessible via local terminal interface and it only supports parallel search. A web service interface is required to enable distributed graph search.
- We only optimised the performance of the system to perform search over databases which is smaller than 2 GB. Performing search on larger databases can be very slow, especially when we're searching for common lemmata. In order to use this solution in practical uses, we still need to solve some I/O bottlenecks and indexing problems.
- The module to convert nature language queries into DMRS queries is omitted at the moment.

1.5 Assumptions

To simplify the problem, we assume that any knowledge is explicitly expressed within a single sentence. Furthermore, we also assume that there is no sarcasm in the data source. Having these two assumptions, we assume that answers can be retrieved based on performing grammar & semantic analysis on textual data, without using any background knowledge or further co-reference or summarisation analysis.

2 Literature Review

Currently, the method is being used for question-answering is pattern-based matching, often relies on regular expressions, as introduced by (Daniel and James 2009, 812-821). For examples:

Table 1: Some answer-extraction patterns (Speech and Language Processing 2009, 819)

Pattern	Question	Answer
<AP> such as <QP>	What is autism?	… **developmental disorders** such as autism …
<QP>, a <AP>	What is a caldera?	…the Long Valley caldera, a **volcanic crater** 19 miles long …

Although these kind of rules can be hand-coded or using machine learning, this method still relies on text surface analysis, and mostly ignore the grammar and semantics information the text data carry. From our observation, the fact that this approach is accepted widely is due to they can deliver results using cheap computational power.

Personally, we aren't happy with just the approach above, and did further research. That led us to more methods such as morphological alternations, lexical alternations, passage retrieval with machine learning, etc. (Sanda and Dan 2004, 561-582). These methods have been used in several TREC QA tracks.

Surface text retrieval, even with token expansion or machine learning, is not promising enough to us. The best result achieved in TREC QA 2006 is 57.8 % for factoid questions (Bob March 25, 2008). Instead of working out yet another information retrieval based method, we had chosen a different research direction: computational linguistics.

2.1 LinGO ERG

"*ERG is one of the most broad-coverage and linguistically precise grammar library of English*". Until July 2003, ERG comes with a "*hand-built lexicon of around 10,000 words*" and it can "*produce semantic representations for about 83 per cent of the utterances in a corpus of transcriptions of some 10,000 utterances, which vary in length from one word to more than thirty words*". (LinGO ERG n.d.). Moreover, we have found that there's effort to use ERG to perform deep linguistic analysis on English Wikipedia corpus (Dan, Stephan and Gisle 2010). Although there are a lot of post-processing need to be done before the parsed results can be used, this suggested a promising direction to our research.

2.2 Minimal Recursion Semantics

Using ERG parsing rules, we can obtain the parsed structures of sentences. There are many ways to represent this parsed information. Syntactic derivation tree is one of the examples. However, different methods represent structural ambiguities in different level of efficiency. Several theories have been introduced, and among those, Minimal Recursion Semantics (MRS) is gaining popularity, especially in recent years.

MRS is a meta-language which enables semantic representation. The most important characteristic of MRS is its computational efficiency. MRS itself is flat representation, yet it's able to express scopes when it is necessary. Normally, scope information is ignored but we can retrieve that in situation where scope information is needed.

This is an example which demonstrates how MRS represents different interpretations of the sentence "Every dog chases some white cats" (Kostadin n.d.). If we understand the sentence as "(There are) some white cats (which) every dog chases (those white cats)". We can represent in MRS (simplified format) like this

```
some(y, white(y) ^ cat(y), every(x, dog(x), chase(x,y)))
```

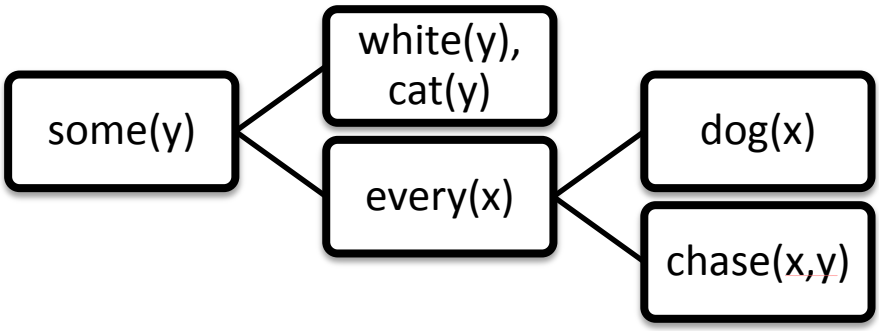

Figure 1: MRS for (There are) some white cats (which) every dog chases (them)

However, if we want to interpret the sentence in another way, which is "(each of) every dog (will) chase some white cats", then we have this representation:

```
every(x, dog(x), some(y, white(y) ^ cat(y), chase(x, y)))
```

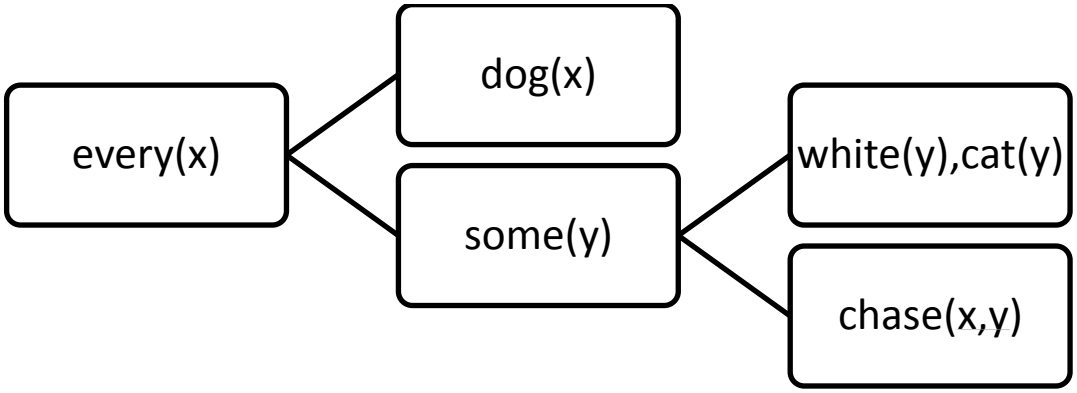

Figure 2: MRS for (each of) every dog (will) chase some white cats

2.3 Dependency MRS

Although MRS is very efficient in representing semantics, its readability is still low, especially for "*consumers of the representation, human annotators or parser comparison*" (Ann 3 April 2009). DMRS is the answer for this problem. Unlike the original MRS, Dependency MRS (DMRS) contains only predicates (or nodes) without variables. The relations or dependency will be represented by introducing links between predicates. This nature makes visualisation of DMRS become very simple. We can think of a DMRS representation of one sentence as a directed graph with predicates (nodes) as vertices and the links between them are edges. Searching information based on sentences' structure can be done easily by using a graph search algorithms.

