CINDI_QA:
A TEMPLATE-BASED BILINGUAL QUESTION ANSWERING SYSTEM

Chedid Haddad

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science
Concordia University
Montreal, Quebec, Canada

April 2008

Your file Votre référence
ISBN: 978-0-494-40939-8
Our file Notre référence
ISBN: 978-0-494-40939-8

Abstract

CINDI_QA: A Template-Based Bilingual Question Answering System

by

Chedid Haddad

CINDI_QA is the result of the CINDI group's efforts at tackling multilingual Question Answering. The goal of the system is to receive a French question and return the answer to that question in English.

This thesis outlines the architecture of CINDI_QA and how it is integrated with several open-source tools. Google Translate is used to produce the English equivalent of the French question. The Link Parser is a semantic parser of English that identifies keywords such as nouns, verbs and adjectives. WordNet is a lexical analyzer that generates synonyms of those keywords. Finally, Lucene handles indexing and searching of the large data collection made up of Wikipedia static pages.

In addition to these tools, CINDI_QA heavily relies on templates to better understand the question entered by the user. Six templates help the system answer questions about different subjects.

The performance of CINDI_QA has been assessed by participating in the 2007 edition of the QA@CLEF competition, an annual conference focusing on multilingual QA, to which the CINDI group submitted two runs that ranked second and third out of eight candidates.

Acknowledgements

I would like to take this opportunity to thank all the people that have helped me through the completion of this project.

First, Prof. Bipin C. Desai truly embodies the meaning of a dedicated supervisor. His door has always been open for suggestions, encouragement and support. He even gave me one of the greatest pieces of advice I have ever heard: "the only way to learn how to swim is to get into the water".

I would like to thank Niculae Stratica for finding time amid his busy work schedule to share with me his experience at Concordia and introducing me to his thesis project that was a great inspiration for mine.

The CINDI group is made of talented and hard-working individuals who constantly encourage each other. Thanks to Varun Sood, Mahmudur Rahman, Krishma Dutta and Adnan Abueid for creating a great working environment in our lab office.

My deepest recognition goes to my girlfriend Tayma for her daily patience, support and encouragement. Without her, this project wouldn't have reached fruition.

Last but certainly not least, I thank my family for providing me with the education, morals, values and confidence that have helped me get through everyday life. Even though they are geographically far, I have felt them by my side every single day.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Program Interface
CE	Capitalized Entity
CINDI	Concordia Index for Navigation and Discovery on the Internet
CINDI_QA	CINDI Question Answering System
CLEF	Cross Language Evaluation Forum
EN	English
FLIP	Formal List Processor
FR	French
GUI	Graphical User Interface
IR	Information Retrieval
JAR	Java Archive
JDK	Java Development Kit
JNCLGI	Java Native Code Link Grammar Interface
JRE	Java Runtime Environment
JSP	Java Server Pages
LISP	List Processing Language
NL	Natural Language
NLP	Natural Language Processing or Parsing
NLPQC	Natural Language Processor for Querying CINDI
OO	Object Oriented
QA	Question Answering

QA@CLEF	Multilingual Question Answering track at CLEF
RAM	Random Access Memory
SQL	Structured Query Language
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Overview

The Concordia Index for Navigation and Discovery on the Internet (CINDI) group has been founded at the Department of Computer Science and Software Engineering of Concordia in the late 1990s [1]. Its purpose is the continuous enhancement of information discovery and retrieval. CINDI_QA is a bilingual Question-Answering system within the spectrum of the CINDI group's research area [2].

CINDI_QA's goal is to receive a question in French and to return the answer to that question in English. The system only interacts with two external entities: the user, from whom it receives the French question as input and to whom it delivers the English answer, and a web-based translator that it uses to obtain the English equivalent to the French question. Figure 1.1 shows the activity diagram of CINDI_QA as a black box.

In addition to the online translator, CINDI_QA makes use of several tools to come up with an acceptable answer. First, a Natural Language Parsing (NLP) module is responsible for understanding the English question and identifying keywords which are nouns, verbs, adjectives and Capitalized Entities (CE). Section 3.3 elaborates more on this NLP tool.

Then, a Lexical Reference tool is used to produce synonyms of certain keywords so they can be included in the search process, as mentioned in section 3.4.

Finally, a search engine provides indexing of the data collection to efficiently search through it. This searching and indexing tool is discussed extensively in section 3.5.

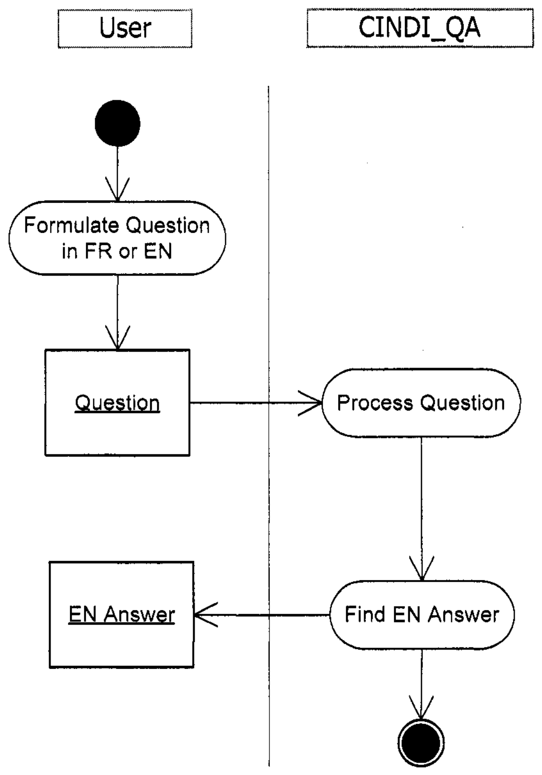


Figure 1.1: CINDI_QA Activity Diagram

CINDI_QA also makes use of templates in order to formulate the correct answer. Templates are constructs that define the structure of several similar questions as one prototype. Chapter 4, which is entirely dedicated to this process, lists all six templates and explains how they help CINDI_QA produce the best possible answer.

1.2 Problem Statement

Question-Answering (QA) systems are very challenging to create, especially bilingual ones, where the language used to ask a question differs from the language of the information corpora and the expected answer.

First, QA systems must be able to understand the question asked by the user. Several approaches have been used for this purpose, ranging from the creation of templates to match the input to the use of Natural Language Processing (NLP) on a question.

Second, they must be able to efficiently find an answer without taking too much time to return it.

Third, finding an answer is not always enough because QA systems are more than search engines and must return to the user the located piece of text formulated in a way that answers the question the way a human would.

Finally, bilingual QA systems must possess the added functionality of a consistent translation module to convert the question into the language of the data collection used.

1.3 Proposed Solution

CINDI_QA addresses these issues. First, CINDI_QA benefits from two separate modules to understand the question once it has been translated to English. On the one hand, a template module contains six different templates that match specific questions asked by the user. This allows the system to know exactly what the user is asking about. On the other hand, a NLP tool is used by CINDI_QA in the case when no template is matched to identity keywords from the question. This process provides CINDI_QA with some knowledge about the intent of the question.

Second, CINDI_QA employs an efficient indexing and searching tool that provides an array of functionalities related to searching and ranking results while still being extremely fast.

Third, CINDI_QA formulates the answer to a question it receives as if it was answered by another human being in the case when a template is matched. In the other case, CINDI_QA returns a snippet of text containing the answer.

Fourth, CINDI_QA somewhat reduces the ambiguity of translation by having its templates stored both in French and English inside its code. This technique allows the system to understand French questions as if they were posed in English when they match a template.

CINDI_QA has a unique architecture that integrates all of these components in a bilingual environment.

1.4 Organization of Thesis

This thesis is organized as follows. Chapter 2 covers what has previously been done both in the fields of Natural Language Processing and Question Answering. The next chapter details the architecture of CINDI_QA and lists the different modules that define it; these are: the Online Translator, the Natural Language Parser, the Lexical Reference and the Document Searching and Indexing module. As for the Template module, chapter 4 is entirely dedicated to it and illustrates each template and what mechanism drives them. Chapter 5 is about the implementation of the project and how all the modules fit nicely together. In chapter 6, we share our experimental results as well as our participation in the 2007 edition of the QA@CLEF task. Finally, chapter 7 concludes this document and suggests what can be done to improve CINDI_QA.

Chapter 2

Previous Work in the Field

2.1 State of the Art

The concept of Information Retrieval (IR) has always been a fascination of mankind, long before it became associated with computing. IR has always consisted of obtaining the most relevant data about a specific subject, but since the dawn of the Digital Age, it has become more and more popular and complex. It is now defined as the science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hypertextually-networked databases such as the World Wide Web [3].

Question-answering (QA) is a type of IR, where a system retrieves answers from a data collection to a question posed in Natural Language (NL). By NL we mean a language that is spoken or written by humans for everyday communication purposes.

Natural Language Processing (NLP) is the process of understanding NLs. NLP is closely related to QA because most QA systems make extensive use of NLP to produce answers.

Because CINDI_QA is a bilingual QA system that uses NLP as one of its driving mechanisms, we shall go over NLP and QA a little deeper by discussing previous successful NLP and QA projects while also mentioning the challenges of multilingual QA.

2.2 Natural Language Processing Domain

2.2.1 Early NLP Attempts

NLP has been around ever since the start of the Digital Age. Early efforts of NLP concentrated on matching NL questions to queries on pre-set databases. An example of this is Formal List Processor, or FLIP [4], an early language for pattern-matching on LISP (List Processing Language) structures. FLIP is able to build a query for the database if the input matches one of its available patterns, as show in table 2.1.

Input	**Matching Pattern**
What is the capital of all countries?	get attribute 'capital' of table 'country'
What is the capital of Canada?	get attribute 'capital' of table 'country' where country = Canada

Table 2.1: FLIP pattern-matching examples

FLIP, being as old as it is, didn't benefit from a high-level implementation so its code and database details were intertwined, making it highly dependent on the range of the matching patterns as well as its database scope. Therefore, FLIP wasn't able to really process the input to understand it; it only tried to match the question directly to the set of available patterns.

2.2.2 The LUNAR System

Later systems did however introduce syntax-based techniques, where the user input is first parsed to understand its meaning before generating a query for the database. One of the best known early NL interfaces is the Lunar Sciences Natural Language Information System – LNSLIS, better know simply as LUNAR [5]. LUNAR is a computer system that supports English language access to a large database of lunar sample information. It allows a lunar geologist to ask questions whose answer lies in a collection of 13,000 chemical analyses of lunar samples and 10,000 document postings. LUNAR relies on a language processing component that has a lexical reference of about 3500 words, a transition network grammar of a good portion of English combined with a set of semantic interpretation rules for translating English input requests into formal database procedures. All the components of the system were implemented in LISP on the Programmed Data Processor model 10 (PDP-10) mainframe computer under the TENEX time-sharing operating system. TENEX used special paging hardware and possessed a virtual core memory of 256 KB. All the equipment was owned by Bolt, Beranek and Newman (BBN) Technologies located in Cambridge, Massachusetts, USA.

The sample behavior of LUNAR is as follows:

```
Request:
(DO ANY SAMPLES HAVE GREATER THAN 13 PERCENT ALUMINUM)
Processing time:
PARSING: 4.614 SECONDS
INTERPRETING: 3.566 SECONDS
```

Parsed into query language:

(TEST (FOR SOME X1 / (SEQ SAMPLES) : T ; (CONTAIN X1 (NPR* X2 / 'AL203) (GREATERTHAN 13 PCT))))

Response:

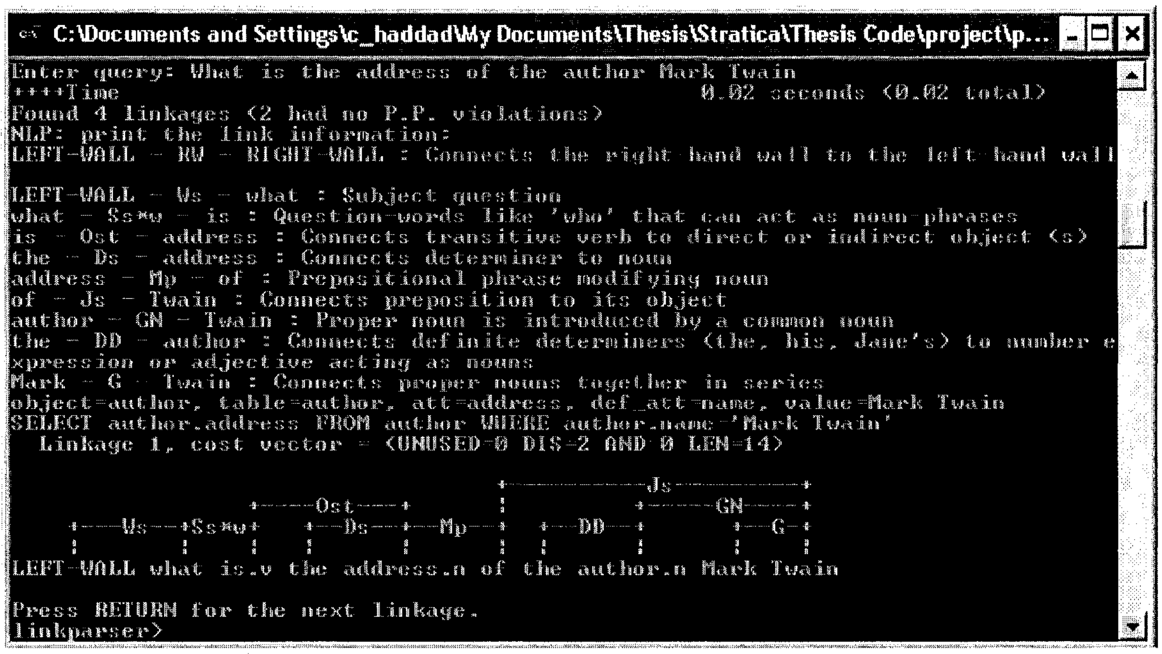
YES

As with FLIP, the main inconvenient of LNSLIS is that it is strictly specific to the lunar domain. It cannot do anything when prompted for trivia outside its scope. In addition, while its processing time surely was impressive during the seventies, waiting almost 10 seconds for an answer isn't up to the standards of today.

2.2.3 The NLPQC System

The Natural Language Processor for Querying CINDI (NLPQC) system is a thesis project implemented by a fellow CINDI group member, Niculae Stratica, in 2002 [6]. At the time, CINDI was the name given to the Concordia Digital Library System, a database that stores information related to books, such as authors' name and titles of bibliographic references, using Semantic Headers [7]. The purpose of the NLPQC system is to semantically parse natural language questions in English and build corresponding SQL queries for use against the CINDI database. It achieves this goal by using rules and templates for the semantic parsing and two open-source tools, WordNet and the Link Parser. WordNet is a lexical reference of the English language, i.e., it generates synonyms of any given word while the Link Parser is a syntactic parser able to identify nouns, adjectives and verbs as well as the global syntactic structure given an English sentence. For more information on those tools, refer to appendix A for the Link Parser and appendix B for WordNet.

Figure 2.1 shows the NLPQC system in action. The question asked is "What is the address of the author Mark Twain" and the correct SQL equivalent query is generated.


C:\Documents and Settings\c_haddad\My Documents\Thesis\Stratica\Thesis Code\project\p...

```
Enter query: What is the address of the author Mark Twain
++++Time                                          0.02 seconds (0.02 total)
Found 4 linkages (2 had no P.P. violations)
NLP: print the link information:
LEFT-WALL - RW - RIGHT-WALL : Connects the right hand wall to the left hand wall

LEFT-WALL - Ws - what : Subject question
what - Ss*w - is : Question-words like 'who' that can act as noun-phrases
is - Ost - address : Connects transitive verb to direct or indirect object (s)
the - Ds - address : Connects determiner to noun
address - Mp - of : Prepositional phrase modifying noun
of - Js - Twain : Connects preposition to its object
author - GN - Twain : Proper noun is introduced by a common noun
the - DD - author : Connects definite determiners (the, his, Jane's) to number e
xpression or adjective acting as nouns
Mark - G - Twain : Connects proper nouns together in series
object=author, table=author, att=address, def_att=name, value=Mark Twain
SELECT author.address FROM author WHERE author.name='Mark Twain'
   Linkage 1, cost vector = (UNUSED=0 DIS=2 AND=0 LEN=14)

                                  +----------Js----------+
                +----Ost----+     |         +-----GN-----+
    +--Ws--+Ss*w+   +--Ds---+-Mp--+   +-DD--+      +--G--+
    |      |    |   |       |     |   |     |      |     |
LEFT-WALL what is.v the address.n of the author.n Mark Twain

Press RETURN for the next linkage.
linkparser>
```

Figure 2.1: NLPQC System

CINDI_QA is greatly inspired by the NLPQC system on many levels. First, it borrows the idea of having pre-defined templates that can match the input question for more efficient answer selection, as is explained in chapter 4. In addition, CINDI_QA also employs WordNet and the Link Parser to better understand the question; the integration of both these tools is covered in chapter 3.

2.3 Question Answering Domain

2.3.1 Early QA Attempts

There is a subtle difference between the goals of IR and one of its main branches, QA. Given a query, an IR system returns a list of potentially relevant documents which the user must then scan to search for pertinent information. This method could not satisfy the user's needs to efficiently extract the adequate information from a huge set of electronic documents. On the other hand, QA is a technology that aims at retrieving the answer to a question written in natural language in large collections of documents. QA systems are presented with natural language questions and the expected output is either the exact answer identified in a text or small text fragments containing the answer. The difference is shown in table 2.2.

	IR Technology	**QA Technology**
Input	Some Keywords	Question in Natural Language
Output	Document List	Correct Answer or Words that Include the Answer

Table 2.2: Difference between IR and QA Technologies

The general QA approach is to have the system provide some facility for analyzing the question and to understand what is being asked, the former part is known as NLP as we have seen in the previous section. The system must also be able to quickly and efficiently search for documents or passages relevant to the question, in order to locate candidate answers. Finally, it needs to determine the correctness of these answers and choose the best among them to return to the user. Figure 2.2 displays the main components of such a general architecture, and the ways in which they interact. The prototypical QA system has four components: question analysis, document retrieval, document analysis and answer selection.

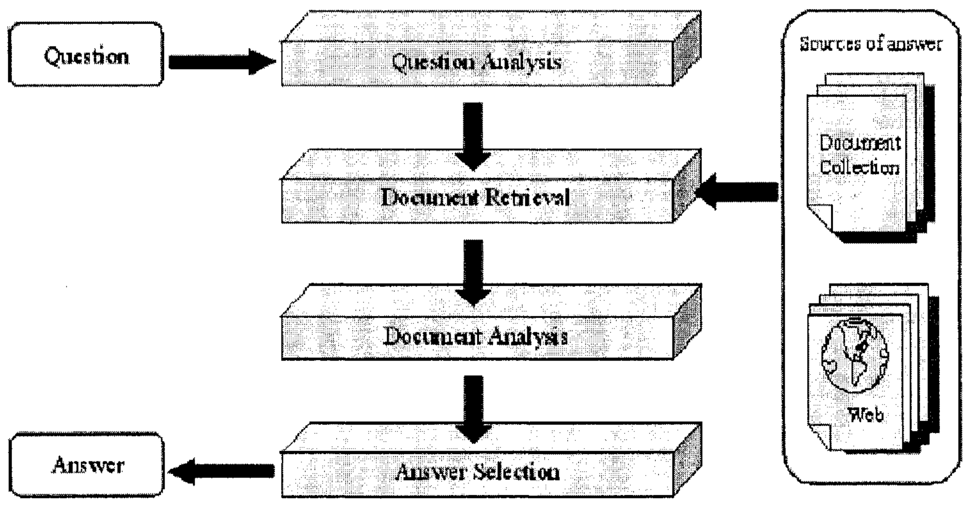


Figure 2.2: QA General Approach

In order to build an efficient QA system, one has to take into account several challenges, some of which are:

- **Existence of Answer**: Before a question is answered, the knowledge sources must be examined in the event the answer doesn't exist in that collection. If this is the case, then the system must have the ability of determining that there is no correct answer or else any result would be incorrect no matter how well question processing and answer retrieval are accomplished.

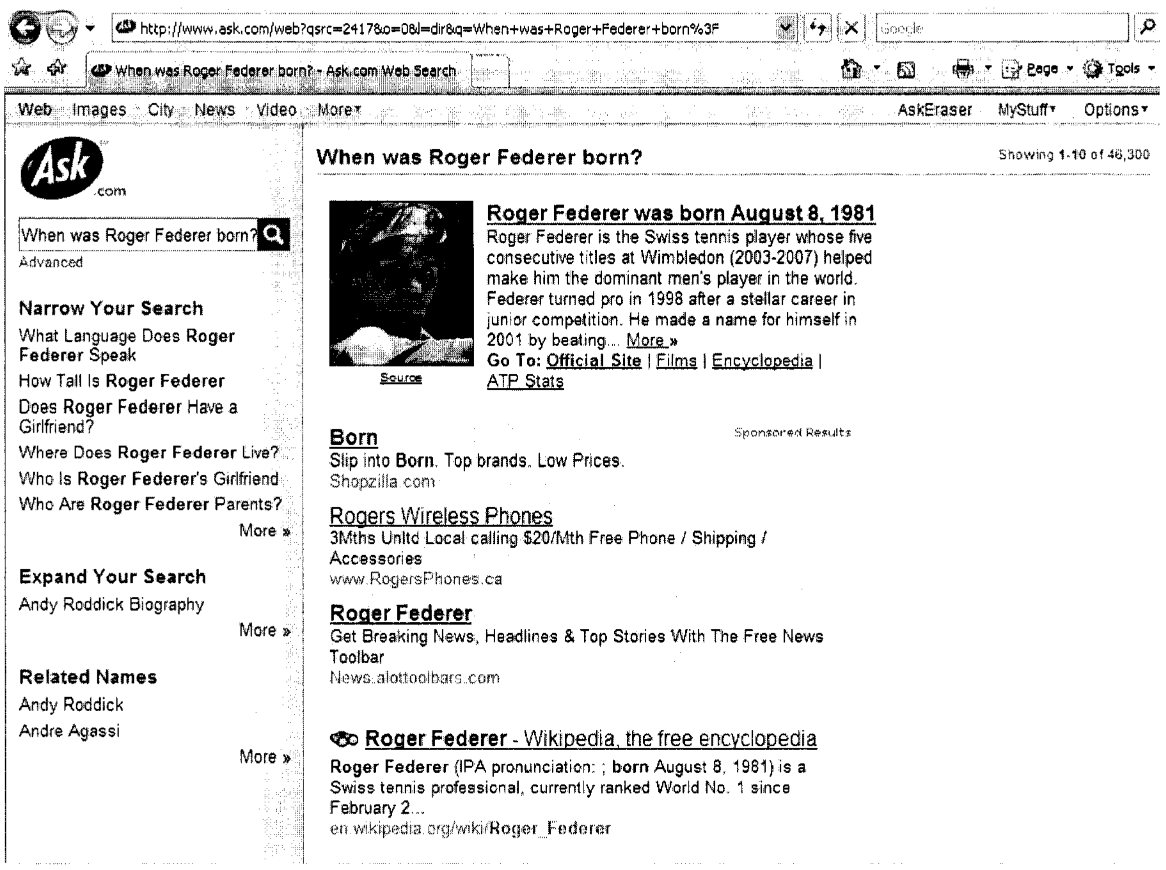
- **Answer formulation**: QA should not only concentrate on finding the correct answer but also expressing it in the same way a Human Being would. This can be trivial for such questions as "When did the Berlin Wall fall?" – where only a date is expected as an answer – but becomes harder when answering questions like "Why did the Berlin Wall fall?" in which case the answer should start with the word "because" and be followed by a complete sentence.
- **QA time performance**: Regardless of the complexity of the question asked and the size of the data collection queried, a good QA system should be able to consistently produce an answer fairly quickly, not exceeding a few seconds.
- **Multilingual QA**: Although English is by far the language in which the most data sources are available throughout the world, other languages shouldn't be neglected. Inquiring in English about documents written in another language and querying English documents in another language should both be supported, even though they add the burden of cross-lingual translation and comprehension to the already large list of challenges of monolingual QA.

The history of QA is almost as old as the field of Computer Science. Naturally, humans have always wondered if a machine could become as or even more intelligent than them ever since the advent of the computer, as depicted in several science fiction movies such as The Terminator and The Matrix trilogies.

The notion of a QA system was born in 1950 when Alan Turing proposed a task he called the "Imitation Game Test" that eventually became known as the famous "Turing Test" in which a human communicates with a machine, most often via a teletype interface for judgment purposes, and asks questions to it [8]. Turing would consider the machine "intelligent" if the human interrogator wouldn't be able to differentiate between the responses of the machine and those of another human.

2.3.2 The Web-based Ask.com System

QA systems have come a long way since then and are now most commonly implemented as web-based interfaces. Ask.com is a reliable compromise between search engines and QA systems. Formerly known as "Ask Jeeves" [9], Ask.com retrieves the answer from the internet and lists all the web-links that mention the answer string. This is done thanks to their "ExpertRank" algorithm that provides relevant search results by identifying the most authoritative sites on the web. Ask.com goes beyond mere link popularity (which ranks pages based on the sheer volume of links pointing to a particular page) to determine popularity among pages considered to be experts on the topic of the question asked, known as subject-specific popularity. So by identifying topics, the experts on those topics and the popularity of millions of pages amongst those experts all at the exact moment the search query is conducted, Ask.com offers a result that is most of the time accurately superior to other systems. A sample answer from Ask.com is shown in figure 2.3.


http://www.ask.com/web?qsrc=2417&o=0&l=dir&q=When+was+Roger+Federer+born%3F

When was Roger Federer born? - Ask.com Web Search

Web Images City News Video More AskEraser MyStuff Options

When was Roger Federer born?

Showing 1-10 of 46,300

Advanced

Narrow Your Search

What Language Does Roger Federer Speak
How Tall Is Roger Federer
Does Roger Federer Have a Girlfriend?
Where Does Roger Federer Live?
Who Is Roger Federer's Girlfriend
Who Are Roger Federer Parents?
More »

Expand Your Search

Andy Roddick Biography
More »

Related Names

Andy Roddick
Andre Agassi
More »

Roger Federer was born August 8, 1981
Roger Federer is the Swiss tennis player whose five consecutive titles at Wimbledon (2003-2007) helped make him the dominant men's player in the world. Federer turned pro in 1998 after a stellar career in junior competition. He made a name for himself in 2001 by beating... More »
Go To: Official Site | Films | Encyclopedia | ATP Stats

Source

Sponsored Results

Born
Slip into Born. Top brands. Low Prices.
Shopzilla.com

Rogers Wireless Phones
3Mths Unltd Local calling $20/Mth Free Phone / Shipping / Accessories
www.RogersPhones.ca

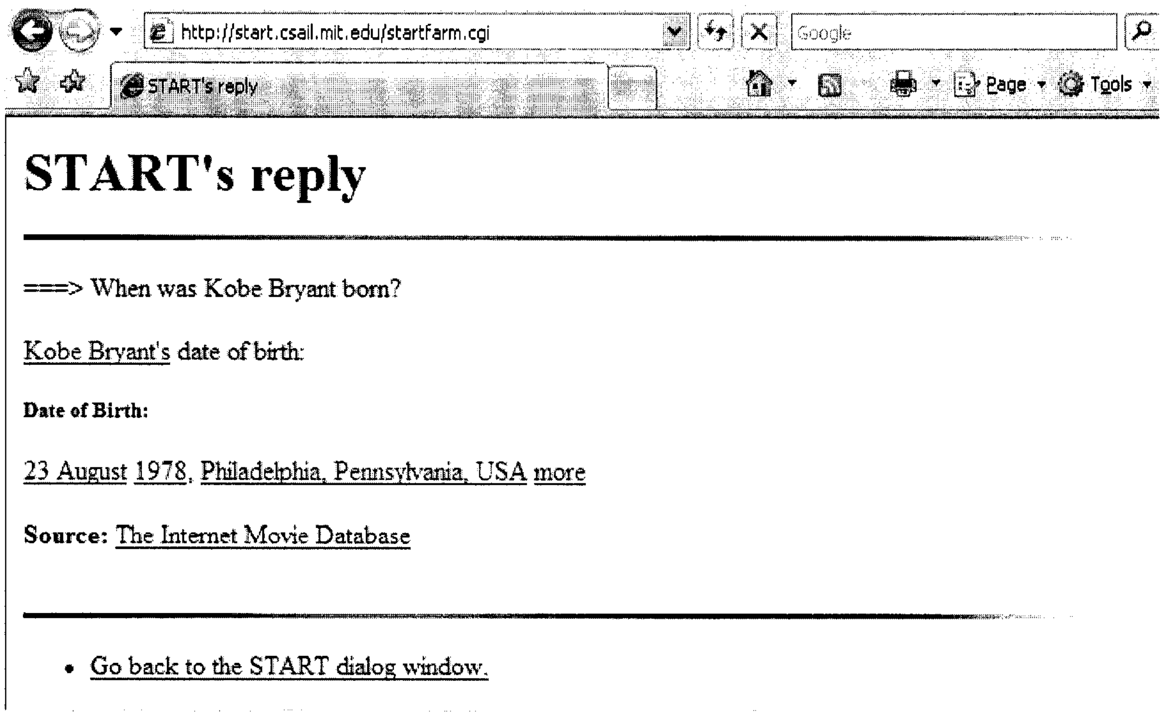
Roger Federer
Get Breaking News, Headlines & Top Stories With The Free News Toolbar
News.alottoolbars.com

Roger Federer - Wikipedia, the free encyclopedia
Roger Federer (IPA pronunciation: ; born August 8, 1981) is a Swiss tennis professional, currently ranked World No. 1 since February 2...
en.wikipedia.org/wiki/Roger_Federer

Figure 2.3: Ask.com Web-based QA System

2.3.3 The Web-based START System

START, the world's first Web-based question answering system, has been on-line and continuously operating since December 1993. It has been developed by Boris Katz and his associates of the InfoLab Group at the MIT Computer Science and Artificial Intelligence Laboratory [10]. Unlike typical information retrieval systems such as search engines, START aims to supply users with just the right information, instead of merely providing a list of hits. Figure 2.4 shows the answer returned by START for the questions "When was Kobe Bryant born?".



START's reply

===> When was Kobe Bryant born?

Kobe Bryant's date of birth:

Date of Birth:

23 August 1978, Philadelphia, Pennsylvania, USA more

Source: The Internet Movie Database

• Go back to the START dialog window.

Figure 2.4: START Web-based QA System

The START system uses T-expressions for the semantic parsing, where T stands for Template. The system uses the pattern <subject relation object> [10]. This approach has the advantage of being intuitive but has a limitation. Sentences with different syntax and close meaning are not considered similar by the system. For example, the phrase "the student surprised the teacher with his answer" is parsed into <student surprises teacher> whereas the phrase "the student's answer surprised the teacher" is parsed into <answer surprises teacher>, which is a different interpretation of the input sentence.

CINDI_QA was inspired by two concepts of the START system. The first is the use of semantic parsing to understand the input question. START parses the question to have a better idea of where to look for the answer, depending on which natural language annotation it matches. CINDI_QA employs a NLP module in order to identify special keywords for later use; section 3.3 has more details on this. The second idea is template-matching. Indeed, CINDI_QA heavily relies on templates to drive its answer extraction and formulation mechanisms. All six templates used by CINDI_QA are elaborated on in chapter 4.

2.4 Multilingual QA

In the previous sections, the systems mentioned were monolingual QA systems, where the source and target languages – the latter being the language of the input question, the former is the language of the document collection containing the answer – are the same. In recent years however, interest has grown over multilingual QA. A multilingual QA system should be able to answer questions regardless of the language used when asking these questions.

The vast majority of multilingual QA efforts are addressed by the Multilingual Question Answering track at the Cross Language Evaluation Forum (QA@CLEF) [11]. Each year since 2003, QA@CLEF invites participants to submit their answers to the same set of questions and compares the results. QA@CLEF now represents the annual benchmark to which multilingual systems from all over the world measure themselves.

QA@CLEF is a prime example of the gap between monolingual and multilingual QA: since its inception, the best correct answering percentage of monolingual QA systems is almost 69% whereas the same for bilingual never surpassed 50%. Moreover, in the 2007 edition of QA@CLEF, the average for monolingual was 22.8% while the average for bilingual was 10.9% [11]. Also, there is a significant drop in both the best and average scores at CLEF07; this is attributed to the fact that 2007 is the first year that Wikipedia pages were used as data collection in addition to the traditional news articles. These numbers are depicted in figure 2.6.

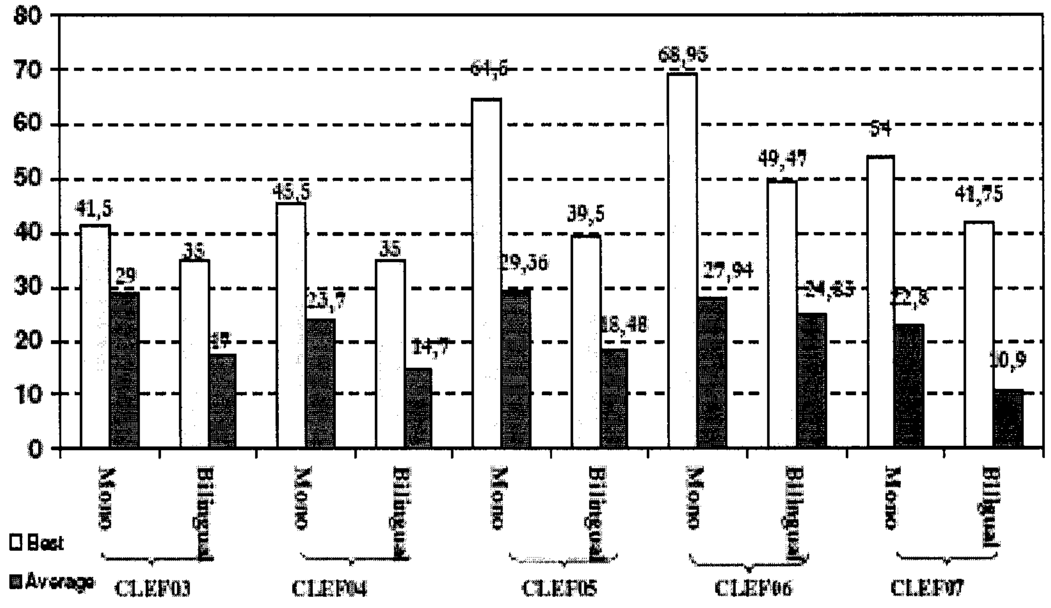


Figure 2.6: Best and Average Scores in QA@CLEF Campaigns

CINDI_QA participated in the 2007 edition of QA@CLEF and the results obtained as well as more information on QA@CLEF can be found in section 6.2.

2.5 CINDI_QA Considerations

CINDI_QA exploits several tools and concepts seen in this chapter. First, in addition to answering questions in English, it can also deal with questions asked in French, making CINDI_QA a bilingual QA system. Second, it relies heavily on templates in order to better understand the input question and come up with a correct answer. All six templates defined by CINDI_QA are explained in chapter 4. Third, it integrates two external tools that were used by the NLPQC system: WordNet and the Link Parser. The Link Parser helps identify keywords and WordNet can generate synonyms of those keywords when needed. The cooperation of these two tools within CINDI_QA is the focus of the next chapter where we talk about the architecture and logic of CINDI_QA.

Chapter 3

CINDI_QA Architecture and Logic

The CINDI_QA system consists of one central unit called the Processor and five separate modules. The Processor has the responsibility of keeping track of the status of the analysis of the input and delegating the work to the specialized modules, which are:

- The Web-based Translator provides the English equivalent to the French input question. More details are given in section 3.2.
- The Natural Language Parsing tool called the Link Parser [12] dissects the English sentence into keywords such as nouns, verbs, and adjectives to better understand the subject of the question. More details are given in section 3.3.
- The Lexical Reference called WordNet [13] identifies the synonyms of the keywords obtained. This is done to broaden the range of the search query performed by the next tool. More details are given in section 3.4.
- The Indexing and Searching module Lucene goes through all the data sources using a query composed of the keywords. It also locates documents that contain candidate answers. More details are given in section 3.5.
- The Template module checks the input question to see if it matches one of the six templates already defined. More details are given in section 3.1.

The advantage of the architecture of CINDI_QA is its adaptability or the ability of the different modules to be replaced by other similar tools without altering the initial design of the system.

Figure 3.1 shows a package diagram detailing the architecture of CINDI_QA. The User Interface (UI) constitutes the top level with the Web portal of the project. The Processor lies in the Application layer with access to both Domain and Services layers containing the four modules mentioned above. Finally, the data collection comprised of Wikipedia content makes up the Foundation layer.

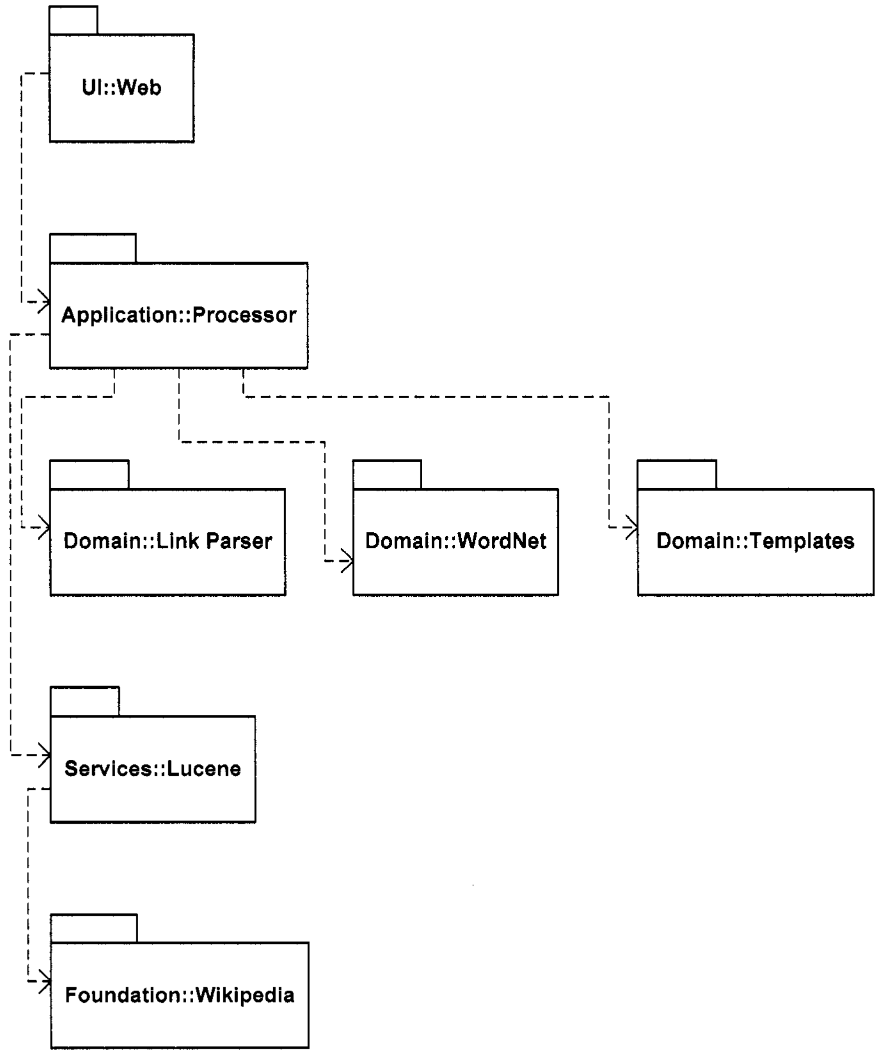


Figure 3.1: CINDI_QA Package Diagram

3.1 Rationale behind Templates

CINDI_QA is strongly dependent on templates. Because questions can vary to query any subject possible, we chose to build templates in order to answer specific types of questions. Although this approach might reduce the scope of the project, it enhances the correctness of the provided answer. In addition, template-matching had already been successfully used in another CINDI project [6].