2.4 WordNet

"*WordNet is an on-line lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory*". Since 1985, WordNet has been hand-coded by linguists and psychologists and evolved into a massive lingua-ontology with more than 206,941 Word-Sense pairs (WordNet 3.0 database statistics n.d.)

WordNet not only contains precise information about different senses of words, but also their type (nouns, verbs, adjectives, etc.), synonym, antonym, hyponym, etc. Therefore, using WordNet, we can perform various linguistic operations such as alternate lemma searching, query expansion. We also can refer to WordNet senses when performing meaning annotation.

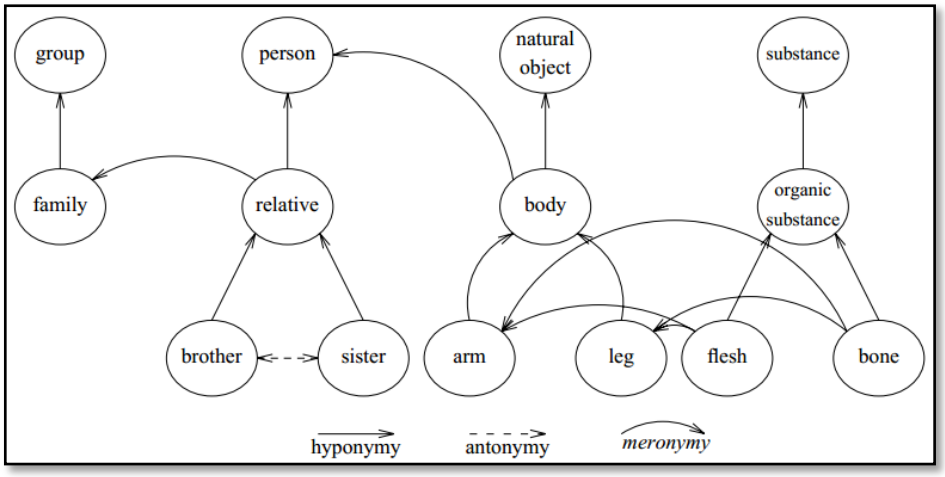

Figure 3: Semantic relations in WordNet (George, Richard and Christiane August 1993)

2.5 Frameworks and Toolsets

There are several tools are being used to take input as ERG grammar rules and text documents to produce parse tree and MRS, for example PET system by DELPH-IN group (Delph-in PET n.d.). We also have accessed to an open source tool written in Python to convert generated MRS representations into DMRS representations. However, we lack of a framework to handle DMRS data efficiently. Currently DMRS data are stored in XML format in text files without any indexing. In addition, a visualisation tool for visualising DMRS is also needed for grammar debugging or annotating.

2.6 Motivations

The dominated method for developing information retrieval system until a decade ago (before the year 2000) is “tokens hacking”. Documents are broken into bags of tokens and indexed with different frequencies (term frequency, inverted document frequency, etc.). After that these pre-calculated frequencies are combined using different methods to retrieve needed information. However, these methods only work on the surface of corpus and ignore the grammar & semantic information (There are attempts to take some kinds of semantic into consideration such as n-gram with sliding windows but those are still classified as surface-based methods).

Let’s say we want to retrieve information based on given tokens which related to each other. With surface based method, most of the time we’ll perform a search to find documents which contains those tokens within some sliding windows. This method doesn’t guarantee that the results contain exactly the information we are looking for. A sentence might be ignored if the distance between two tokens is larger than the window size. In addition, “bag of words” method doesn’t take grammar into account, for example: “Mary kills John” is equivalent to “John kills Mary”. We believe that deep linguistic analysis method can help us to overcome those existing problems.

2.7 Challenges

The theories and techniques mentioned above can be used to develop our Question-Answering Machine. Applying ERG parser on collected text data, we are able to generate MRS data and then convert them into DMRS format. After that we can search for answers in text-based data by performing graph search at the DMRS level.

However, as pointed out in section 2.5, we lack of toolsets and software frameworks to develop this system. Storing DMRS in text format without any indexing method means that searching are extremely slow and impractical. We need to find a better way to represent and store DMRS data. This representation must be search friendly, ideally with indexing feature.

Another problem is we don't have a mechanism to compare different DMRS representations and detecting parsing errors. ERG is not bug-free and we need to debug parsed structures. But displaying DMRS in XML-format is not human friendly and take much longer time to figure out what went wrong. Therefore building a DMRS visualisation tool is one of our top priorities.

A query method enables efficient travelling and searching on massive amount of DMRS data is our third target. Containing deep parsed linguistic structure of text documents, DMRS data are often bigger than the surface character streams of the documents. So the next challenge will be designing a good retaining and indexing strategy.

2.8 Feasibility Study

Due to the mentioned challenges we faced and the limitations of time and resources, we decided to scale down the project scope. We didn't aim at providing a full working solution in the first stage of this project. Instead, we only develop the framework with important core functions. The remaining features will be implemented as proof of concepts.

As ERG, PET and MRS2DMRS conversion tools are available; generating DMRS from raw text corpus is technically feasible. We need to consider the storing, visualising and the searching mechanisms. For storing DMRS structures, a relational database engine can be use, as this is the most widely used data storage mechanism. Most of the relational database engines come with T-SQL query language, indexing feature, and data view so handling data with RDBMS is much easier compare to developing a data manager from scratch.

Graphical user interface on computers today are very powerful and visualisation DMRS data is not a big problem. What's left is allocating the time to develop the visualisation tool only.

The last thing to consider here is DMRS query language. Although T-SQL query language is available, writing SQL queries is not simple plus it's meant for searching over table structures, not graphs. More than that, query DMRS base on T-SQL will make our system depends on a particular data persistent layer underneath thereby losing the design portability. To overcome this problem, we will introduce another simpler query language to implement search function.