The Processor decides on which process flow to follow, depending on whether the input question matches a template or not.

If a template is matched, CINDI_QA acts as a black-box requiring no additional input from the user. Instead, the system uses the information from the Template Module to directly search the data collection and find the correct answer. CINDI_QA then returns that answer to the user.

When no template is matched, the Processor delegates the work to the specialized modules that analyze the question in their own way with some help from the user. This collaboration results in CINDI_QA displaying a snippet of text that contains the answer.

Figure 3.2 illustrates the general process flow of CINDI_QA.

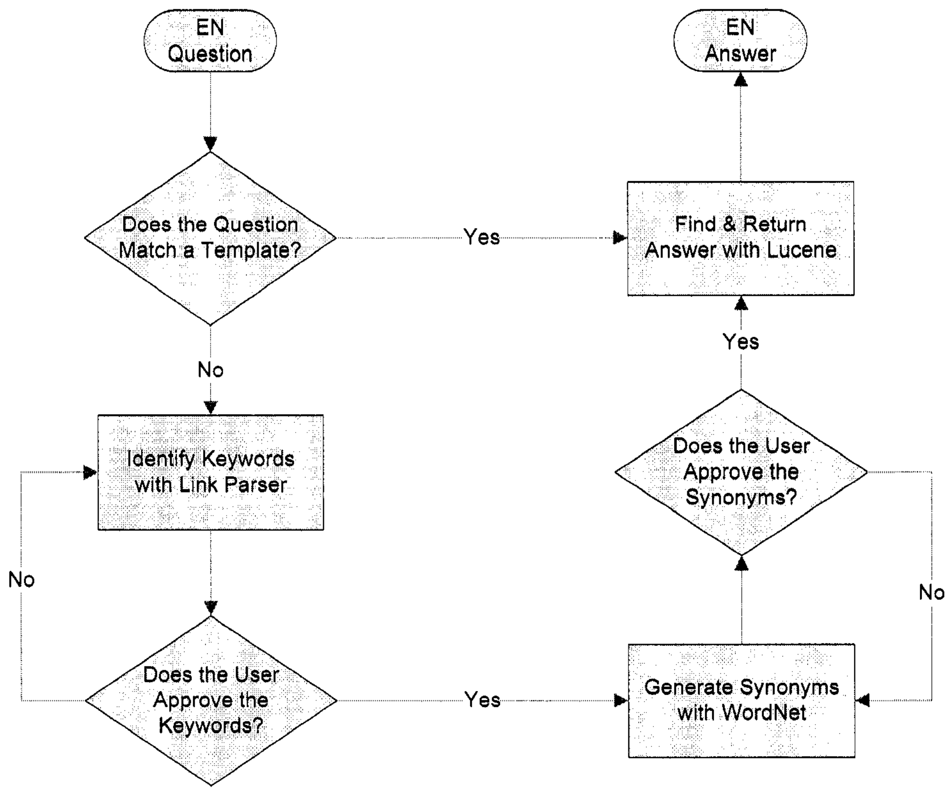

EN Question
Does the Question Match a Template?
Yes
Find & Return Answer with Lucene
EN Answer
No
Identify Keywords with Link Parser
No
Does the User Approve the Keywords?
Yes
Generate Synonyms with WordNet
Does the User Approve the Synonyms?
Yes
No

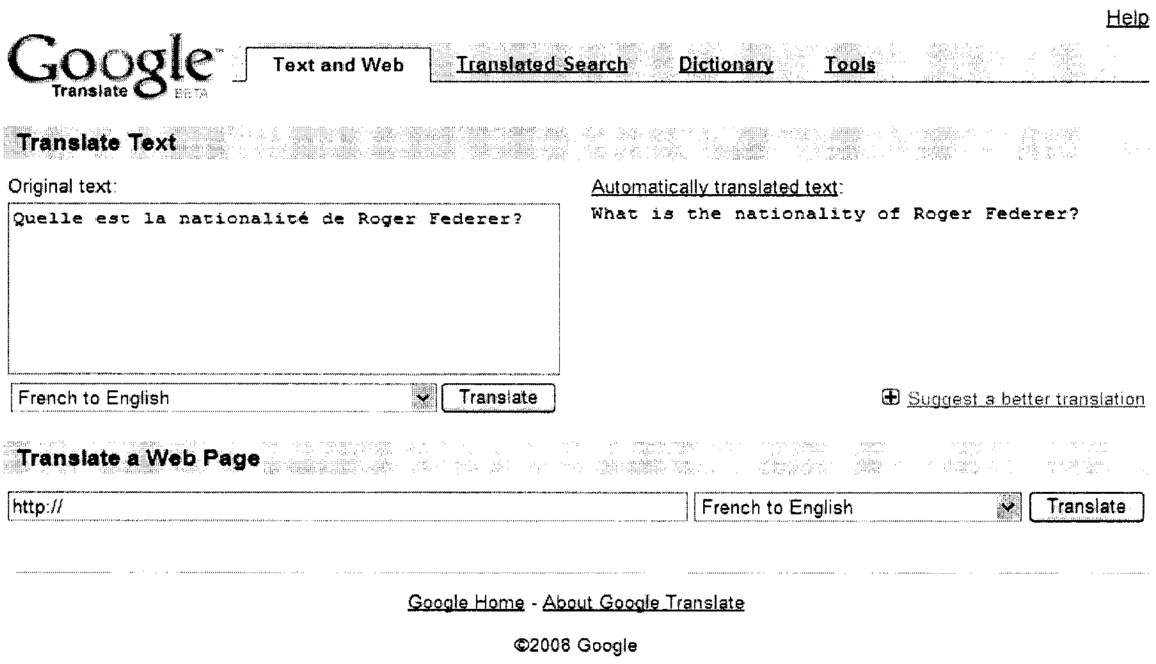
Figure 3.2: CINDI_QA Process Flow

3.2 Online Translation Module

Since we are working in a bilingual environment, the system is queried in a language different from the data collection it is using as reference. A translation tool is needed for this reason. After researching the available tools, we noticed that Google [14], Babel Fish [15] and Systran [16] translators are all powered by the same engine: Systran.

We chose to use Google Translate in our system due to its better interface and speed of processing. A JSP script is responsible of delivering the French question typed in by the user to the Google Translate webpage and bringing back the translated English equivalent to the CINDI_QA Processor.

Figure 3.3 shows the web interface of Google Translate after asking it to translate a French sentence to English.



Quelle est la nationalité de Roger Federer?

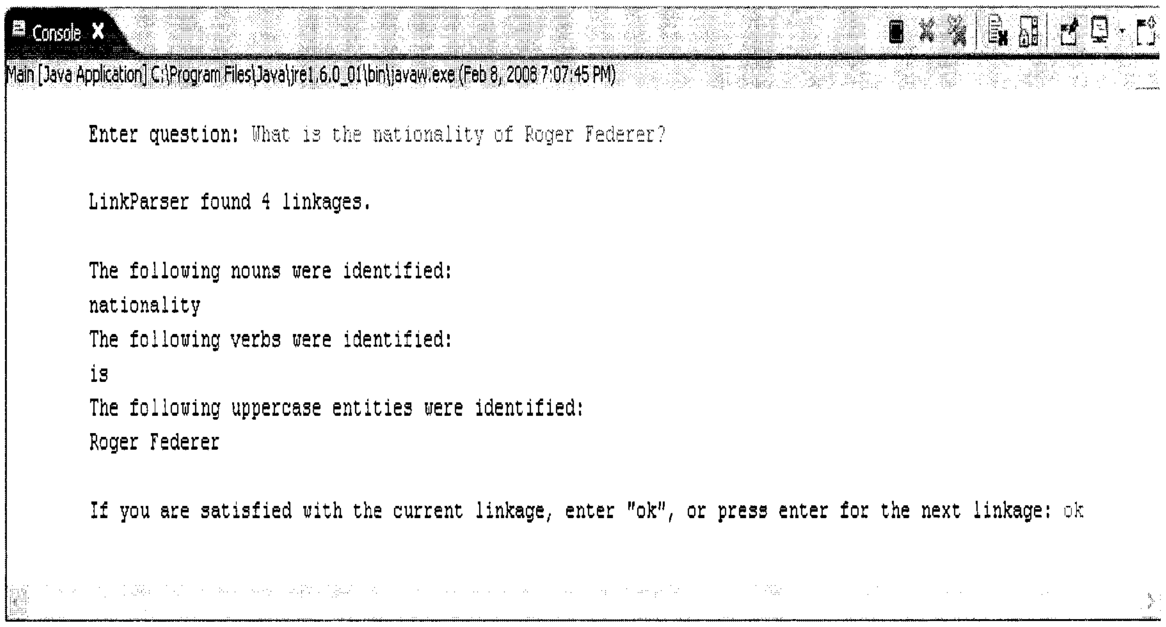
What is the nationality of Roger Federer?

Figure 3.3: Google Translate

3.3 Natural Language Parsing Module

We need a way to understand the question asked by the user in order to single out the keywords. This was achieved thanks to the Link Grammar Parser [12], a syntactic parser of English based on link grammar, an original theory of English syntax. Given a sentence, the Link Parser assigns to it a syntactic structure which consists of a set of labeled links connecting pairs of words. Each structure is called a linkage and several linkages are generated for the same input. The Link Parser is plugged into our system to generate linkages of a translated English question. Using one linkage, we are able to determine which words form nouns, verbs, adjectives and Capitalized Entities. If those keywords appear wrong or incomplete, we go on to the next linkage. The user has the option of choosing the most appropriate linkage in the case where no template was matched.

Figure 3.4 demonstrates the use of the Link Parser within CINDI_QA.


Console
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 8, 2008 7:07:45 PM)

```
Enter question: What is the nationality of Roger Federer?

LinkParser found 4 linkages.

The following nouns were identified:
nationality
The following verbs were identified:
is
The following uppercase entities were identified:
Roger Federer

If you are satisfied with the current linkage, enter "ok", or press enter for the next linkage: ok
```

Figure 3.4: LinkParser in CINDI_QA

3.4 Lexical Reference Module

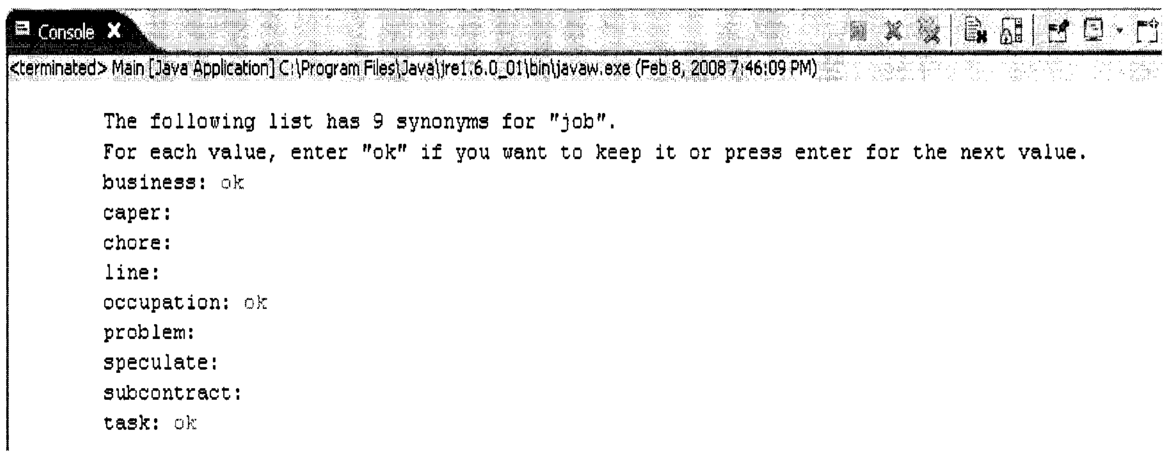
To increase the chances of finding candidate answers among the data collection, we include synonyms of the keywords in addition to the keywords themselves. WordNet [13] is a lexical database for the English language developed at Princeton University and has been successfully used in other CINDI related projects [6] so its selection was pretty obvious.

WordNet was used in concordance with Lucene. Lucene enables us to create an index composed strictly of synonyms defined by WordNet that can be queried like a regular index so we can actually get a list of synonyms of a specific word. This strategy is highlighted in pages 292-296 of *Lucene in Action* [17].

When no template is matched and after defining the keywords using the Link Parser, we query the WordNet index to obtain the synonyms of each keyword, except for Capitalized

Entities. Since some of those synonyms are irrelevant or out of context, the user has the choice of discarding them and selecting only the appropriate ones.

Figure 3.5 illustrates the use of WordNet within CINDI_QA.



```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 8, 2008 7:46:09 PM)

        The following list has 9 synonyms for "job".
        For each value, enter "ok" if you want to keep it or press enter for the next value.
        business: ok
        caper:
        chore:
        line:
        occupation: ok
        problem:
        speculate:
        subcontract:
        task: ok
```

Figure 3.5: WordNet in CINDI_QA

3.5 Document Searching and Indexing Module

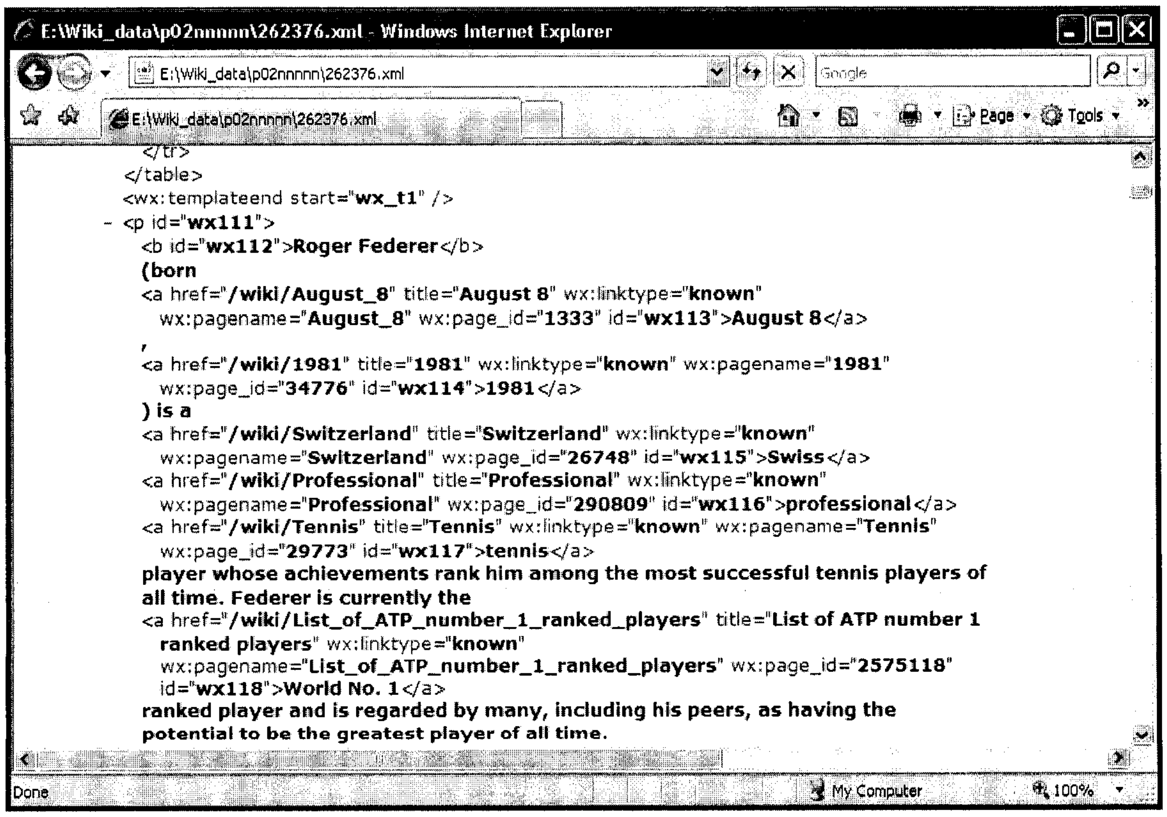
We need a tool that can not only index all the documents in our data collection but also search the created index with some level of intelligence such as ranking results and highlighting query terms. A perfect match for this requirement is the Apache Software Foundation's Lucene [18], a high-performance, full-featured text search engine library written entirely in Java.

CINDI_QA makes extensive use of Lucene. As mentioned in the previous section, it is used in concordance with WordNet to get synonyms of keywords. Lucene also creates the index and ranks the results found. Let us first have a look at our data collection before mentioning two features of Lucene that are of great importance to our system in sections 3.5.2 and 3.5.3.

3.5.1 CINDI_QA Data collection

Since CINDI_QA participated in the 2007 edition of the Multiple Language Question Answering Cross Language Evaluation Forum (QA@CLEF) [11], we decided to keep a part of their data for the purpose of our project. QA@CLEF mainly relied upon a static version of Articles from Wikipedia [19] as a reference. This constitutes a large downloaded collection of XML files, making up Wikipedia's entire English database dating from November 2006. Originally, this was done so that all participants would have access to the same source of information hence simplifying the judgment of answers.

Figure 3.6 shows a snippet of a random file selected from the data collection, in this case the page about Roger Federer.


E:\Wiki_data\p02nnnnn\262376.xml - Windows Internet Explorer
E:\Wiki_data\p02nnnnn\262376.xml
Google
E:\Wiki_data\p02nnnnn\262376.xml
Page
Tools
Done
My Computer
100%

```
      </tr>
    </table>
    <wx:templateend start="wx_t1" />
  - <p id="wx111">
      <b id="wx112">Roger Federer</b>
      (born
      <a href="/wiki/August_8" title="August 8" wx:linktype="known"
        wx:pagename="August_8" wx:page_id="1333" id="wx113">August 8</a>
      ,
      <a href="/wiki/1981" title="1981" wx:linktype="known" wx:pagename="1981"
        wx:page_id="34776" id="wx114">1981</a>
      ) is a
      <a href="/wiki/Switzerland" title="Switzerland" wx:linktype="known"
        wx:pagename="Switzerland" wx:page_id="26748" id="wx115">Swiss</a>
      <a href="/wiki/Professional" title="Professional" wx:linktype="known"
        wx:pagename="Professional" wx:page_id="290809" id="wx116">professional</a>
      <a href="/wiki/Tennis" title="Tennis" wx:linktype="known" wx:pagename="Tennis"
        wx:page_id="29773" id="wx117">tennis</a>
      player whose achievements rank him among the most successful tennis players of
      all time. Federer is currently the
      <a href="/wiki/List_of_ATP_number_1_ranked_players" title="List of ATP number 1
        ranked players" wx:linktype="known"
        wx:pagename="List_of_ATP_number_1_ranked_players" wx:page_id="2575118"
        id="wx118">World No. 1</a>
      ranked player and is regarded by many, including his peers, as having the
      potential to be the greatest player of all time.
```

Figure 3.6: Static Wikipedia page of Roger Federer

3.5.2 Query Building using Proximity Search

Once we have identified the keywords and their synonyms, the building of the query takes place. The query is constructed by putting together each keyword or its synonym with the other keywords or their synonyms. The crucial point here is to add the proximity search flag to the built query so that Lucene will not look for a sentence that has our keywords adjacent to each other, but rather one where the keywords are close to each other but also spread out in a paragraph. This is done by adding the tilde character '~' and a number to the end of the query.

For example, say that the user asked how many planets the Solar System has. CINDI_QA will identify "Solar System" as a Capitalized Entity and "planets" as a noun. Simply searching for the string "Solar System planets" will not be helpful because it is unlikely that those three words will appear next to each other. So building the query "Solar System planets"~20, where Lucene searches for the three words with a maximum gap of twenty words between each one, is more realistic and yields better results.

3.5.3 Highlighting Query Terms

Once the Lucene query is built, it is searched against the index of the document collection. Lucene then returns a list of filenames ranked according to the frequency of occurrence of the words in the query. At this point, we have a few candidate files where the answer exists but we do not yet hold the actual string containing the answer. Consequently, we take advantage of the Lucene Highlighter, a wonderful tool that actually displays snippets of text from the candidate file with the query terms highlighted. This allows us not only to know which document has the answer, but also to obtain a sentence in that document that displays the actual answer. This mechanism is mentioned on page 300 of *Lucene in Action* [17]. The actual highlighting that occurs is done by surrounding each query term with <B> tags, as demonstrated by table 3.1.

Query Term	fox
Located String	The quick brown fox jumped over the lazy dog.
Highlighted Result	The quick brown **<B>fox</B>** jumped over the lazy dog.

Table 3.1: Highlighter Example

Chapter 4

Template Matching

A template is defined as a model or pattern used for making multiple copies of a single object [20]. In the field of Computer Science, a template is a high-level construct that represents several similar entities by identifying a general structure that matches those entities. In the scope of CINDI_QA, a template identifies the format of a question so that the system can produce the best answer. All the templates used by CINDI_QA have a wildcard word, represented by the uppercase letter X, Y or Z. This wildcard takes the place of a word or a bunch of words within the definition of a template.

In most cases, the wildcard is a Capitalized Entity (CE) that consists of a continuous sequence of words whose first letter is capitalized. So a CE could be one word or a composition of words as long as adjacent words are capitalized. Table 4.1 highlights the status of a few expressions as Capitalized Entities.

Expression	**CE Status**
Chedid	True
Chedid Haddad	True
Chedid amine Haddad	False
Ludwig Van Beethoven	True
hsbc	False
NASA	True

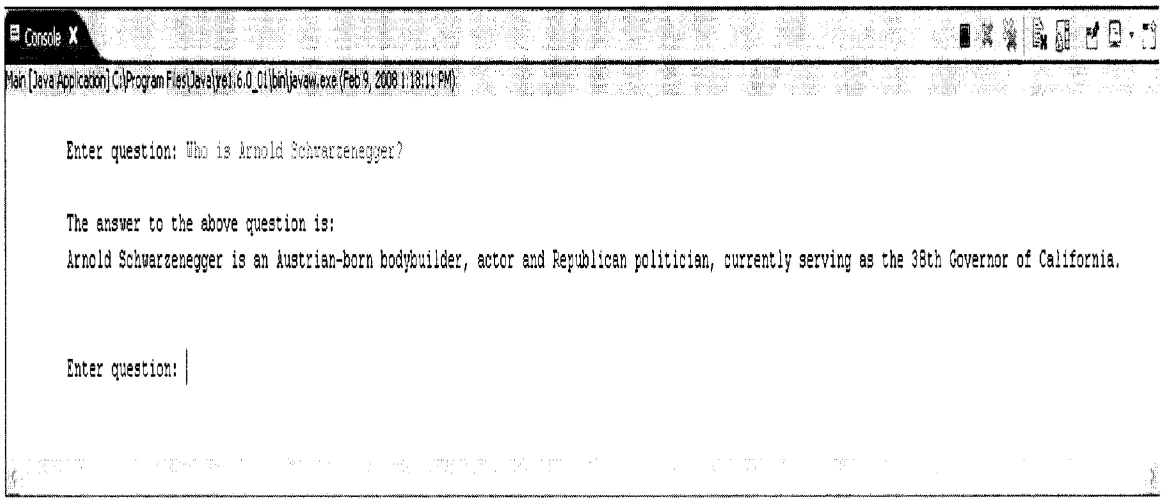
Table 4.1: Capitalized Entity examples

4.1 The Living Person Definition Template: *Who is X?*

In order for this template to be matched, the English question must meet the following criteria: it must start with the word "Who" (capitalization of the first letter is optional but expected since this is the first word of the sentence), then have the verb "is" and end with a CE X.

After the English question has been obtained from the Online Translation module, we check it against our set of templates. In this case, it would have matched the first template, the Living Person Definition template. So now we know that the user is asking for general information about someone who is still alive. Since we matched a template, we skip the process of parsing the question and identifying keyword synonyms. We go straight to searching our index for the page whose title is X. We might get more than one result, so we go through them until we find a sentence that actually explains who X is, using the Lucene highlighter on the string "X is". If we do find such a string, we simply return it to the user.

Figure 4.1 shows the answer returned by CINDI_QA to an English question that matches the Living Person Definition template.



```
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 9, 2008 1:18:11 PM)

Enter question: Who is Arnold Schwarzenegger?

The answer to the above question is:
Arnold Schwarzenegger is an Austrian-born bodybuilder, actor and Republican politician, currently serving as the 38th Governor of California.


Enter question:
```

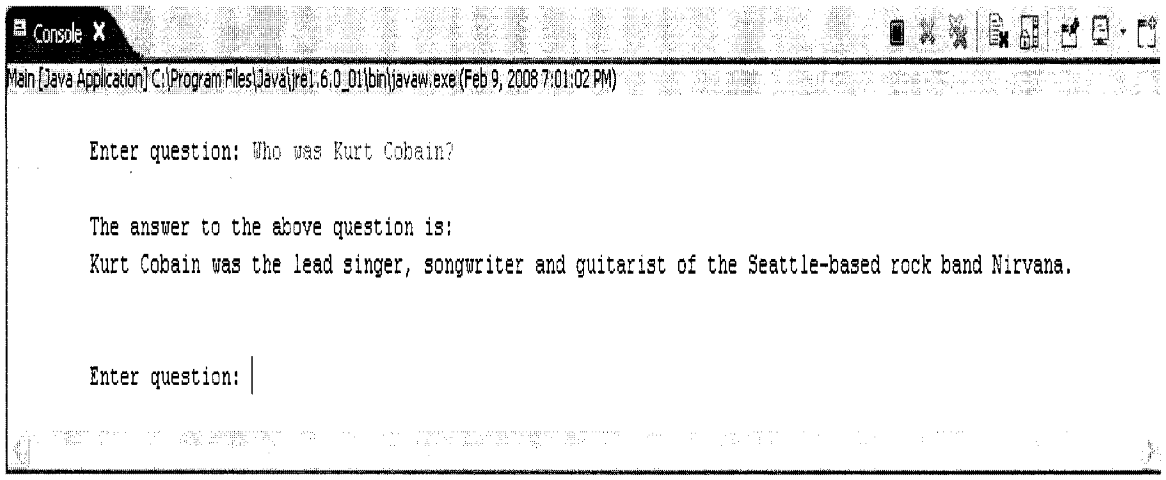
Figure 4.1: Living Person Definition template example

4.2 The Deceased Person Definition Template: *Who was X?*

This template is very similar to the previous one except that it deals with someone that has already passed away. The English question must start with the word "who", then have the verb "was" and end with a CE X.

The reason we separate this template from the first one is that Wikipedia describes alive and deceased people in different ways. Indeed, the top of the page about X doesn't start as "X is …" but rather as "X was …". Hence, after we get a page whose title is X from our Indexing module, we look for the string "X was". Again, since Wikipedia differentiates between the living status of a person when describing him/her, we define two separate templates whose process flow, although related, is different.

Figure 4.2 shows the answer returned by CINDI_QA to an English question that matches the Deceased Person Definition template.

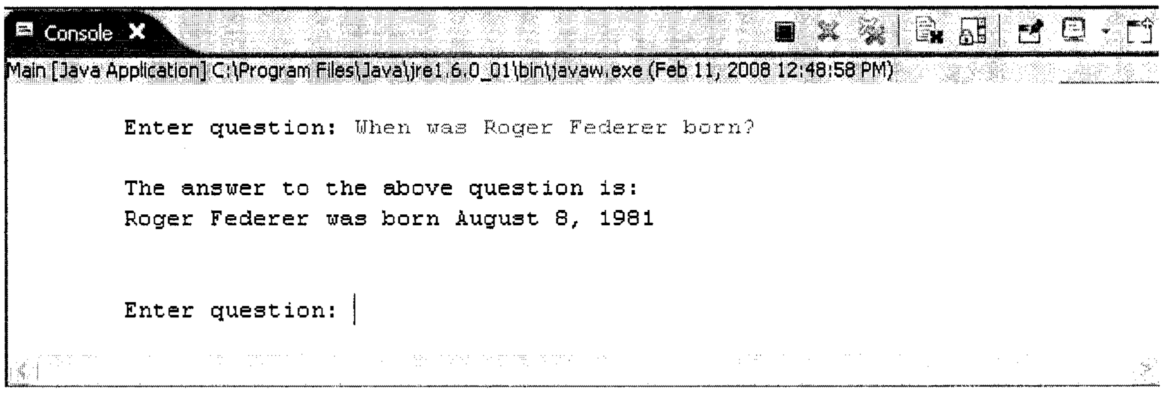

Console X
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 9, 2008 7:01:02 PM)

```
Enter question: Who was Kurt Cobain?

The answer to the above question is:
Kurt Cobain was the lead singer, songwriter and guitarist of the Seattle-based rock band Nirvana.


Enter question:
```

Figure 4.2: Deceased Person Definition template example

4.3 The Person Date of Birth Template: *When was X born?*

This template deals with people's date of birth without differentiating between alive or dead people: it works for both cases. The English question must start with the word "when", then have the verb "was" followed by a CE X, and end with the word "born". Since we matched a template, we skip the process of parsing the question and identifying keyword synonyms. We go straight to searching our index for the page whose title is X. We might get more than one result, so we go through them until we find a sentence that mentions when X was born. This is facilitated by the structure of Wikipedia's xml files. In fact, Wikipedia states the date of birth between parentheses of everyone right next to their full name at the top of the page about that person (this is observed by looking at figure 3.5 but specially when going online at the HTTP Wikipedia page for anyone). Figure 4.3 shows the answer returned by CINDI_QA to an English question that matches the Person Date of Birth template.



```
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 11, 2008 12:48:58 PM)

        Enter question: When was Roger Federer born?

        The answer to the above question is:
        Roger Federer was born August 8, 1981


        Enter question: |
```

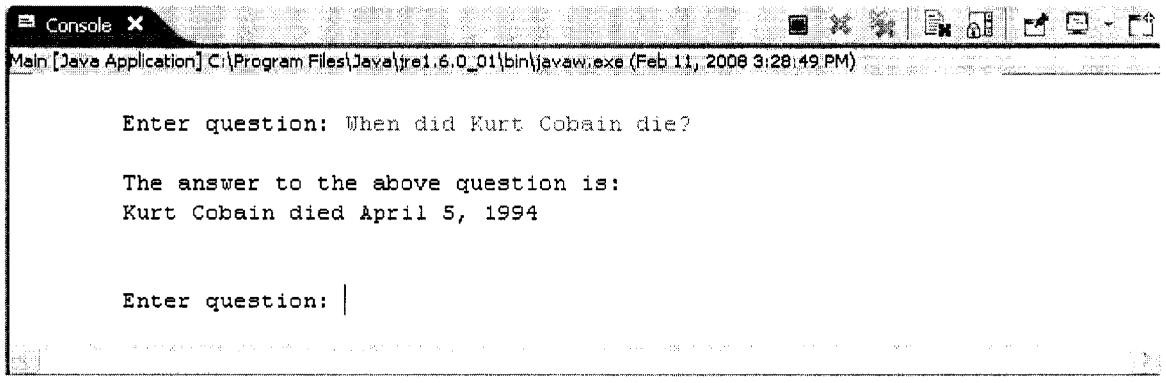
Figure 4.3: Person Date of Birth template example

4.4 The Person Date of Death Template: *When did X die?*

This template is very similar to the previous one except that it deals with someone that has already passed away. So the English question must start with the word "when", then have the verb "did" followed by a CE X, and end with the word "die".

The reason we separate this template from the first one is that Wikipedia describes alive and deceased people in different ways. Indeed, the top of the page about X doesn't start as "X is …" but rather as "X was …". But here as well, we take advantage of the fact that Wikipedia states the dates of birth and death between parentheses of every deceased person right next to their full name at the top of their page. Hence, after we get a page whose title is X from our Indexing module, we look for the sentence that mentions when X passed away.

Figure 4.4 shows the answer returned by CINDI_QA to an English question that matches the Person Date of Death template.



```
Console
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 11, 2008 3:28:49 PM)

        Enter question: When did Kurt Cobain die?

        The answer to the above question is:
        Kurt Cobain died April 5, 1994


        Enter question: |
```

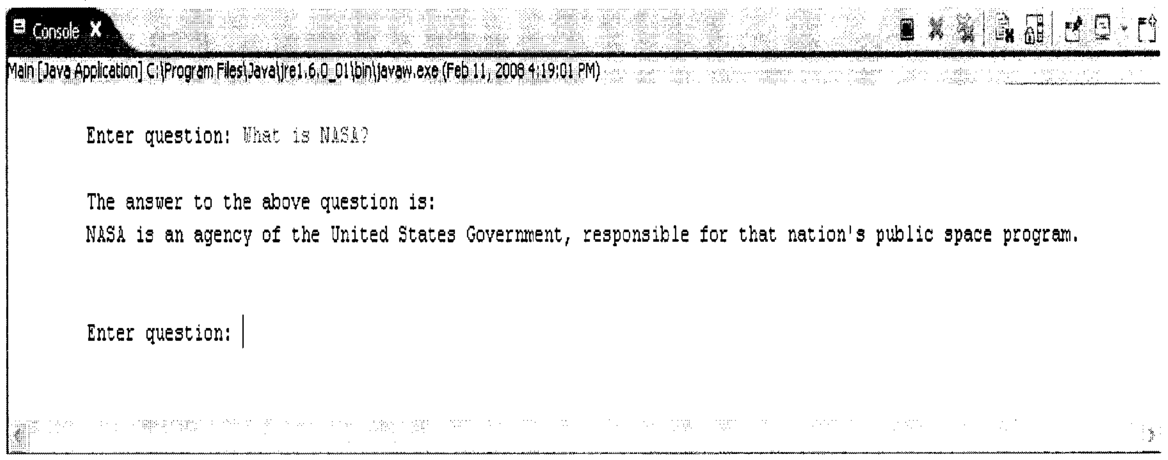
Figure 4.4: Person Date of Death template example

4.5 The Object Definition Template: *What is Z?*

This is a special template because it is the only template that doesn't inquire about people. Indeed, the CE Z can represent organizations, musical bands, international brands, company names and the like. It will try to answer any question as long as Z starts with an uppercase letter, so it cannot answer questions about objects like "What is a table?". The English question must start with the word "what", then have the verb "is" followed by a CE Z.

Since we matched a template, we skip the process of parsing the question and identifying keyword synonyms. We go straight to searching our index for the page whose title is Z. We might get more than one result, so we go through them until we find a sentence that actually explains what Z is, using the Lucene highlighter on the string "Z is". If we do find such a string, we simply return it to the user.

Figure 4.5 shows the answer returned by CINDI_QA to an English question that matches the Object Definition template.



```
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 11, 2008 4:19:01 PM)

        Enter question: What is NASA?

        The answer to the above question is:
        NASA is an agency of the United States Government, responsible for that nation's public space program.


        Enter question:
```

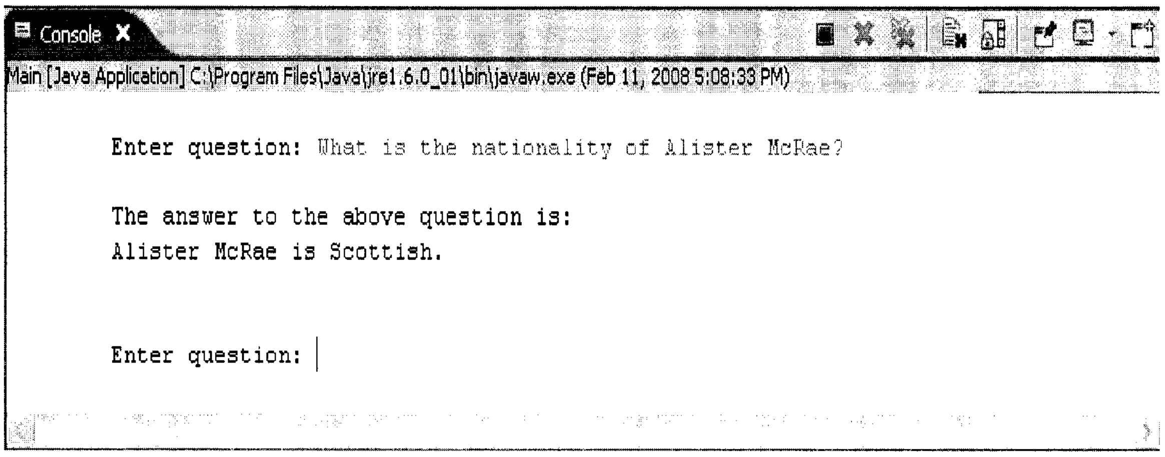
Figure 4.5: Object Definition template example

4.6 The Person Info Template: *What is the Y of X?*

This is the most versatile template in CINDI_QA because it can match several types of questions. Indeed, in addition to X representing a CE, the wildcard Y stands for a noun. The advantage of this method is that the implementation of this template can be expanded to handle a large number of nouns for Y. To this day however, the template works if Y belongs to the lexical reference of two words: "nationality" and "job".

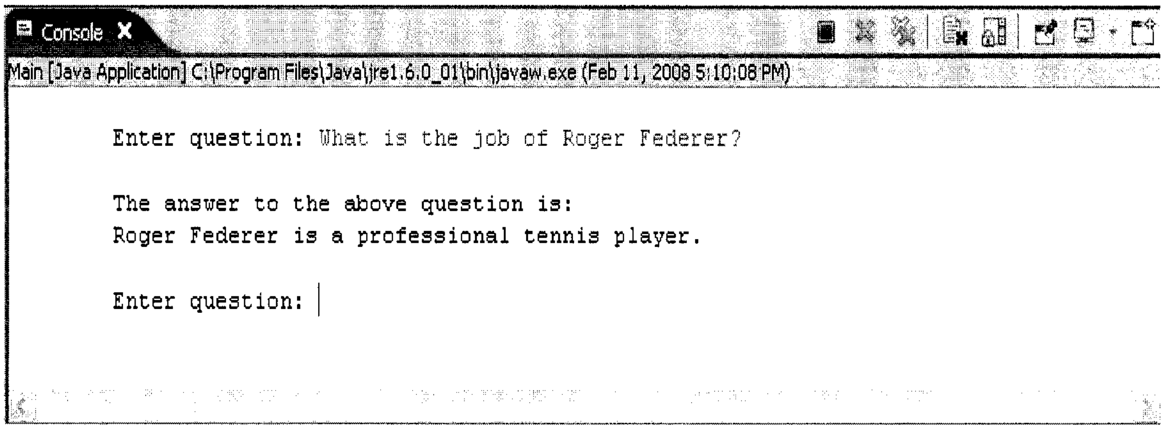
We mentioned the lexical reference because this template can not only have a large range, but it is also flexible throughout said range. In fact, for this template to be matched, Y isn't restricted to only being one of "nationality" or "job"; as long as Y is a synonym of one of those words - Y could be "business", "trade" or "occupation" - this template will be chosen. The difference in the process flow of this template compared to previous ones resides in the fact that we actually use WordNet to get the synonyms of Y. So this is the only template where we use both Lucene and WordNet, the rest of the matching is basically the same as before.