3 System Design

3.1 Knowledge Representation

In section 1.5, we assumed that the textual data contains knowledge and knowledge can be expressed within a sentence. So, the knowledge to be managed in this situation is the extracted DMRS graphs. We represent the whole knowledge base as the available corpora; each contains a number of documents. Each document contains a number of sentences and each sentence has some valid representations expressed in DMRS format.

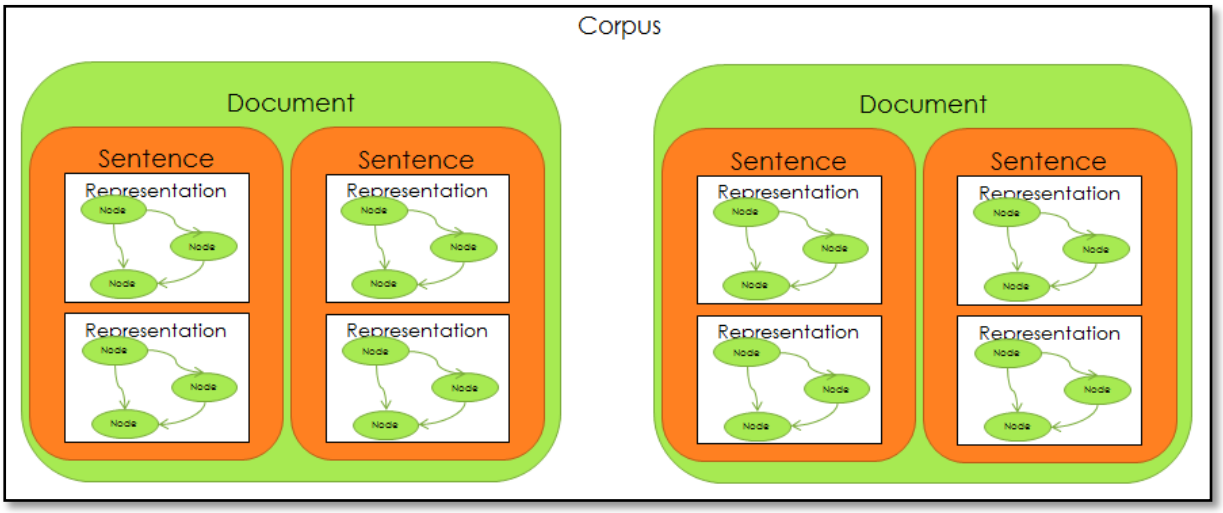

Figure 4: Knowledge Representation

Below is the formal class diagram emphasises our knowledge representation.

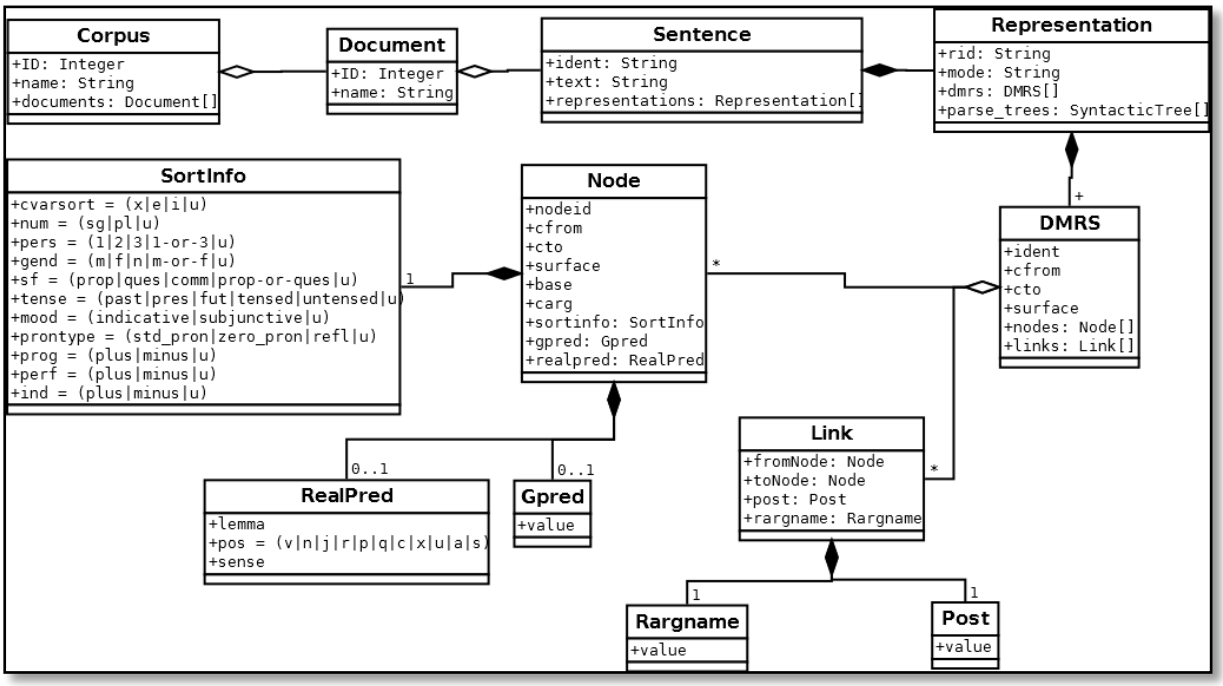

Figure 5: Class diagram of knowledge representation

3.2 Knowledge Retaining

After formally defining the structure of knowledge representation, knowledge retaining is simply a process of serialising the data entities into a storable format (binary stream, text stream, database rows, etc.) and transferring them to the designated storage location. Among the retaining methods, two possibilities of storing the DMRS data into storage devices are taken into consideration: using text files and database engine. Although writing data to text files sometimes can be simpler than to a database, this method requires manual indices, more difficult to maintain and low retrieval performance compare to a database engine.

After carefully evaluating the situation, we decided to use both of the mentioned methods. We store raw DMRS data in text files and store indexed DMRS data in SQL database. Text file is appropriate when we want to retrieve the original raw file of a particular DMRS representation while database can achieve faster indexing and retrieval performance.

3.3 Knowledge Retrieval

In this system, knowledge retrieval is performing DMRS searches given a particular DMRS query. As one can observe from DMRS structures, predicates or nodes are mostly extracted from the original string (constant strings or lemmata) except grammar predicates. Therefore, the number of nodes generated from a sentence should not be too different from the number of tokens extracted from the words. A rough upper bound can be estimated as three or four times more than the number of tokens. Performing search in a graph with less than a few hundred of nodes can be considered a small graph search problem.