Until now, two sub-templates are defined, one where Y concerns the "nationality" and one where Y concerns the "job". Figures 4.6 and 4.7 illustrate both cases.



```
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 11, 2008 5:08:33 PM)

      Enter question: What is the nationality of Alister McRae?

      The answer to the above question is:
      Alister McRae is Scottish.


      Enter question:
```

Figure 4.6: Person Info template example with Y = "nationality"



```
Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Feb 11, 2008 5:10:08 PM)

      Enter question: What is the job of Roger Federer?

      The answer to the above question is:
      Roger Federer is a professional tennis player.

      Enter question:
```

Figure 4.7: Person Info template example with Y = "job"

Chapter 5

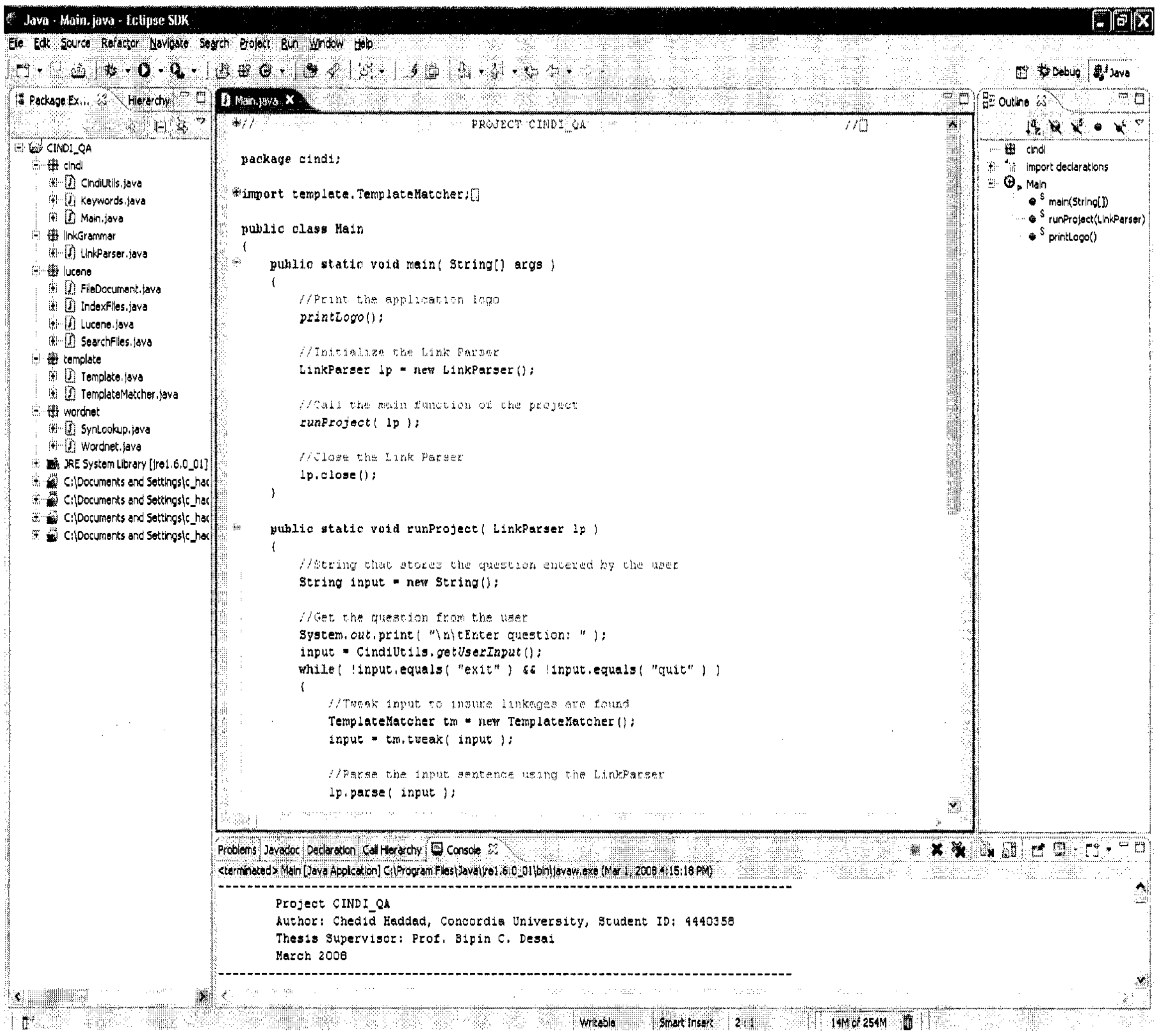
CINDI_QA Implementation

CINDI_QA has been completely developed in the JAVA Object-Oriented (OO) programming language. In order to do so, the Java Development Kit (JDK) and Java Runtime Environment (JRE) were both downloaded from the Sun Microsystems website [21]. The Operating System (OS) used was Windows XP Professional Service Pack 2. Table 5.1 provides a more detailed list of the implementation parameters for the project.

Lab Room	EV 9.105
Machine Name	cindi6
Processor Type	AMD Athlon
Processor Speed	950 MHz
Operating System	Windows XP Pro SP2
Random Access Memory	1 GB
Development Language	Java
JDK Version Number	5.0
JRE Version Number	6.0
Development Platform	Eclipse Europa 3.3.2
Number of Classes	12

Table 5.1: CINDI_QA Implementation Parameters

Eclipse is an open-source development platform deployed by the Eclipse Foundation [22]. It is specially tailored for the Java programming language. Eclipse was chosen because of a neat interface, a simple debugging mode and an on-the-go compiling system that displays warnings and errors as soon as a file is saved, without troubling the developer of explicitly performing a build. Figure 5.1 shows the Eclipse interface with the CINDI_QA project open.



```
//                                    PROJECT CINDI_QA                    //

package cindi;

import template.TemplateMatcher;

public class Main
{
    public static void main( String[] args )
    {
        //Print the application logo
        printLogo();

        //Initialize the Link Parser
        LinkParser lp = new LinkParser();

        //Call the main function of the project
        runProject( lp );

        //Close the Link Parser
        lp.close();
    }

    public static void runProject( LinkParser lp )
    {
        //String that stores the question entered by the user
        String input = new String();

        //Get the question from the user
        System.out.print( "\n\tEnter question: " );
        input = CindiUtils.getUserInput();
        while( !input.equals( "exit" ) && !input.equals( "quit" ) )
        {
            //Tweak input to insure linkages are found
            TemplateMatcher tm = new TemplateMatcher();
            input = tm.tweak( input );

            //Parse the input sentence using the LinkParser
            lp.parse( input );
```

```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.6.0_01\bin\javaw.exe (Mar 1, 2008 4:15:18 PM)
--------------------------------------------------------------------------------
        Project CINDI_QA
        Author: Chedid Haddad, Concordia University, Student ID: 4440358
        Thesis Supervisor: Prof. Bipin C. Desai
        March 2008
--------------------------------------------------------------------------------
```

Figure 5.1: Eclipse interface

The following two sections explain how the tools used by CINCI_QA were integrated in the project.

5.1 Java Native Code Link Grammar Interface

The Link Parser is a syntactic parser of English [12] and is used by CINDI_QA to identify the keywords from the input question, such as nouns, verbs and adjectives. At first, we were facing an inconsistency problem because the Link Parser is entirely written in generic C code whereas CINDI_QA was intended to be implemented in Java. Fortunately enough, we came across the Java Native Code Link Grammar Interface (JNCLGI) developed by Chris Jordan [23], which is a Java interface to the Link Parser. The great advantage of the JNCLGI is that it is released as a Java Archive (JAR) file which essentially is a collection of several Java classes packaged into a single file. By adding the 'linkGrammar.jar' file to the include definition of our project, we are able to benefit from the Link Parser while working in a Java environment.

5.2 WordNet and Lucene Cooperation

Lucene is an open-source IR library implemented in Java and distributed by the Apache Software Foundation [18]. It is used by CINDI_QA to index and search the data collection. Lucene is incorporated into CINDI_QA by importing JAR files the same way it is done with the Link Parser.

First, the file 'lucene-core-2.1.0.jar' is added to the project. This archive contains the main interface to the Lucene engine, mainly the classes to index and search the data collection.

Second, another file, 'lucene-highlighter-2.1.0.jar', is responsible for providing the necessary mechanism to perform highlighting of chunks of words, as discussed in section 3.5.3.

Third, the file 'lucene-wordnet-2.1.0.jar' includes the appropriate classes to query WordNet for lexical references of some keywords. As mentioned in section 3.4, Lucene and WordNet cooperate so that Lucene creates an index composed of synonyms defined by WordNet. The fact that WordNet is distributed as another Lucene JAR file eliminates the burden of trying to incorporate each tool by itself.

JAR files are an easy way of adding external packages of Java classes into a separate project, thus underlining the basic OO principle of code reuse.

The major Java classes of CINDI_QA are organized in a class diagram shown in figure 5.2. Because the majority of the methods are static, the relationships between classes consist of dependency relations instead of the traditional associations.

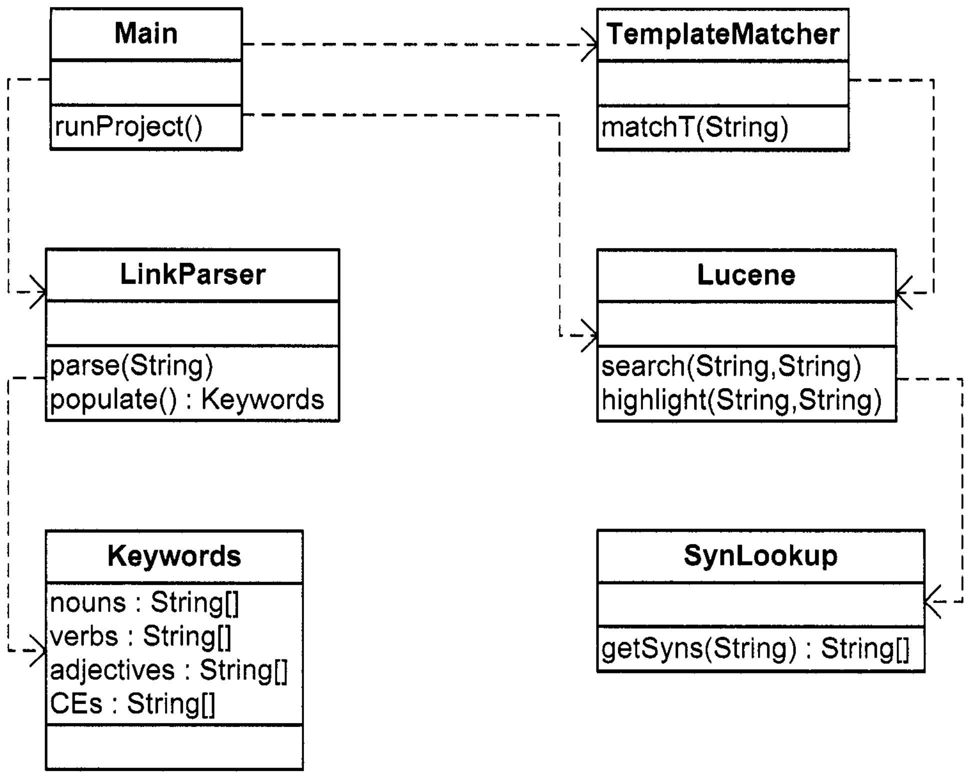


Figure 5.2: CINDI_QA Class Diagram

Chapter 6

Experimental Results

This chapter aims at providing an in-depth look at the behavior of CINDI_QA under different circumstances. Sample answers generated by the system are provided as well as explanations on why CINDI_QA fails to correctly answer some questions. The case when no template is matched is also explored. After that, the participation of the CINDI_QA group at the 2007 edition of QA@CLEF is examined.

First, let us revisit the six templates that were explained in detail in chapter 4. Because of their long title, an abbreviation is assigned to each template as listed in table 6.1.

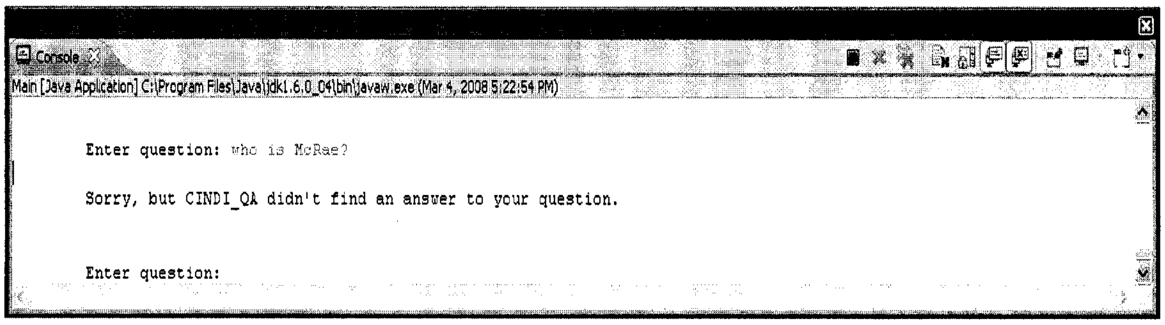
Abbreviation	Title	Matching Pattern
T1	Living Person Definition Template	Who is X?
T2	Deceased Person Definition Template	Who was X?
T3	Person Date of Birth Template	When was X born?
T4	Person Date of Death Template	When did X die?
T5	Object Definition Template	What is Z?
T6	Person Info Template	What is the Y of X?

Table 6.1: CINDI_QA Templates

Note that X stands for a Capitalized Entity referencing a person, Z from T5 is a CE referencing an object or organization and Y from T6 matches a noun.

6.1 Examples

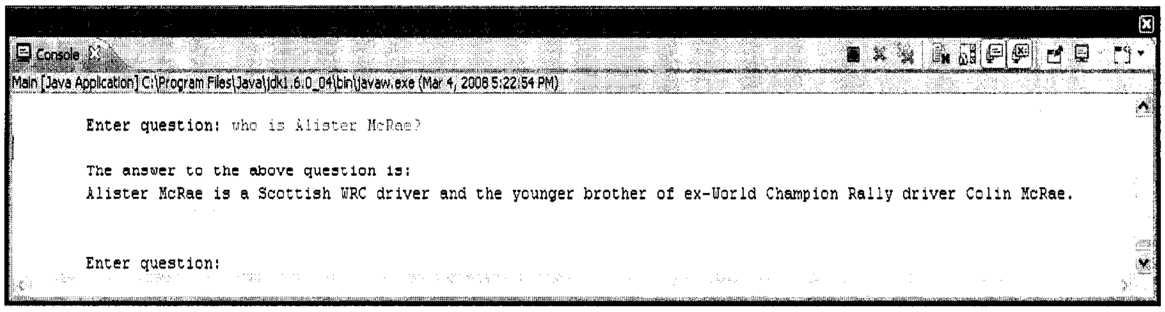
Figure 4.1 showed a working example of T1 where X = “Arnold Schwarzenegger”. Figure 6.1 displays what happens when only the family name of a person is given for X.


Console
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 5:22:54 PM)

```
Enter question: who is McRae?

Sorry, but CINDI_QA didn't find an answer to your question.


Enter question:
```

Figure 6.1: T1 Example with X = “McRae”

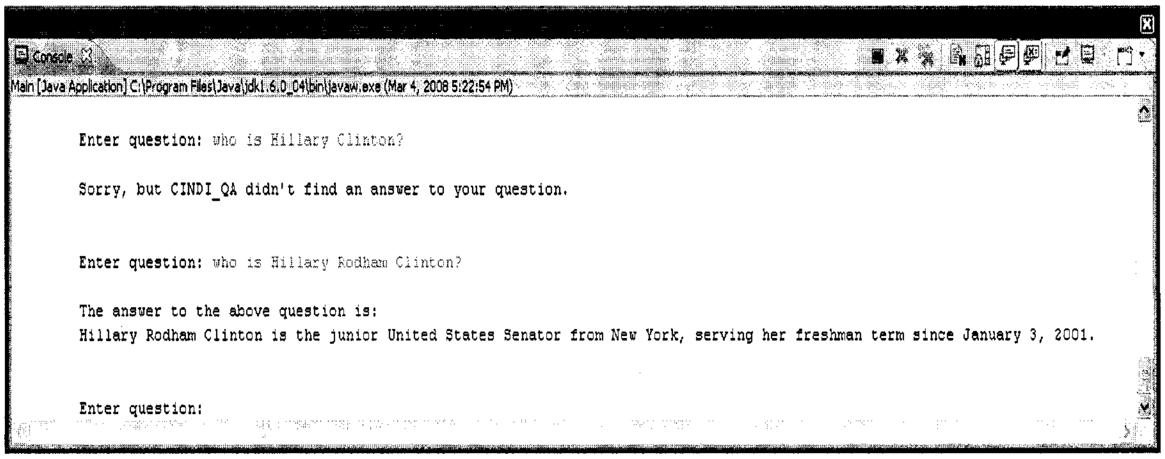
CINDI_QA doesn’t find an answer because there is no file in the data collection whose title simply is “McRae”. However, giving the full name “Alister McRae” for X generates the correct answer, as evidenced by figure 6.2.


Console
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 5:22:54 PM)

```
Enter question: who is Alister McRae?

The answer to the above question is:
Alister McRae is a Scottish WRC driver and the younger brother of ex-World Champion Rally driver Colin McRae.


Enter question:
```

Figure 6.2: T1 Example with X = “Alister McRae”

Another crucial factor in the ability of CINDI_QA to find answers is the spelling of the CEs. Every CE must be correctly spelled and the full name of a person should be provided in the question. For instance, asking a T1 question with X = "Hillary Clinton" wouldn't be sufficient. X should be "Hillary Rodham Clinton" for CINDI_QA to find an answer as demonstrated by figure 6.3.

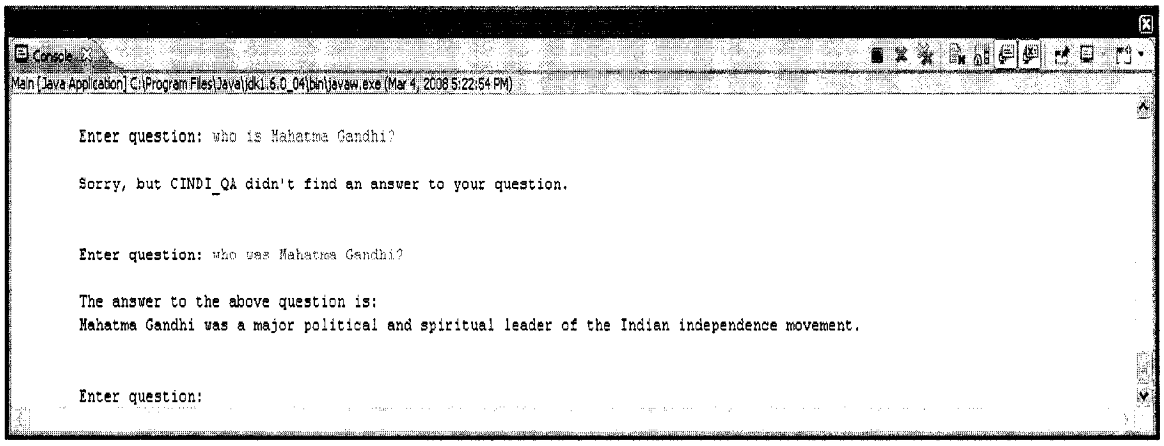

Console
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 5:22:54 PM)

```
Enter question: who is Hillary Clinton?

Sorry, but CINDI_QA didn't find an answer to your question.


Enter question: who is Hillary Rodham Clinton?

The answer to the above question is:
Hillary Rodham Clinton is the junior United States Senator from New York, serving her freshman term since January 3, 2001.


Enter question:
```

Figure 6.3: T1 Examples with X = "Hillary Clinton" and X = "Hillary Rodham Clinton"

Figure 6.3 also emphasizes the fact that one CE doesn't necessarily need to be composed of only one or two distinct capitalized words. As mentioned in the beginning of chapter 4, a CE can contain three capitalized words, as long as they are adjacent to each other like in this case where X = "Hillary Rodham Clinton".

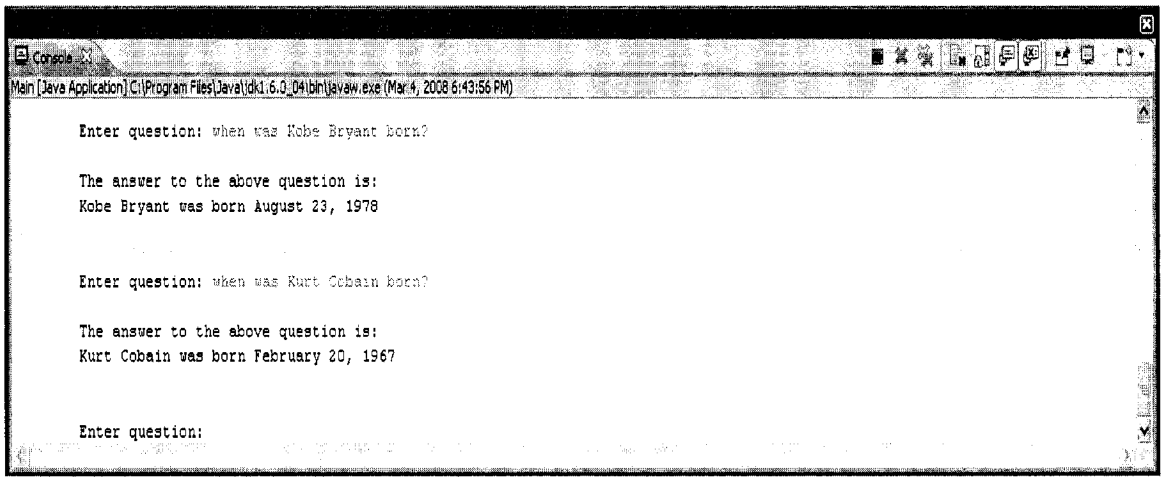
T1 and T2 were separated so that CINDI_QA can expect different questions when querying about living and deceased people. Indeed, asking about Gandhi in the present tense doesn't make sense, but using the past tense to inquire about him should trigger an answer as shown in figure 6.4.



```
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 5:22:54 PM)

Enter question: who is Mahatma Gandhi?

Sorry, but CINDI_QA didn't find an answer to your question.


Enter question: who was Mahatma Gandhi?

The answer to the above question is:
Mahatma Gandhi was a major political and spiritual leader of the Indian independence movement.


Enter question:
```

Figure 6.4: T1 and T2 Examples with X = "Mahatma Gandhi"

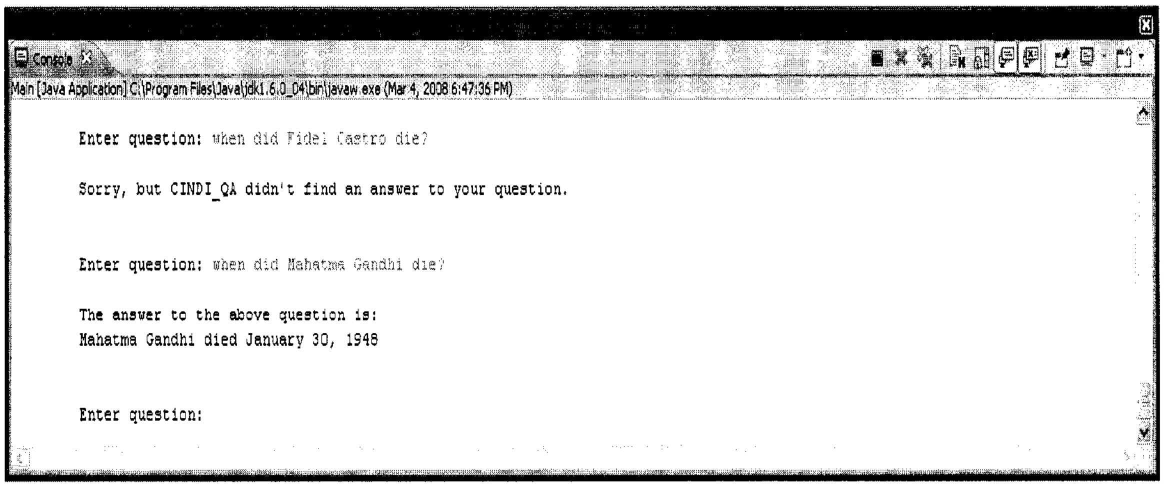
Template T3, which asks for the date of birth of a person, works both for living and deceased people. Figure 6.5 illustrates this by asking CINDI_QA for the birthdays of Kobe Bryant (alive) and Kurt Cobain (deceased).



```
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 6:43:56 PM)

Enter question: when was Kobe Bryant born?

The answer to the above question is:
Kobe Bryant was born August 23, 1978


Enter question: when was Kurt Cobain born?

The answer to the above question is:
Kurt Cobain was born February 20, 1967


Enter question:
```

Figure 6.5: T3 Examples with X = "Kobe Bryant" and X = "Kurt Cobain"

Template T4, which asks for the day on which a person passed away, is supposed to generate answers only for deceased people. Therefore, when queried about the date of death of someone who is still alive, CINDI_QA doesn't find any answer, as shown in figure 6.6.

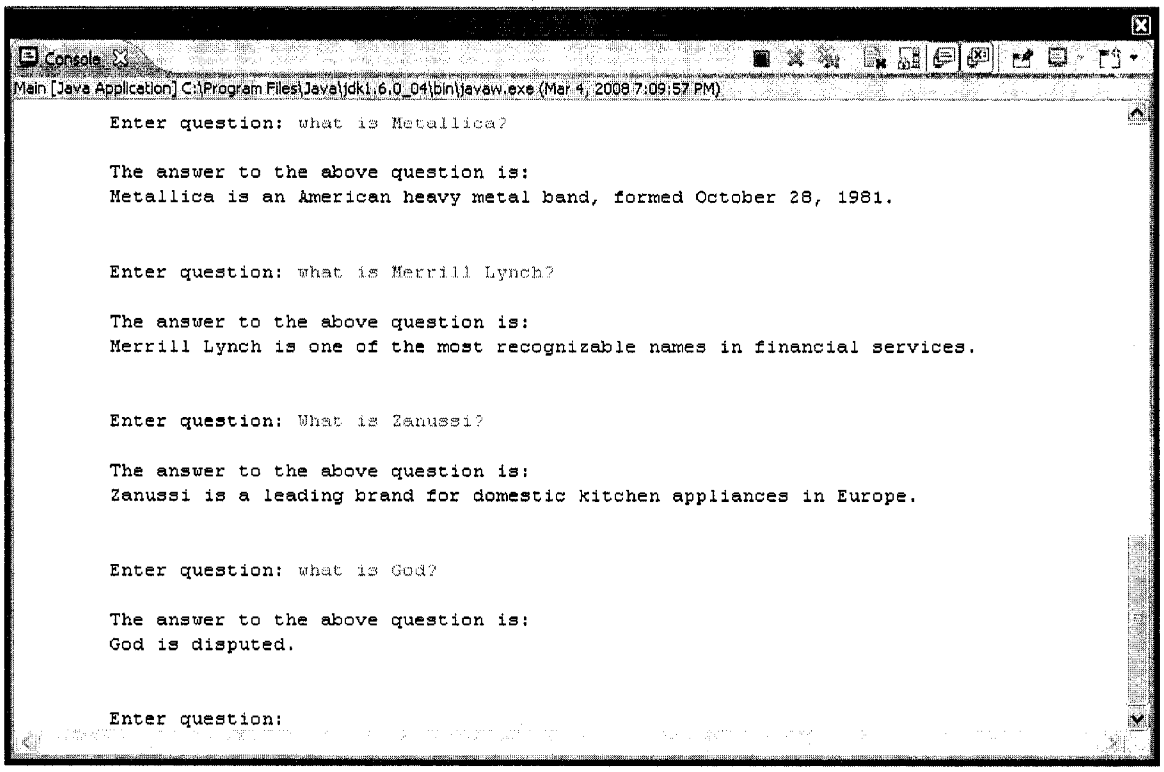


Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 6:47:36 PM)

```
Enter question: when did Fidel Castro die?

Sorry, but CINDI_QA didn't find an answer to your question.


Enter question: when did Mahatma Gandhi die?

The answer to the above question is:
Mahatma Gandhi died January 30, 1948


Enter question:
```

Figure 6.6: T4 Examples with X = "Fidel Castro" and X = "Mahatma Gandhi"

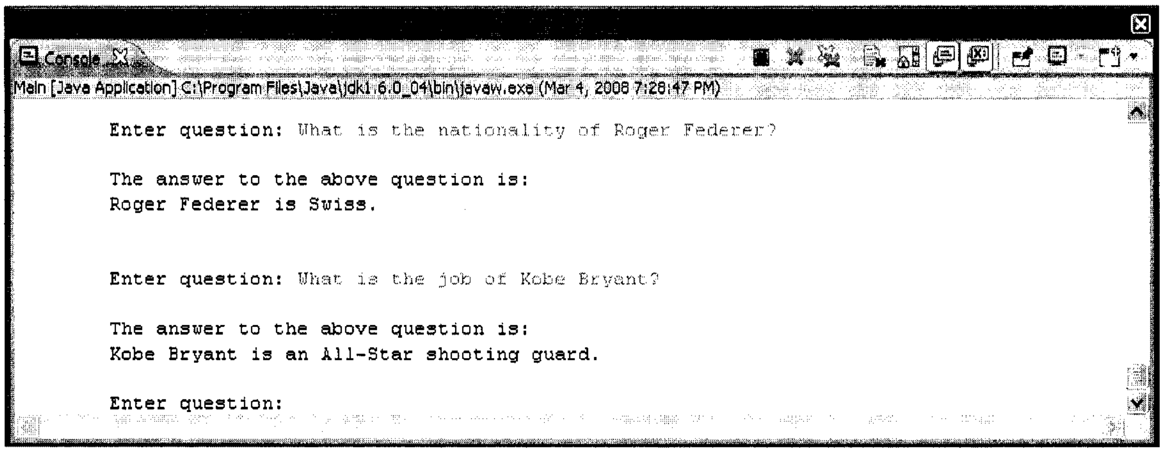
Template T5 handles definitions of objects Z. Z can be anything from well-known brands to governmental agencies to music groups. Figure 6.7 lists a few examples. Notice that the last question in figure 6.7 asks about God. Although the answer generated by CINDI_QA seems appropriate to some extent (the existence of God is indeed disputed), the answer should be considered false because the proper Wikipedia definition of the term "God" isn't the one displayed by CINDI_QA. Instead, CINDI_QA returned the first sentence it wrongly assumed to be correct from the file whose title is "God", as the process is mentioned in section 4.5. This example is still worth mentioning because of the humorous answer.


Console
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 7:09:57 PM)

```
Enter question: what is Metallica?

The answer to the above question is:
Metallica is an American heavy metal band, formed October 28, 1981.


Enter question: what is Merrill Lynch?

The answer to the above question is:
Merrill Lynch is one of the most recognizable names in financial services.


Enter question: What is Zanussi?

The answer to the above question is:
Zanussi is a leading brand for domestic kitchen appliances in Europe.


Enter question: what is God?

The answer to the above question is:
God is disputed.


Enter question:
```

Figure 6.7: T5 Examples

Template T6 is the most versatile template of CINDI_QA. It defines two distinct tokens that can match different values. On the one hand, X is the same X as in T1, T2, T3 and T4 so it stands for a CE of a person. On the other hand, Y takes the place of a noun that provides more precision in the question. As mentioned in section 4.6, until now T6 only works when Y is one of two nouns, “nationality” or “job”. Both cases are illustrated in figure 6.8.

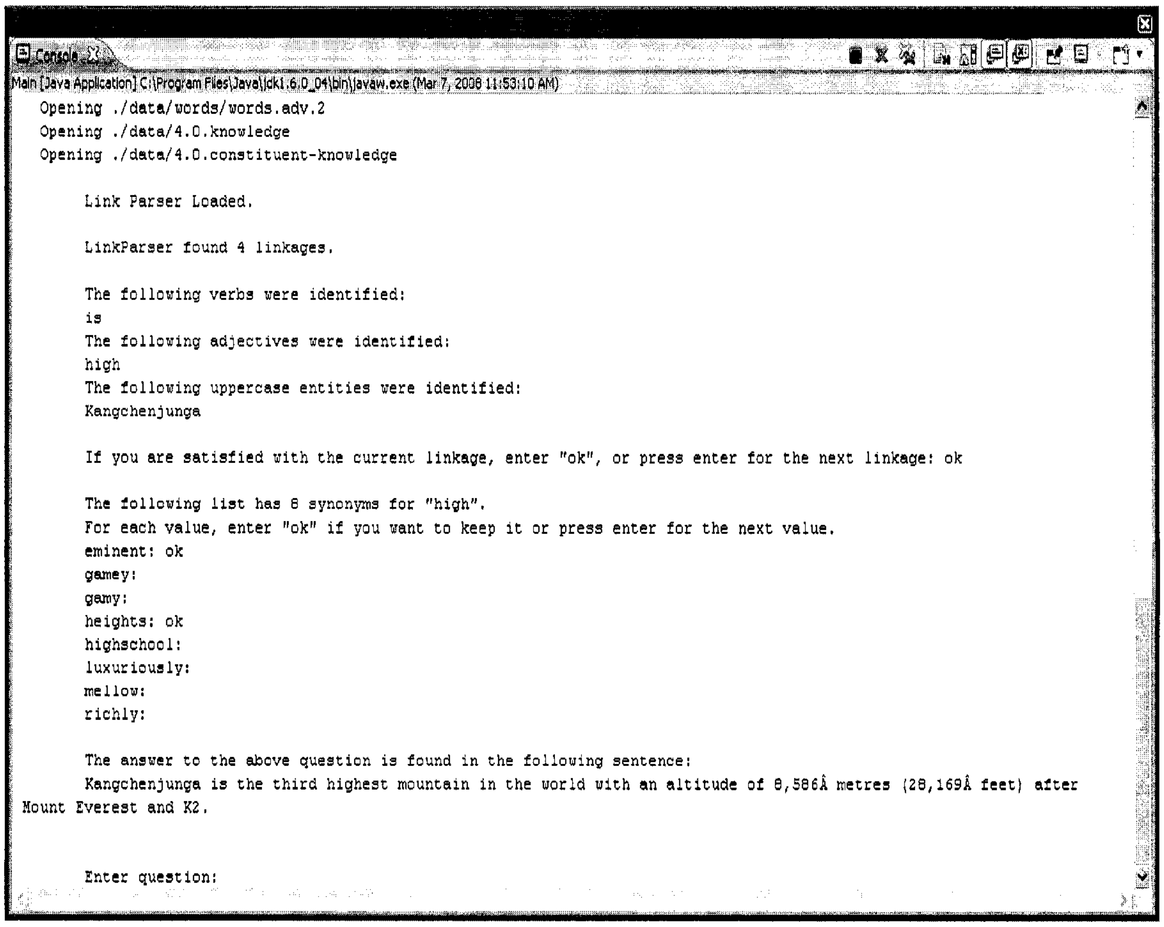

Console

Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 4, 2008 7:28:47 PM)

```
        Enter question: What is the nationality of Roger Federer?

        The answer to the above question is:
        Roger Federer is Swiss.


        Enter question: What is the job of Kobe Bryant?

        The answer to the above question is:
        Kobe Bryant is an All-Star shooting guard.

        Enter question:
```

Figure 6.8: T6 Examples

When no template is matched (NO_T), CINDI_QA uses the Link Parser and WordNet in addition to Lucene to answer the question. Since there is no concrete way of specifically knowing what the user is asking about, CINDI_QA acts more like an Information Retrieval system than a Question Answering one in this case. Figure 6.9 illustrates CINDI_QA's response to the question "How high is Kangchenjunga?". The question entered by the user is not apparent in the figure. The reason for this is that loading the Link Parser generates console statements from the Application Program Interface (API) of the Link Parser itself. These statements all start with the word "Opening" and 52 of them are outputted every time the Link Parser is loaded; the last three are visible in figure 6.9.


Console
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 7, 2008 11:53:10 AM)

```
  Opening ./data/words/words.adv.2
  Opening ./data/4.0.knowledge
  Opening ./data/4.0.constituent-knowledge

        Link Parser Loaded.

        LinkParser found 4 linkages.

        The following verbs were identified:
        is
        The following adjectives were identified:
        high
        The following uppercase entities were identified:
        Kangchenjunga

        If you are satisfied with the current linkage, enter "ok", or press enter for the next linkage: ok

        The following list has 8 synonyms for "high".
        For each value, enter "ok" if you want to keep it or press enter for the next value.
        eminent: ok
        gamey:
        gamy:
        heights: ok
        highschool:
        luxuriously:
        mellow:
        richly:

        The answer to the above question is found in the following sentence:
        Kangchenjunga is the third highest mountain in the world with an altitude of 8,586Â metres (28,169Â feet) after
Mount Everest and K2.


        Enter question:
```

Figure 6.9: NO_T example 1

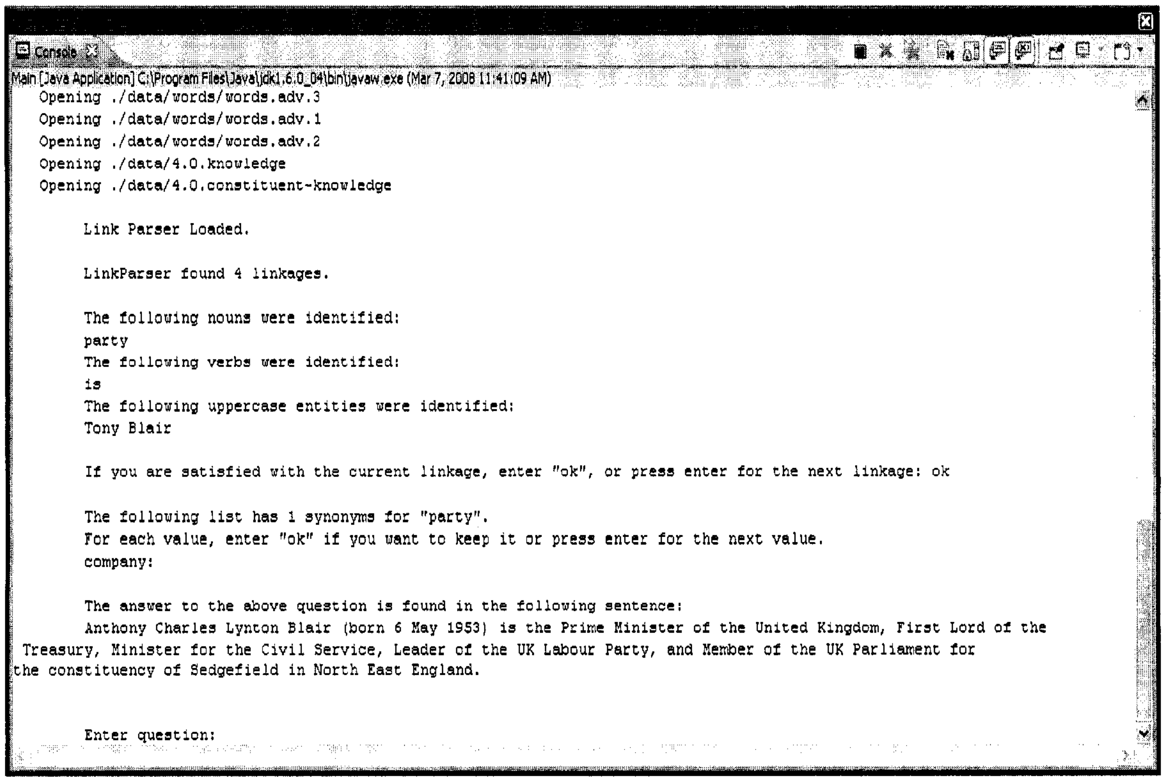
Figure 6.9 shows how the Link Parser and WordNet come into play during execution of CINDI_QA. First, CINDI_QA informs the user of the number of linkages found to make sure the right one is selected. The user enters "ok" to confirm which linkage to use depending on the keywords identified. Here, the sentence "How high is Kangchenjunga?" was parsed into the verb "is", the adjective "high" and the CE "Kangchenjunga" which sounds about right.

Then, CINDI_QA lists the synonyms of the keywords. Synonyms of common verbs like "is" and CEs are excluded of course. In this example, the user selected the words "eminent" and "heights" as synonyms of the keyword "high".

Finally, using all the information provided by the user, CINDI_QA performs a search and retrieval of the best piece of text that contains the answer to display it on screen.

CINDI_QA can have difficulty understanding the question entered by the user. For example, no answer is found for the question “What party does Tony Blair belong to?” because the Link Parser didn’t find any linkages. This is attributed to the limitations of the JNCLGI which CINDI_QA employs as its semantic parser. So CINDI_QA doesn’t go any further if the Link Parser cannot generate linkages for a given input.