Let refer back to section 1.5 where we assumed that sentences in the corpus are independent, and any knowledge is expressed within a sentence. Hence a search task performed on a corpus can be decomposed into many independent search tasks where each task handles a small graph. By observing this characteristic, we recognised that the graph search problem we are trying to solve in this project is parallel search in nature.

We also need a language to represent our DMRS query. As discussed in section 2.8, T-SQL is not suitable to write our query because we don't want to tight couple our design with any particular storage mechanism. Thus we have introduced DMRS Query Language: DMRSQL.

The query language is simple and contains many borrowed ideas from DMRS representation. We can form a query to search for nodes by using "**?**" (any node), "**L:lemma**" for lemmata, "**C:Carg**" for constant strings, or "**G:named_rel**" for grammar predicates. Any node search clause X without the prefix L:, C: or G: will be treated as (**L:X OR C:X**). We also can search for links with **([fromnode] [post]/[rargname] [tonode]).** Additionally, we can join the search clauses using Boolean operators (AND, OR, NOT) to form more complex queries.

For example, if we want to search for a sentence with a constant string "Torvalds", the string "Torvalds" links to a lemma "develop" and lemma "develop" links to the constant string "Linux", and a node "scratch" (either a lemma or a constant string), we can use the following query:

```
(C:Torvalds) AND (C:Torvalds / L:develop) AND (L:develop / C:Linux) AND scratch
```

3.4 System Architecture

Finally, the system architecture is designed as in Figure 6.

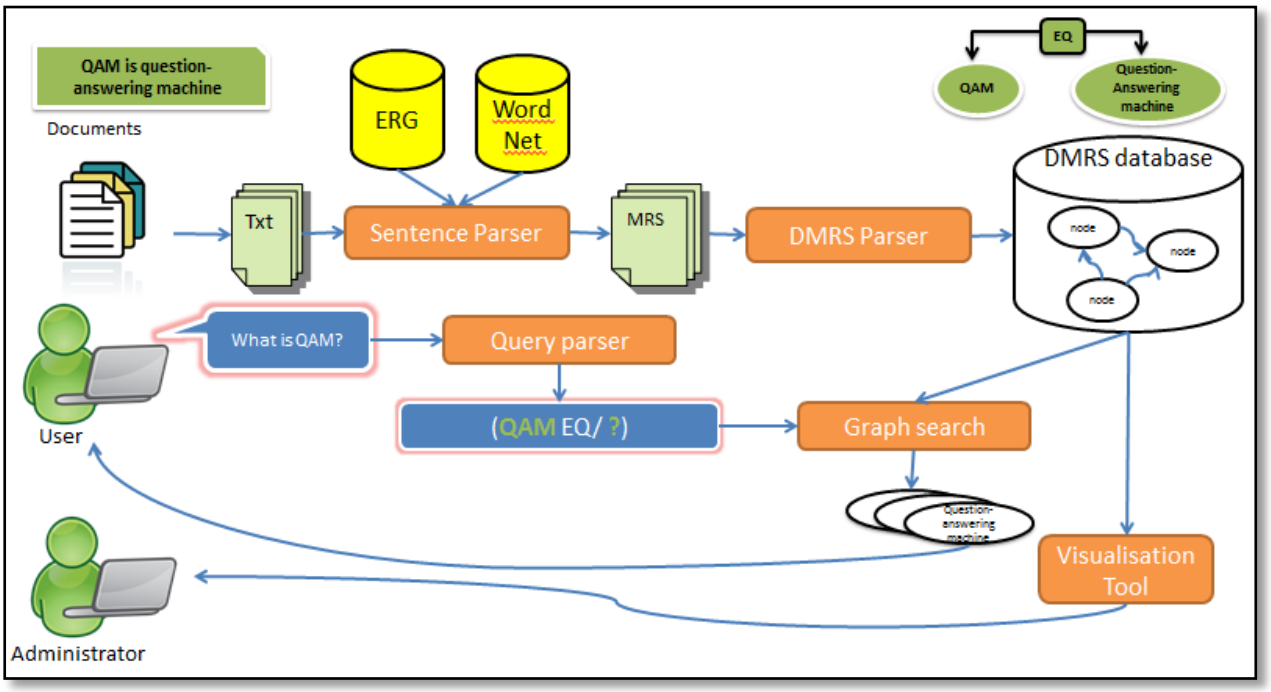

Figure 6: System architecture

Our system has three main modules: data parsing module (consists of Sentence Parser and DMRS Parser), graph search module and visualisation module.

When there're new documents to be added to the system we'll convert all of them into pure text format. Next, we'll perform sentence segmentation to segment each document into a list of sentences. After that, we will use Sentence Parser (with ERG and WordNet) to convert those textual data into MRS format. Finally we use the tool MRS-to-DMRS to convert MRS to DMRS format with nodes and links and then save them to database.

Now, the knowledge base is ready to use. When a user enters a query in nature language, for example "What is QAM?" this query will be converted into machine understandable format, in this case is DMRS query language "(QAM EQ/ ?)". That means: find any sentence with an EQ link from a node which is a constant string "QAM" to any other node. This query will be passed into Graph Search module to perform search and the results (the list of the nodes represent the question mark) will be displayed on user screen. Currently in the prototype, we don't implement the nature language query parser and we only support query written in DMRS query format. The DMRS query language is not fully implemented but only node search, link search and Boolean AND operator.

The knowledge-base administrator can use visualisation tool to visualise DMRS contents stored in the database and perform grammar comparison and grammar debugging.

4 Implementation

We have implemented the system as a web application using Python and Django web framework. Knowledge representation is done using OOP. When we want to store some information, they will be serialised and then stored into SQLite databases.

The visualisation module is one of its own kind. Since nobody has implemented this visualisation before, we have to build it from scratch with HTML, CSS, JavaScript and Raphael web graphic library.

The DMRS query parser and the SQLite query converter also needs to be coded as well. To build the parallel search cluster, we relied on Python multithread programming technique.