However, asking the same question by formulating it differently triggers CINDI_QA to produce an answer, as shown in figure 6.10, with the question being “What party is Tony Blair?”.



```
Main [Java Application] C:\Program Files\Java\jdk1.6.0_04\bin\javaw.exe (Mar 7, 2008 11:41:09 AM)
   Opening ./data/words/words.adv.3
   Opening ./data/words/words.adv.1
   Opening ./data/words/words.adv.2
   Opening ./data/4.0.knowledge
   Opening ./data/4.0.constituent-knowledge

        Link Parser Loaded.

        LinkParser found 4 linkages.

        The following nouns were identified:
        party
        The following verbs were identified:
        is
        The following uppercase entities were identified:
        Tony Blair

        If you are satisfied with the current linkage, enter "ok", or press enter for the next linkage: ok

        The following list has 1 synonyms for "party".
        For each value, enter "ok" if you want to keep it or press enter for the next value.
        company:

        The answer to the above question is found in the following sentence:
        Anthony Charles Lynton Blair (born 6 May 1953) is the Prime Minister of the United Kingdom, First Lord of the
 Treasury, Minister for the Civil Service, Leader of the UK Labour Party, and Member of the UK Parliament for
the constituency of Sedgefield in North East England.


        Enter question:
```

Figure 6.10: NO_T example 2

Here again, the Link Parser and WordNet participate in the answer generation of CINDI_QA. Four linkages are found with the first one being selected. The sentence "What party is Tony Blair?" is parsed into the verb "is", the noun "party" and the CE "Tony Blair".

Then, CINDI_QA lists the synonyms of the noun "party". The user chose to discard the only synonym for the word "party" identified by WordNet, "company".

Using all the previous information, CINDI_QA locates the snippet of text that contains the answer and displays it.

6.2 CINDI_QA at QA@CLEF 2007

The CINDI group participated in the 2007 edition of the Multilingual Question Answering track at the Cross Language Evaluation Forum (QA@CLEF) that was held September 19 to 21 in Budapest, Hungary. CLEF defines target language as the language in which the data collection is written and source language as the language in which the question is asked. CINDI competed in the FR to EN track, where French was selected as the source language and English as the target language.

CINDI was one of only three groups based on the American continent whereas the majority of participants came from Europe [11], as shown in table 6.2.

	America	Europe	Asia	Australia	Total
CLEF 2003	3	5	-	-	**8**
CLEF 2004	1	17	-	-	**18**
CLEF 2005	1	22	1	-	**24**
CLEF 2006	4	24	2	-	**30**
CLEF 2007	3	17	1	1	**22**

Table 6.2: QA@CLEF Participation

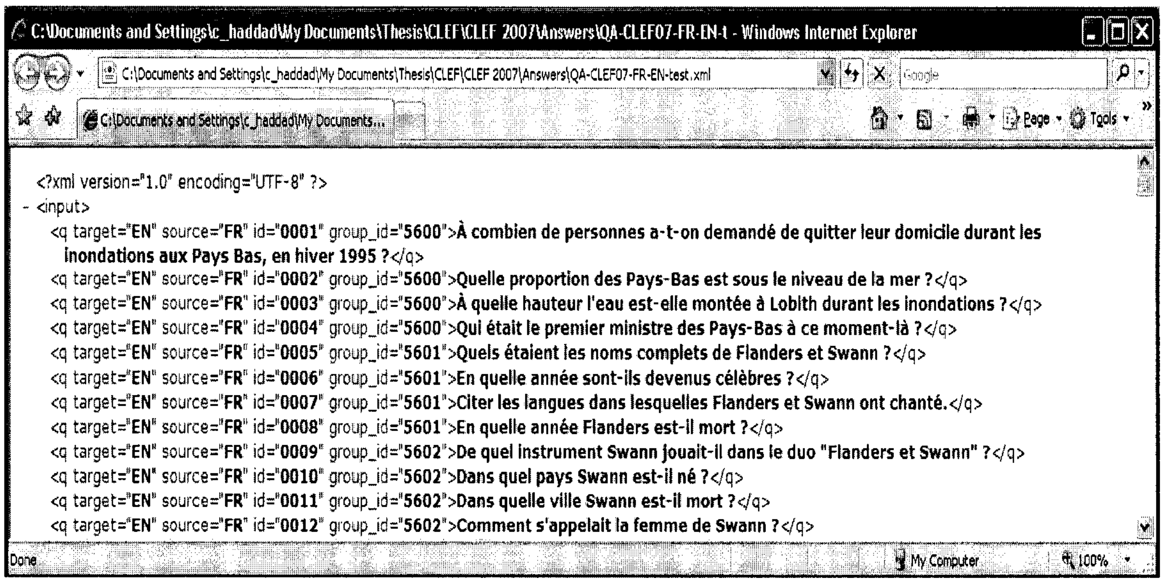
The data collection considered for EN as target consisted of news articles and Wikipedia pages. The news articles were taken from the 1994 Los Angeles Times – 113,005 documents of size 425 MB – and the 1995 Herald Tribune – 56,472 documents of size 154 MB. The Wikipedia pages were borrowed from a static version of Wikipedia dating back to the end of November 2006, which represent the same data collection generally used by CINDI_QA. This temporal restriction was made so that all participants have access to the same data.

QA@CLEF identifies three different categories of questions that are mutually exclusive and exhaustive, i.e., every question is either one of those categories and cannot fall into more than one category. Factoid (F) questions are fact-based, asking for the name of a person, a location, the extent of something, the day on which something happened, etc. Definition (D) questions, as their name suggests, ask about definitions of people, organizations and objects. Closed List (L) questions require one answer containing a determined number of items. Moreover, any of those questions can be temporally restricted (T) so that a temporal specification provides important information for the retrieval of the correct answer. These categories as well as their subtypes when available are listed in table 6.3 with examples.

Category	Subtype	Question	Answer
Factoid (F)	Person	Who was called the "Iron-Chancellor"?	Otto von Bismarck.
	Time	What year was Martin Luther King murdered?	1968.
	Location	Which town was Wolfgang Amadeus Mozart born in?	Salzburg.
	Organization	What party does Tony Blair belong to?	Labour Party.
	Measure	How high is Kanchenjunga?	8598m.
	Count	How many people died during the Terror of Pol Pot?	1 million.
	Object	What does magma consist of?	Molten rock.
	Other	Which treaty was signed in 1979?	Israel-Egyptian peace treaty.
Definition (D)	Person	Who is Robert Altmann?	Film maker.
	Organization	What is the Knesset?	Parliament of Israel.
	Object	What is Atlantis?	Space Shuttle.
	Other	What is Eurovision?	Song contest.
Closed List (L)		Name all the airports in London, England.	Gatwick, Stansted, Heathrow, Luton and City.
Temporally Restricted (T)		Who was the Chancellor of Germany from 1974 to 1982?	Helmut Schmidt.

Table 6.3: QA@CLEF Question Types

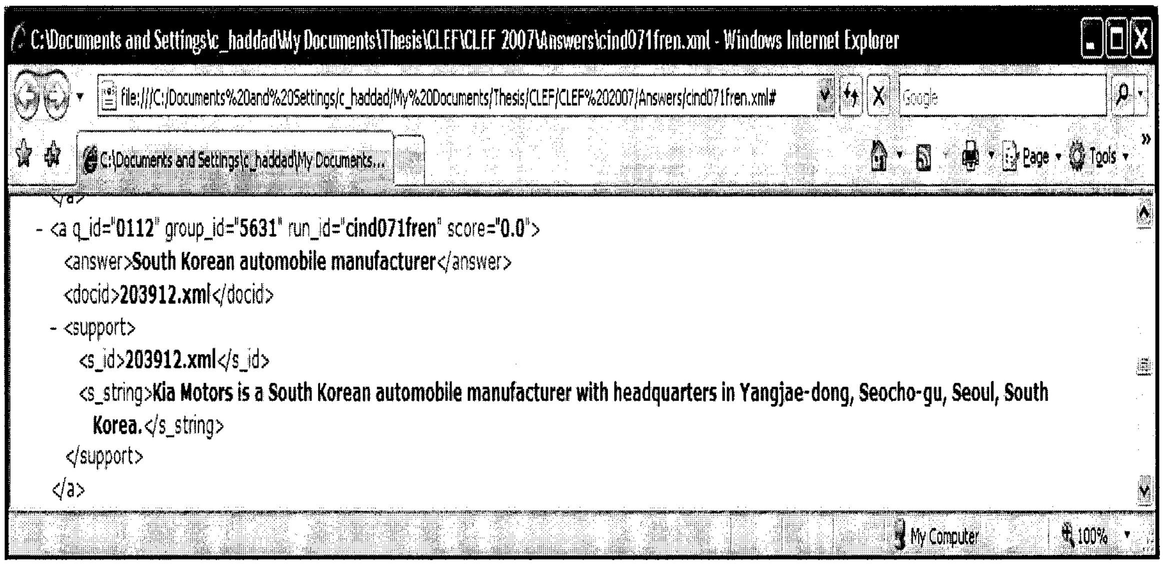
The questions were released by CLEF as XML files. Each question was enclosed in one XML tag that contained information about the target language, the source language, the question number and the question topic. One to four questions could be grouped in one topic so that they asked about the same entity but different information. This means that the topic was inferred from the first question and then no longer explicitly mentioned. Figure 6.11 displays twelve questions belonging to three topics taken from the original test set for QA@CLEF 2007.


C:\Documents and Settings\c_haddad\My Documents\Thesis\CLEF\CLEF 2007\Answers\QA-CLEF07-FR-EN-t - Windows Internet Explorer
C:\Documents and Settings\c_haddad\My Documents\Thesis\CLEF\CLEF 2007\Answers\QA-CLEF07-FR-EN-test.xml
Google
C:\Documents and Settings\c_haddad\My Documents...
Page
Tools
Done
My Computer
100%

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <input>
    <q target="EN" source="FR" id="0001" group_id="5600">À combien de personnes a-t-on demandé de quitter leur domicile durant les
      inondations aux Pays Bas, en hiver 1995 ?</q>
    <q target="EN" source="FR" id="0002" group_id="5600">Quelle proportion des Pays-Bas est sous le niveau de la mer ?</q>
    <q target="EN" source="FR" id="0003" group_id="5600">À quelle hauteur l'eau est-elle montée à Lobith durant les inondations ?</q>
    <q target="EN" source="FR" id="0004" group_id="5600">Qui était le premier ministre des Pays-Bas à ce moment-là ?</q>
    <q target="EN" source="FR" id="0005" group_id="5601">Quels étaient les noms complets de Flanders et Swann ?</q>
    <q target="EN" source="FR" id="0006" group_id="5601">En quelle année sont-ils devenus célèbres ?</q>
    <q target="EN" source="FR" id="0007" group_id="5601">Citer les langues dans lesquelles Flanders et Swann ont chanté.</q>
    <q target="EN" source="FR" id="0008" group_id="5601">En quelle année Flanders est-il mort ?</q>
    <q target="EN" source="FR" id="0009" group_id="5602">De quel instrument Swann jouait-il dans le duo "Flanders et Swann" ?</q>
    <q target="EN" source="FR" id="0010" group_id="5602">Dans quel pays Swann est-il né ?</q>
    <q target="EN" source="FR" id="0011" group_id="5602">Dans quelle ville Swann est-il mort ?</q>
    <q target="EN" source="FR" id="0012" group_id="5602">Comment s'appelait la femme de Swann ?</q>
```

Figure 6.11: QA@CLEF 2007 Test Set Excerpt

CLEF requested a specific format in which the answers should be returned, also in XML. Every answer had to be accompanied by the question number, the question topic number, the name of the run and an optional confidence score. The confidence score is a floating point value that can range from 0.0 to 1.0 where 0.0 means that the system has no evidence of the correctness of the answer, and 1.0 means that the system is absolutely confident about the correctness of the answer. CINDI_QA didn't support this feature so all values returned were 0.0. In addition to that, the answer XML tag had to reference the document filename from where the answer was extracted and attach a snippet of text from that document supporting the answer string. An example of an answer tag is given in figure 6.12 taken from the original answer file submitted at QA@CLEF 2007.


C:\Documents and Settings\c_haddad\My Documents\Thesis\CLEF\CLEF 2007\Answers\cind071fren.xml - Windows Internet Explorer
file:///C:/Documents%20and%20Settings/c_haddad/My%20Documents/Thesis/CLEF/CLEF%202007/Answers/cind071fren.xml#
My Computer
100%