4.1 Implementation Packages

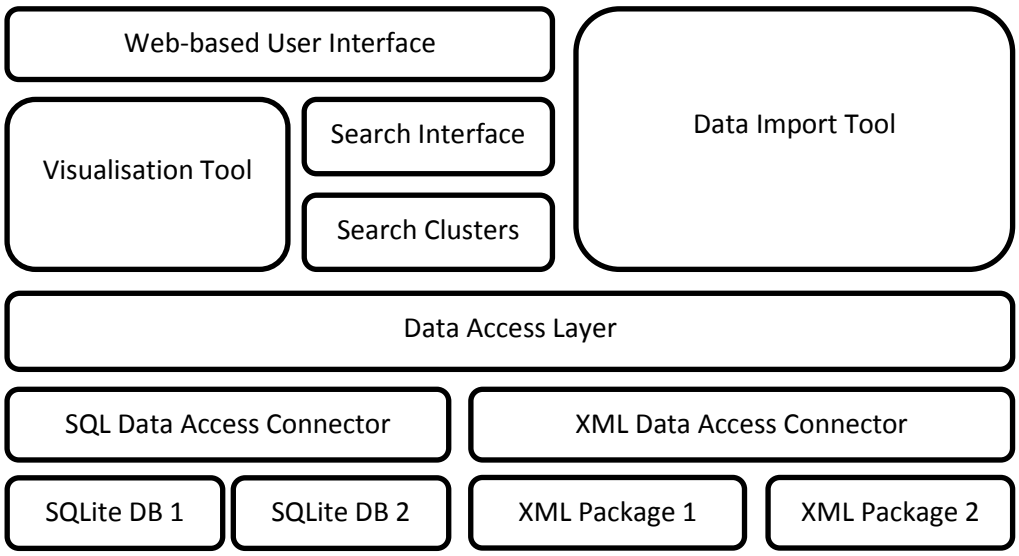

Figure 7: System Package Design

4.2 Implementation Challenges

As easy as it might sound, we actually faced a lot of challenges during the implementation phase. The first problem is the parse optimisation. After text data have been parsed to DMRS in text format, we still need to parse them one more time from text file to store into the database. And this task slows down the whole process. DMRS text-based contents are stored in many files on storage devices, so we have to perform many I/O actions in order to read the whole dataset to store into database.

Writing efficient database insert script is another challenge. At the beginning, we need to run the import script for a few hours to store the test document “the Cathedral and the Bazaar”. After we have access to the real data, the ACL conference paper corpus with more than 25000 documents, we realised that it’s impossible to convert the data before the project deadline with that slow parser. After many tweaks and hacks, the parser can import the whole ACL corpus into our system within a day!

The text-based format reveals another I/O problem. When we have too many files in a directory (even a few KB in size as DMRS files), reading and processing files becomes very slow. We solved this problem by create compressed archives with very low compress ratio for each corpus and retrieve from this archive when we need access to the raw DMRS data.

Next issue is the big data nature of this project. Since we have to analyse and store the deep structures of each sentence in a document as well as the related indices. So in total a document with a few hundreds sentences takes roughly 3 MB of storage in database. The whole ACL corpus is about 75 GB of data. On the

machine we used to develop this system which has 2 GB of RAM, a dual core 1.6 GHz CPU and a HDD operates at 5400 rpms, it's extremely difficult to handle this huge amount of data, even with help of the database indexing. We have tried to solve this problem with parallel processing, but it doesn't help much in this situation where the actual bottleneck is the HDD reading speed (20 MB/s).

4.3 Design Validation

We have consulted several reliable sources to validate our system design. Throughout our project life cycle, we have frequently attended discussions with professor Francis Bond (NTU), Mr Mathieu Morey and our supervisor Ms Fan Zhenzhen to refine the conceptual design of our system. We took minute notes after every meeting and produce a check list to ensure we haven't left out any suggestions. The DMRS representation and database schema follow the DMRS DTD specification provided by Ms Zina Pozen and have been double-checked every time we are notified about any update.

4.4 Verification & System Testing

We have used several methods to test the implemented system. Python Unit Test is adopted to test tasks which can be performed using automation test such as testing DMRSQL parser. For search feature, we used a set of known test cases to ensure we've got correct results. The test results given in this section mostly retrieved from the document "the Cathedral and the Bazaar" and ACL-X corpus.

4.4.1 Contrasting DL-based method and surface-based method

Assume we want to find information about training data. A surface search lemmatized query is "**(train) AND (data)**", while a lemmatized DL-based query is "**(train / data)**".

We executed the first query and found 160 sentences:

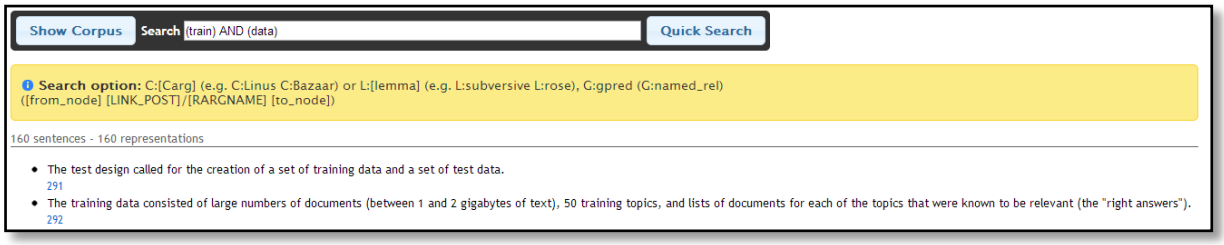

Figure 8: Surface search result for (train) AND (data)

In the results list, we found the following sentences:

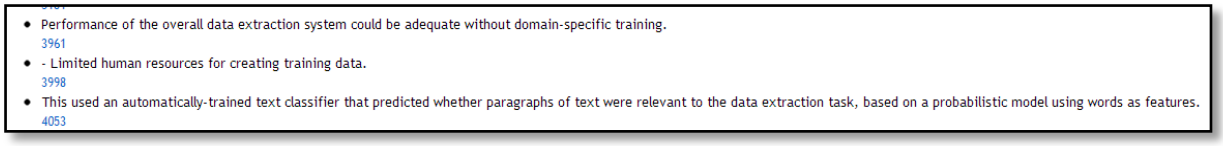

Figure 9: Surface search return irrelevant results

As we can see, the sentences 3961 and 4053 are not about training data, although they contain the lemma "train" and "data".

Now, we perform the DL-based query, which returns only 24 sentences.

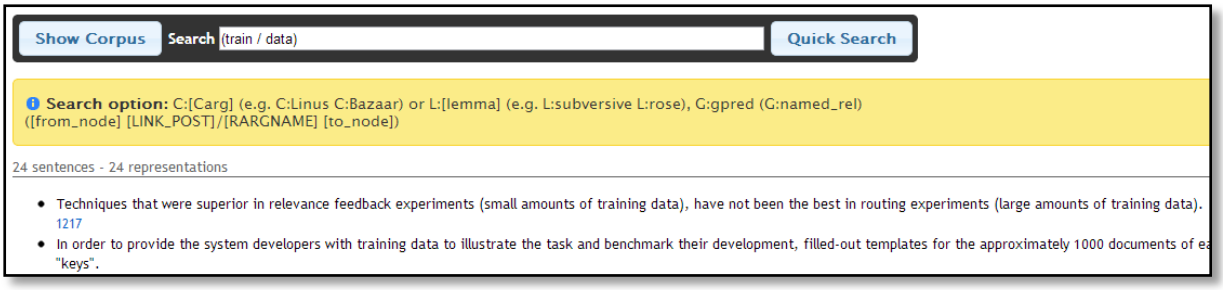

Figure 10: DL-based search results for (train / data)

All of these 24 sentences are about training data. One more interesting fact is the two lemmata don't necessary stay next to each other, as given in the sentences 5785 and 5895 below:

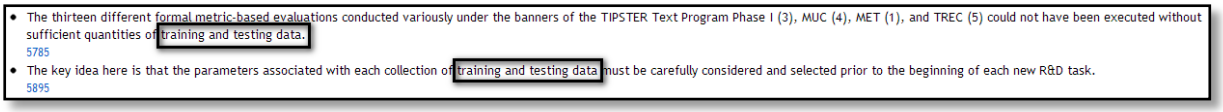

Figure 11: DL-based method returns "training and testing data"

Similarly, we can perform another search for "parse document". A DL-based query **"(parse / document)"** return 3 sentences with "**…documents are parsed…**", "**…parsing documents…**", "**…each individual document is reparsed…**". This result has demonstrated the ability to search for information based on deep grammar & context analysis.

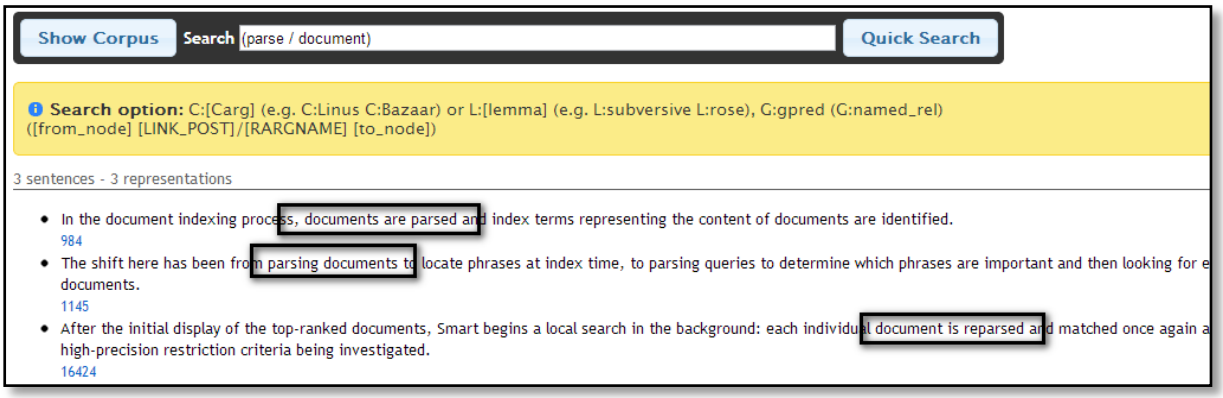

Figure 12: DL-based query (parse / document)

In contrast, a surface matching with "**(parse and document)**" returns 13 sentences, which contains irrelevant sentences such as "The documents are uniformly formatted into SGML, with a DTD included for each collection to allow easy parsing." or "Different processes, such as tagging or parsing, can be applied to documents or collections and the results compared and analysed."

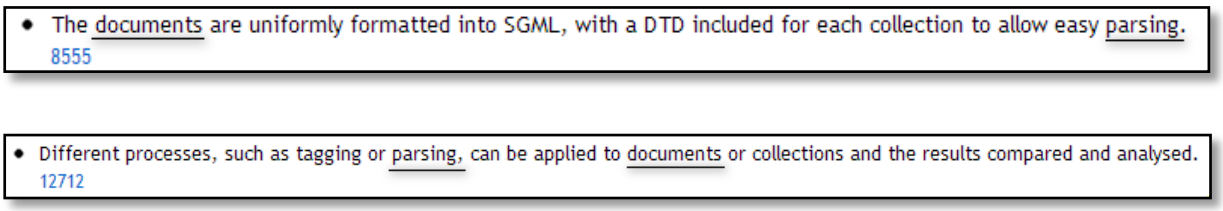

Figure 13: Surface search (parse AND document)

With the flexible of DMRS query language, we can perform more complex query, such as looking for “results from a university”. First, we define the query as **“(from / result) AND (from / ?) AND (G:compound_name_rel / ?) AND (G:compound_name_rel / university)**”. This means search for sentences contain the lemma “result” which links to a lemma “from”. The lemma “from” will link to another node which links to a compound name relation. That compound name relation links to a lemma “university”.

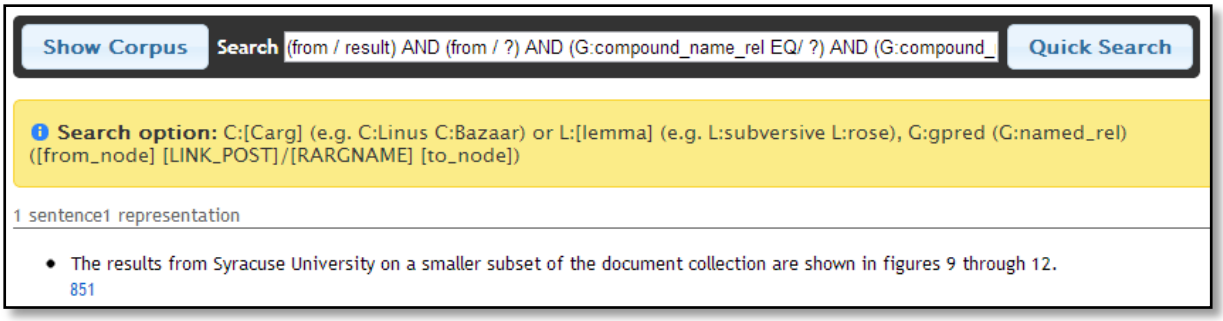

Figure 14: Find "result from a university"

4.5 Screenshots

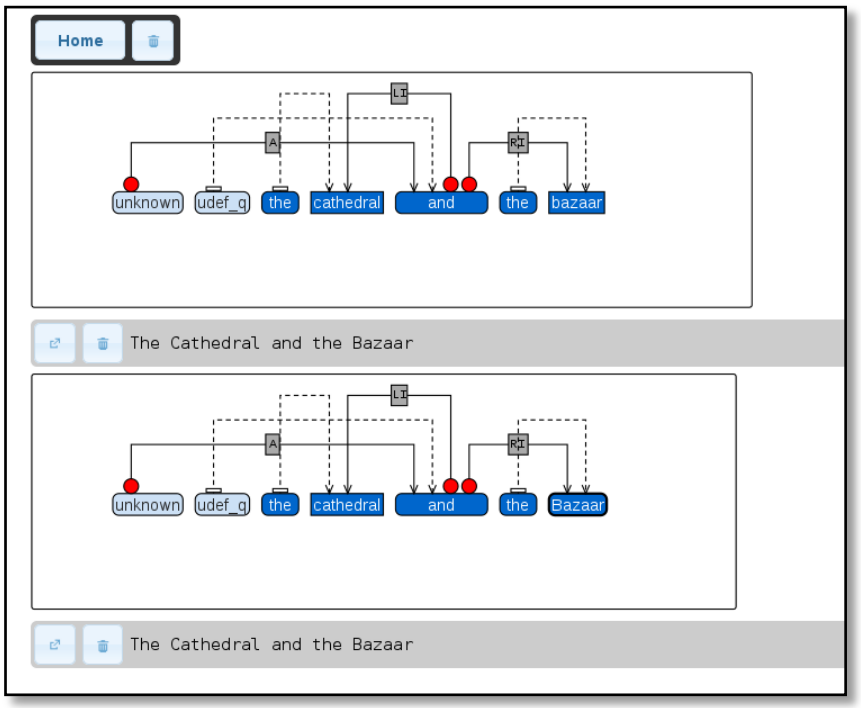

Figure 15: DMRS comparison feature

Figure 16: Corpus display

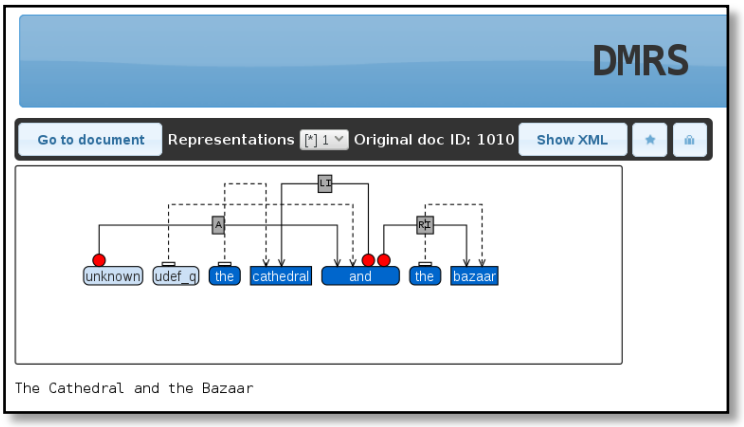

Figure 17: DMRS visualisation

Figure 18: Document view

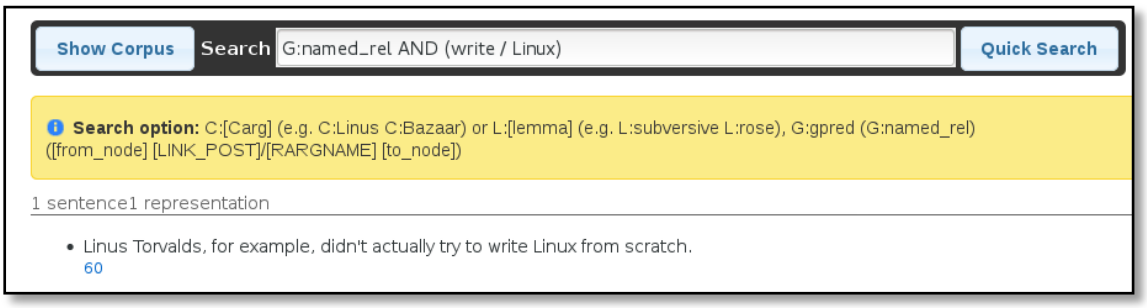

Figure 19: DMRS Query Language

5 Source code

The source code of this project has been released under General Public License version 3.0. We believe that releasing the source code to the public is the best way to contribute to our community. The source code can be obtained by two ways:

- Download compressed package from: https://code.google.com/p/visual-kopasu/downloads/list
- Use Git version control tool to clone the repository by the following command: `git clone https://code.google.com/p/visual-kopasu/`

6 Further Research

6.1 WordNet & DMRS Integration

At the time we write this report, WordNet integration specification in DMRS specification is almost finalised. In fact, we have received the test data and the XML DTD specification to implement this feature in our project. However due to the limitation of time, we have postponed this development into the next phase of our project.

6.2 Full implementation of DMRS Query Language

DMRSQL proposed in our system is only a proof of concept and lacks of many important features. In this prototype, DMRSQL can only search for node pattern, single-link pattern and Boolean AND operator. In the next step we'll implement other Boolean operators (OR, NOT) together with links chaining to deliver richer search patterns.

6.3 Better graph search algorithm

DMRS searching on a corpus as big as our ACL corpus is performing millions of small & disconnected graph search tasks. To make this solution becomes more scalable, we should implement better graph search algorithm. A good research direction is small graph search tasks require more of I/O speed and parallel processing capability rather than the power of a single processing core. We have attempted to enhance the current search clusters with parallel processing into distributed architecture but there're still a lot of works to be done. One of them is setting up a grid computing farm to plug our search clusters in. This can be a future research.

6.4 Consideration of using other database technologies

Recently there are a lot of developments in computer science technologies of how to handle complex and big data. Since our DMRS representation is graph in nature, using graph database engine (Neo4J) instead of relational database engine might lead to better performance. Another possibility is to research distributed database engine such as Apache HBase to handle our knowledge base. Apache HBase's goal is "the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware" (Apache HBase n.d.). These two directions are very promising and we'll investigate them in the next phase of this project.

7 Conclusion

By implementing this project, we have achieved two important results. First, we contributed to the development of computational linguistics with our DMRS visualisation tool, a much easier way to do grammar comparison, and a mechanism to store and search a large number of DMRS graphs. Second, we have demonstrated that question-answering is practically possible with our DMRS persistent framework and DMRSQL.

8 Acknowledgement

We would like to express our most sincerely thanks to the following people:

- Professor Francis Bond who has guided us with various literatures, resources, toolkits, connections and valuable comments throughout the project.
- Our supervisor Ms Fan Zhenzhen and Mr Mathieu Morey who also helps us with a lot of literature and explanation, especially during the knowledge acquisition phase.
- Other researchers in DELPH-IN group:
    - Dr Ulrich Schaefer who has kindly provided us the ACL corpus.
    - Mr Michael Wayne Goodman who has published the open source tool for MRS-DMRS conversion
    - Ms Zina Pozen who helped us with DMRS-WordNet integration specification and test data.

9 References

Ann, Copestake. “Slacker semantics: why superficiality, dependency and avoidance of commitment can be the right way to go.” *Proceedings of the 12th Conference of the European Chapter of the ACL*, 3 April 2009: 1–9.

Ann, Copestake, Flickinger Dan, Pollard Carl, and A. Sag Ivan. “Minimal Recursion Semantics: An Introduction.” *Research on Language and Computation*, 2005: 281–332.

Apache HBase. n.d. http://hbase.apache.org/ (accessed April 1, 2013).

Bob, New. *Question Answering at TREC, Pre-Internship Report.* University of Washington, Ling 590, March 25, 2008.

Dan, Flickinger, Oepen Stephan, and Ytrestøl Gisle. “WikiWoods: Syntacto-Semantic Annotation for English Wikipedia.” *LREC 2010.* European Language Resources Association, 2010. 1665-1671.

Daniel, Jurafsky, and H. Martin James. *Speech and Language Processing.* 2nd. Pearson Education, Inc., 2009.

Delph-in PET. n.d. http://moin.delph-in.net/PetTop (accessed April 1, 2013).

George, A. Miller, Beckwith Richard, and Fellbaum Christiane. “Introduction to WordNet: An On-line Lexical Database.” August 1993.

Kostadin, Cholakov. “Minimal Recursion Semantics (Presentation slides).” n.d.

LinGO ERG. *English Resource Grammar.* n.d. http://www.delph-in.net/erg/ (accessed April 01, 2013).

Sanda, Harabagiu, and Moldovan Dan. *The Oxford handbook of Computational Linguistics.* Oxford University Press, 2004.

WordNet 3.0 database statistics. n.d. (accessed April 01, 2013).

10 Appendix A: Text – MRS – DMRS

10.1 Sentence text

"Linux is subversive."

10.2 MRS representation

```
<mrs>
<label vid='1'/><var vid='2'/>
<ep cfrom='0' cto='5'><pred>PROPER_Q_REL</pred>
    <label vid='3'/>
    <fvpair><rargname>ARG0</rargname><var vid='5' sort='x'>
    <extrapair><path>PERS</path><value>3</value></extrapair>
    <extrapair><path>NUM</path><value>SG</value></extrapair>
    <extrapair><path>IND</path><value>+</value></extrapair></var></fvpair>
    <fvpair><rargname>RSTR</rargname><var vid='4' sort='h'></var></fvpair>
    <fvpair><rargname>BODY</rargname><var vid='6' sort='h'></var></fvpair>
</ep>
<ep cfrom='0' cto='5'><pred>NAMED_REL</pred><label vid='7'/>
    <fvpair><rargname>ARG0</rargname><var vid='5' sort='x'>
    <extrapair><path>PERS</path><value>3</value></extrapair>
    <extrapair><path>NUM</path><value>SG</value></extrapair>
    <extrapair><path>IND</path><value>+</value></extrapair></var></fvpair>
    <fvpair><rargname>CARG</rargname><constant>Linux</constant></fvpair>
</ep>
<ep cfrom='9' cto='20'><spred>_subversive_a_1_rel</spred><label vid='8'/>
    <fvpair><rargname>ARG0</rargname><var vid='2' sort='e'>
    <extrapair><path>SF</path><value>PROP</value></extrapair>
    <extrapair><path>TENSE</path><value>PRES</value></extrapair>
    <extrapair><path>MOOD</path><value>INDICATIVE</value></extrapair>
    <extrapair><path>PROG</path><value>-</value></extrapair>
    <extrapair><path>PERF</path><value>-</value></extrapair></var></fvpair>
    <fvpair><rargname>ARG1</rargname><var vid='5' sort='x'>
    <extrapair><path>PERS</path><value>3</value></extrapair>
    <extrapair><path>NUM</path><value>SG</value></extrapair>
    <extrapair><path>IND</path><value>+</value></extrapair></var></fvpair>
</ep>
<hcons hreln='qeq'><hi><var vid='4' sort='h'></var></hi><lo><var vid='7'
sort='h'></var></lo></hcons>
</mrs>
```

10.3 DMRS representation

```
<dmrs cfrom='-1' cto='-1'>
     <node nodeid='10001' cfrom='0'
     cto='5'><gpred>proper_q_rel</gpred><sortinfo/></node>
     <node nodeid='10002' cfrom='0' cto='5'
     carg='Linux'><gpred>named_rel</gpred><sortinfo cvarsort='x' pers='3' num='sg'
     ind='plus'/></node>
     <node nodeid='10003' cfrom='9' cto='20'><realpred lemma='subversive' pos='a'
     sense='1'/><sortinfo cvarsort='e' sf='prop' tense='pres' mood='indicative'
     prog='minus' perf='minus'/></node>

     <link from='10001' to='10002'><rargname>RSTR</rargname><post>H</post></link>
     <link from='10003' to='10002'><rargname>ARG1</rargname><post>NEQ</post></link>
</dmrs>
```