```
- <a q_id="0112" group_id="5631" run_id="cind071fren" score="0.0">
    <answer>South Korean automobile manufacturer</answer>
    <docid>203912.xml</docid>
  - <support>
      <s_id>203912.xml</s_id>
      <s_string>Kia Motors is a South Korean automobile manufacturer with headquarters in Yangjae-dong, Seocho-gu, Seoul, South
        Korea.</s_string>
    </support>
  </a>
```

Figure 6.12: QA@CLEF 2007 Answer File Excerpt

CLEF provided four different judgments for an answer string:

- **W** (incorrect): the answer string does not contain a correct answer or the answer is not responsive.
- **U** (unsupported): the answer string contains a correct answer but the provided text snippets do not support it, or the snippets do not originate from the provided document.
- **X** (inexact): the answer string contains a correct answer and the provided text snippets support it, but the answer string is incomplete or truncated or is longer than the minimum amount of information required.
- **C** (correct): the answer string consists of an exact and correct answer, supported by the text snippets.

The way CLEF assessed different bilingual QA systems was by comparing their performance on the same target language. For instance, CINDI participated in the FR to EN track but we were the only group to use FR as source language. Other groups used Indonesian (IN), Romanian (RO), German (DE), Spanish (ES) and Dutch (NL) but we were all part of the same evaluation track because our target language was EN, meaning that we were using the same data collection and were supposed to obtain the same correct answers.

CLEF also advised us on the name of the files we had to submit. The first four characters stood for the name of the participating team and were followed by the current year (07), the number of the run (could be either 1 or 2 because a maximum of two runs per team was allowed) and a task identifier that included both source and target language. CINDI submitted two runs, cind071fren and cind072fren. The difference between both runs lies mainly in the length of the answers we provided. It turns out that it didn't have much of an impact on the judgment of the answers since both runs ended up with an overall accuracy of 13%. The detailed results are listed in table 6.4.

Rank	Run	R #	W #	X #	U #	% F [161]	% T [3]	% D [30]	% L [9]	Overall Accuracy %
1	wolv071roen	28	166	2	4	9.32	0.00	43.33	0.00	14.00
2	**cind072fren**	**26**	**170**	**2**	**2**	**11.18**	**0.00**	**23.33**	**11.11**	**13.00**
3	**cind071fren**	**26**	**171**	**1**	**2**	**11.18**	**0.00**	**23.33**	**11.11**	**13.00**
4	csui071inen	20	175	4	1	10.56	0.00	10.00	0.00	10.00
5	dfki071deen	14	178	6	2	4.35	0.00	23.33	0.00	7.00
6	dfki071esen	5	189	4	2	1.86	0.00	6.67	0.00	2.50
7	mqaf071nlen	0	200	0	0	0.00	0.00	0.00	0.00	0.00
8	mqaf072nlen	0	200	0	0	0.00	0.00	0.00	0.00	0.00

Table 6.4: QA@CLEF 2007 Results

The test set released by QA@CLEF 2007 consisted of 200 questions, 161 of which were Factoid questions, 30 were Definition and 9 were Closed List. Among those questions there were also 3 Temporally Restricted questions.

CINDI_QA was ranked second and third out of eight submitted runs, trailing the first ranked group only by one percent in overall accuracy. CINDI_QA was much better at answering Definition questions, as evidenced by its 23.33% rate, compared to other types of questions. This is no surprise because D questions are easier to understand and answer. Moreover, CINDI_QA employs three distinct templates to handle D questions as described in chapter 4: the Living Person Definition template, the Deceased Person Definition and the Object Definition template. So because half of the templates used by CINDI_QA are aimed at D questions, CINDI_QA's efficiency to answer those questions is expected to be higher than the rest.

CINDI_QA obtained the highest answering percentage in Factoid questions at 11.18%. This is also attributed to the success of CINDI_QA's other templates – the Person Data of Birth template and the Person Date of Death template – and especially the Person Info template which represents the perfect prototype of a Person Factoid question with its "What is the Y of X" construct. This template had a lot of success because several questions asked about the job of people where "Y" would match "job" as explained in chapter 4.

Finally, CINDI_QA was the only system able to answer a Closed List question and this was done without matching any template. Indeed, benefiting from the collaboration of the Link Parser, WordNet and Lucene, CINDI_QA was able to locate and extract the correct answer to one L question.

Chapter 7

Conclusion and Future Work

7.1 Contribution

CINDI_QA represents the first attempt of the CINDI group at tackling multilingual Question Answering. We have detailed the architecture of the system which consists of several open-source tools collaborating with each other in order to find and return the best possible answer to the user.

The Link Parser is the module responsible for semantically parsing the input question and identifying keywords such as nouns, verbs and adjectives, with the user confirming the result. The Link Parser was originally written in C so we had to use JNCLGI, a Java interface to that tool.

WordNet is responsible for providing synonyms of some keywords and CINDI_QA retains the ones selected by the user. This helps widen the range of the search query by including those synonyms when looking for the answer.

Lucene has the job of indexing the data collection as well as searching through it and assembling a list of the documents that contain candidate answers. Lucene is a library that is written in Java so it fitted very nicely into the project and generated quick response times.

Last but certainly not least, CINDI_QA makes extensive use of templates. There are six templates employed by the system, each targeting specific types of questions. When a template is matched, CINDI_QA acts as a complete black-box and directly returns the correct answer.

The performance of CINDI_QA has been assessed by participating in the 2007 edition of the QA@CLEF competition, an annual conference focusing on multilingual QA. The details of our involvement in QA@CLEF are described in a paper that we wrote and that got published in the CLEF 2007 Working Notes [2].

The two runs generated by CINDI_QA that we submitted to QA@CLEF 2007 allowed us to be ranked second and third out of eight participants with an overall accuracy of 13%. This value is encouraging because the 2007 edition of QA@CLEF represented the first participation of the CINDI group and also because the CINDI group only invested 1 man/year on the project.

7.2 Limitations

CINDI_QA has a few limitations and they are two-fold.

On the one hand, the approach used to create the system is bound by several restrictions. First, the templates used are exclusive of each other, i.e. we cannot combine them in one question. For example, CINDI_QA cannot answer a question like "Who is Kobe Bryant and when was he born?" which is a concatenation of the first and third templates.

Second, Capitalized Entities must be complete and written correctly. So for CINDI_QA to find information about a CE X, X must be comprised of the first and last names of the person and shouldn't have any typing mistake.

Third, the user must already know whether or not the person he is inquiring about is alive or deceased. Indeed, CINDI_QA distinguishes between the two cases and has different templates for both situations (T1 and T2).

On the other hand, the technologies used by CINDI_QA also have some limitations.

The Link Parser is a wonderful tool but it has difficulty correctly parsing long sentences entered as input. When no template is matched and no linkages are found, CINDI_QA cannot process the long question and doesn't find any answer. This happens because the API of the Link Parser was meant to be handled in the C language whereas we tried to benefit from it using its Java interface, JNCLGI.

In parallel, the quality of the translation of the French question depends solely on the performance of the tool CINDI_QA used for that purpose. Therefore, if from the start we obtain a translation that isn't accurate or complete, CINDI_QA has no ability to rectify that and fails to produce a correct answer.

7.3 Suggested Enhancements

We had the opportunity of assessing the performance of CINDI_QA by participating in the 2007 edition of the QA@CLEF track. While the two runs we submitted ranked second and third out of eight, there is still room for much more improvement. The following is a list of encountered problems and their suggested enhancements.

- Tools improvement: In the previous section, we mentioned the limitations CINDI_QA had with respect to the tools it used. While the issue of translation quality will only be solved when the available translation engines start generating better results, the semantic parsing can be made more solid if another tool is used, especially one intended for Java.
- Online Portability: Nowadays, most QA systems have a web-based interface that makes them available online, as mentioned in section 2.3. Therefore, an important enhancement of CINDI_QA should be to provide an online interface to the

system. In addition to making CINDI_QA easily accessible, this option would also facilitate the communication between the system and the Online Translator because of the simplicity of the implementation if done using JSP and Servlets.

- Number of Templates: There are six templates used by CINDI_QA. Although this number might seem more than enough for the scope of this project, many more templates should be used to build a solid QA system. The number of templates needed is relative to the range of the data collection. Indeed, having a small and restrictive dataset enables the creation of templates that expect questions related to that same small dataset and makes the system more robust as a whole. Since the data collection used by CINDI_QA is a static version of Wikipedia which has information about pretty much anything, CINDI_QA should be strengthened by increasing the number of templates it uses.
- NIL answers: There should be a difference in handling questions that do not have an answer in the data collection and those who do have one but whose answer wasn't found. CINDI_QA doesn't differentiate between the two scenarios and displays a "did not find an answer" message in both cases. This calls for a great level of intelligence on behalf of a QA system and certainly represents the most challenging improvement that can be made to CINDI_QA.

References

[1] Desai, B.C., “Concordia INdexing and DIscovery System”, paper presented at the First Canadian Database Research Workshop, UQAM Montreal, August 1999.

[2] Haddad, C. and Desai, B. C., “Cross Lingual Question Answering using CINDI_QA for QA@CLEF 2007”, in Working Notes for the CLEF 2007 Workshop, 19-21 September, Budapest, Hungary. <http://www.clef-campaign.org/2007/working_notes/HaddadCLEF2007.pdf>, visited March 10, 2008.

[3] Wikipedia definition of “Information Retrieval” <http://en.wikipedia.org/wiki/Information_retrieval>, visited March 3, 2008.

[4] Teitelman, W., "FLIP - A Format List Processor", MAC Memo M-263, MIT, Cambridge, Massachusetts, September 1965. <http://portal.acm.org/citation.cfm?id=807968 >, visited March 1, 2008.

[5] Woods, W.A., Kaplan, R.M. and Webber, B.N., “The Lunar Sciences Natural Language Information System: Final Report”, BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972. <http://portal.acm.org/citation.cfm?id=511804&dl= >, visited February 22, 2008.

[6] Stratica, N., “NLPQC: A Natural Language Processor for Querying CINDI”, Master Thesis, Department of Software Engineering and Computer Science, Concordia University, 2002.

[7] Desai, B.C., Haddad, S.S. and Ali, A., “Automatic Semantic Header Generator”, Proceedings of ISMIS 2000, Springer-Verlag, Charlotte, North Carolina, pages 444-452.

[8] Turing, A. M., “Computing Machinery and Intelligence”, in Mind A Quarterly Review of Psychology and Philosophy, Volume 59, Number 236, 1950, pages 433-460. <http://www.loebner.net/Prizef/TuringArticle.html>, visited March 3, 2008.

[9] The Ask.com Search Engine
<http://www.ask.com/>, visited March 3, 2008.

[10] Katz, B., “Annotating the World Wide Web using Natural Language”, in Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97), 1997.
<http://groups.csail.mit.edu/infolab/publications/Katz-RIAO97.pdf>, visited March 3, 2008.

[11] Giampiccolo, D., Forner, P., Peñas, A., Ayache, C., Cristea, D., Jijkoun, V., Osenova, P., Rocha, P., Sacaleanu, B. and Sutcliffe, R., “Overview of the CLEF 2007

Multilingual Question Answering Track", in Working Notes for the CLEF 2007 Workshop, 19-21 September, Budapest, Hungary.
<http://www.clef-campaign.org/2007/working_notes/CLEF2007WN-Contents.html>, visited March 3, 2008.

[12] Sleator, D. and Temperley, D., "Parsing English with a Link Grammar", Carnegie Mellon University, Computer Science technical report CMU-CS-91-196, 1991.
<http://www.link.cs.cmu.edu/link/>, visited March 1, 2008.

[13] Miller, A.G. et al., "WordNet – A lexical database for the English language", in Communications of the ACM, 38 (1), November 1995, pp. 39-41, ACM Press, New York, ISSN:0001-0782.
<http://wordnet.princeton.edu/links#semantic>, visited March 1, 2008.

[14] Google Translate
<http://translate.google.com/translate_t>, visited March 1, 2008.

[15] Babel Fish Translation
<http://babelfish.altavista.com/>, visited March 1, 2008.

[16] Systran Box
<http://www.systransoft.com/>, visited March 1, 2008.

[17] Gospodnetic, O. and Hatcher, E., "Lucene in Action", Manning, 2005.

[18] The Apache Software Foundation's Lucene
<http://lucene.apache.org/>, visited March 3, 2008.

[19] Wikipedia, the Free Encyclopedia
<http://en.wikipedia.org/wiki/Main_Page>, visited March 8, 2008.

[20] Wiktionary definition of "Template"
<http://en.wiktionary.org/wiki/template>, visited March 1, 2008.

[21] Sun Microsystems
<http://www.sun.com/>, visited March 3, 2008.

[22] The Eclipse Foundation
<http://www.eclipse.org/>, visited March 8, 2008.

[23] Java Native Code Link Grammar Interface
<http://chrisjordan.ca/projects>, visited March 1, 2008.

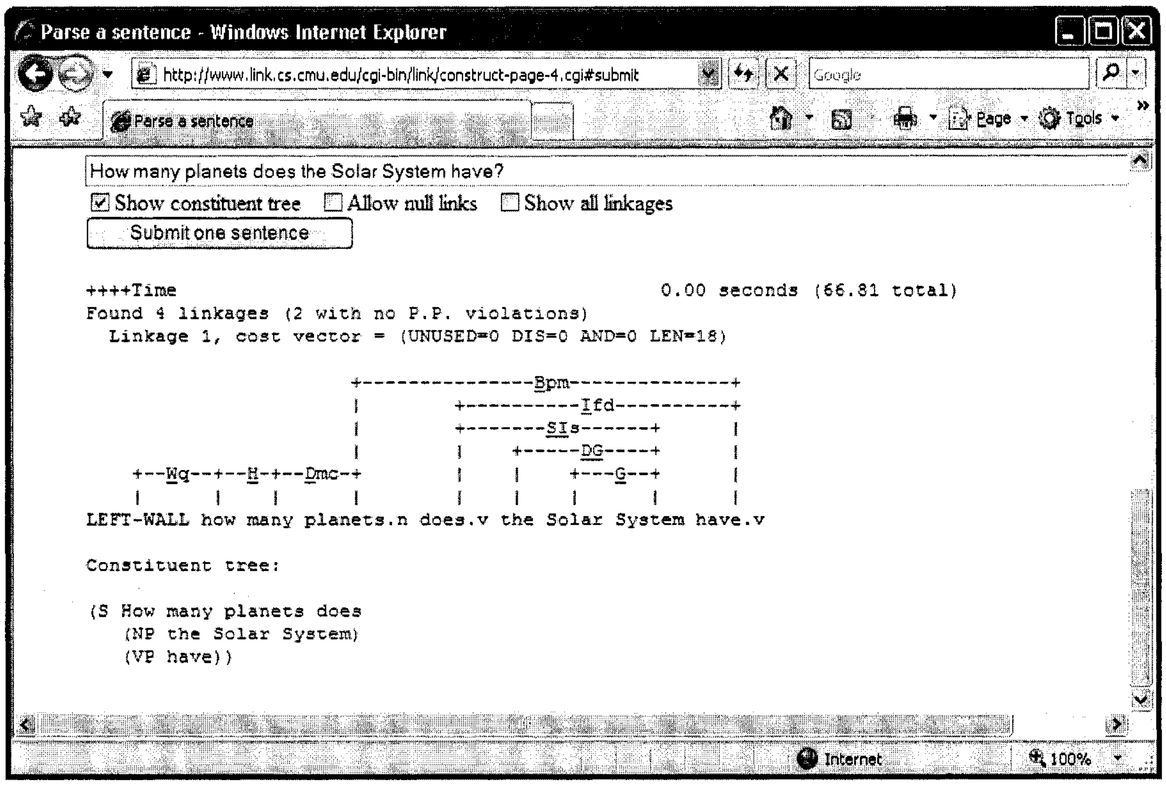
Appendix A: Link Parser

The Link Grammar Parser can be downloaded from http://www.link.cs.cmu.edu/link. NSF, ARPA, and the School of Computer Science, at Carnegie Mellon University, supported this research project.

The Link Grammar Parser is a syntactic parser of English, based on link grammar, an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. As of March 2008, the authors released version 4.0 of the parser. Among the new features of version 4.0 is a system that derives a constituent or phrase-structure representation of a sentence (showing noun phrases, verb phrases, etc.) The thesis implementation integrates the Link Parser at the source code level.

The entire system is available for download on the web. The system is written in generic ANSI C, and runs on all platforms with a C compiler. There is an application program interface (API) to make it easy to incorporate the parser into other applications. The parser has a dictionary of about 60000 word forms. It has coverage of a wide variety of syntactic constructions, including many rare and idiomatic ones. The parser is robust; it is able to skip over portions of the sentence that it cannot understand, and assign some structure to the rest of the sentence. It is able to handle unknown vocabulary, and make intelligent guesses from context and spelling about the syntactic categories of unknown words. It has knowledge of capitalization, numerical expressions, and a variety of punctuation symbols [12].

Figure A.1 shows an actual output from the online Link Grammar Parser. The input phrase is "How many planets does the Solar System have?"


Parse a sentence - Windows Internet Explorer
http://www.link.cs.cmu.edu/cgi-bin/link/construct-page-4.cgi#submit
Google
Parse a sentence
Page
Tools
Internet
100%

How many planets does the Solar System have?

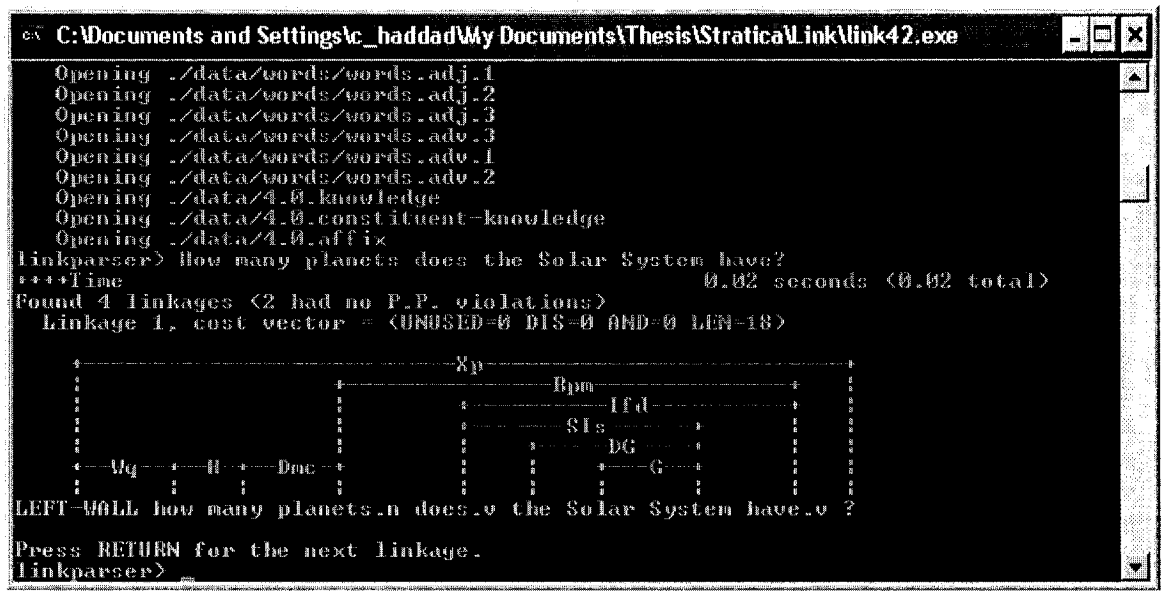
[✓] Show constituent tree ☐ Allow null links ☐ Show all linkages

Submit one sentence

```
++++Time                                          0.00 seconds (66.81 total)
Found 4 linkages (2 with no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=18)

                       +---------------Bpm--------------+
                       |        +----------Ifd----------+
                       |        +-------SIs------+      |
                       |        |    +-----DG----+      |
    +--Wq--+--H-+--Dmc-+        |    |    +---G--+      |
    |      |    |      |        |    |    |      |      |
LEFT-WALL how many planets.n does.v the Solar System have.v

Constituent tree:

(S How many planets does
   (NP the Solar System)
   (VP have))
```

Figure A.1: LinkParser Web Interface

Below it is the same output generated by the compiled version (ANSI C).



```
C:\Documents and Settings\c_haddad\My Documents\Thesis\Stratica\Link\link42.exe
   Opening ./data/words/words.adj.1
   Opening ./data/words/words.adj.2
   Opening ./data/words/words.adj.3
   Opening ./data/words/words.adv.3
   Opening ./data/words/words.adv.1
   Opening ./data/words/words.adv.2
   Opening ./data/4.0.knowledge
   Opening ./data/4.0.constituent-knowledge
   Opening ./data/4.0.affix
linkparser> How many planets does the Solar System have?
++++Time                                          0.02 seconds (0.02 total)
Found 4 linkages (2 had no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=18)

    +--------------------------Xp---------------------------+
    |                   +--------------Rpm---------------+  |
    |                   |        +----------Ifd----------+  |
    |                   |        +------SIs-------+      |  |
    |                   |        |    +----DG-----+      |  |
    +--Wq--+-H--+--Dmc--+        |    |    +--G---+      |  |
    |      |    |       |        |    |    |      |      |  |
LEFT-WALL how many planets.n does.v the Solar System have.v ?

Press RETURN for the next linkage.
linkparser> _
```

Figure A.2: LinkParser Compiled Interface

Here is the summary of the links [12] found on the Internet at http://www.link.cs.cmu.edu/link:

A connects pre-noun ("attributive") adjectives to following nouns: "The BIG DOG chased me", "The BIG BLACK UGLY DOG chased me".

AA is used in the construction "How [adj] a [noun] was it?". It connects the adjective to the following "a".

AF connects adjectives to verbs in cases where the adjective is fronted, such as questions and indirect questions: "How BIG IS it?"

AL connects a few modifiers like "all" or "both" to following modifiers: "ALL THE people are here".

AM connects "as" to "much" or "many": "I don't go out AS MUCH now".

AN connects noun-modifiers to following nouns: "The TAX PROPOSAL was rejected".

AZ connects the word "as" back to certain verbs that can take "[obj] as [adj]" as a complement: "He VIEWED him AS tall".

B serves various functions involving relative clauses and questions. It connects transitive verbs back to their objects in relative clauses, questions, and indirect questions ("The DOG we CHASED", "WHO did you SEE?"); it also connects the main noun to the finite verb in subject-type relative clauses ("The DOG who CHASED me was black").

BI connects forms of the verb "be" to certain idiomatic expressions: for example, cases like "He IS PRESIDENT of the company".

BT is used with time expressions acting as fronted objects: "How many YEARS did it LAST?"

BW connects "what" to various verbs like "think", which are not really transitive but can connect back to "what" in questions: "WHAT do you THINK?"

C links conjunctions to subjects of subordinate clauses ("He left WHEN HE saw me"). It also links certain verbs to subjects of embedded clauses ("He SAID HE was sorry").

CC connects clauses to following coordinating conjunctions ("SHE left BUT we stayed").

CO connects "openers" to subjects of clauses: "APPARENTLY / ON Tuesday, THEY went to a movie".

CP connects paraphrasing or quoting verbs to the wall (and, indirectly, to the paraphrased expression): "///// That is untrue, the spokesman SAID."

CQ connects to auxiliaries in comparative constructions involving s-v inversion: "SHE has more money THAN DOES Joe".

CX is used in comparative constructions where the right half of the comparative contains only an auxiliary: "She has more money THAN he DOES".

D connects modifiers to nouns: "THE DOG chased A CAT and SOME BIRDS".

DD connects definite modifiers ("the", "his") to certain things like number expressions and adjectives acting as nouns: "THE POOR", "THE TWO he mentioned".

DG connects the word "The" with proper nouns: "the Riviera", "the Mississippi".

DP connects possessive modifiers to gerunds: "YOUR TELLING John to leave was wrong".

DT connects modifiers to nouns in idiomatic time expressions: "NEXT WEEK", "NEXT THURSDAY".

E is used for verb-modifying adverbs which precede the verb: "He is APPARENTLY LEAVING".

EA connects adverbs to adjectives: "She is a VERY GOOD player".

EB connects adverbs to forms of "be" before an object or prepositional phrase: "He IS APPARENTLY a good programmer".

EC connects adverbs to comparative adjectives: "It is MUCH BIGGER".

EE connects adverbs to other adverbs: "He ran VERY QUICKLY".

EF connects the word "enough" to preceding adjectives and adverbs: "He didn't run QUICKLY ENOUGH".

EI connects a few adverbs to "after" and "before": "I left SOON AFTER I saw you".

EL connects certain words to the word "else": something / everything / anything / nothing, somewhere (etc.), and someone (etc.).

EN connects certain adverbs to expressions of quantity: "The class has NEARLY FIFTY students".

ER is used the expression "The x-er..., the y-er...". It connects the two halves of the expression together, via the comparative words (e.g. "The FASTER it is, the MORE they will like it").

EZ connects certain adverbs to the word "as", like "just" and "almost": "You're JUST AS good as he is".

FL connects "for" to "long": "I didn't wait FOR LONG"

FM connects the preposition "from" to various other prepositions: "We heard a scream FROM INSIDE the house".

G connects proper noun words together in series: "GEORGE HERBERT WALKER BUSH is here".

GN (stage 2 only) connects a proper noun to a preceding common noun which introduces it: "The ACTOR Eddie MURPHY attended the event".

H connects "how" to "much" or "many": "HOW MUCH money do you have".

I connects infinitive verb forms to certain words such as modal verbs and "to": "You MUST DO it", "I want TO DO it".

ID is a special class of link-types generated by the parser, with arbitrary four-letter names (such as "IDBT"), to connect together words of idiomatic expressions such as "at_hand" and "head_of_state".

IN connects the preposition "in" to certain time expressions: "We did it IN DECEMBER".

J connects prepositions to their objects: "The man WITH the HAT is here".

JG connects certain prepositions to proper-noun objects: "The Emir OF KUWAIT is here".

JQ connects prepositions to question-word modifiers in "prepositional questions": "IN WHICH room were you sleeping?"

JT connects certain conjunctions to time-expressions like "last week": "UNTIL last WEEK, I thought she liked me".

K connects certain verbs with particles like "in", "out", "up" and the like: "He STOOD UP and WALKED OUT".

L connects certain modifiers to superlative adjectives: "He has THE BIGGEST room".

LE is used in comparative constructions to connect an adjective to the second half of the comparative expression beyond a complement phrase: "It is more LIKELY that Joe will go THAN that Fred will go".

LI connects certain verbs to the preposition "like": "I FEEL LIKE a fool."

M connects nouns to various kinds of post-noun modifiers: prepositional phrases ("The MAN WITH the hat"), participle modifiers ("The WOMAN CARRYING the box"), prepositional relatives ("The MAN TO whom I was speaking"), and other kinds.

MF is used in the expression "Many people were injured, SOME OF THEM children".

MG allows certain prepositions to modify proper nouns: "The EMIR OF Kuwait is here".

MV connects verbs and adjectives to modifying phrases that follow, like adverbs ("The dog RAN QUICKLY"), prepositional phrases ("The dog RAN IN the yard"), subordinating conjunctions ("He LEFT WHEN he saw me"), comparatives, participle phrases with commas, and other things.

MX connects modifying phrases with commas to preceding nouns: "The DOG, a POODLE, was black". "JOHN, IN a black suit, looked great".

N connects the word "not" to preceding auxiliaries: "He DID NOT go".

ND connects numbers with expressions that require numerical modifiers: "I saw him THREE WEEKS ago".

NF is used with NJ in idiomatic number expressions involving "of": "He lives two THIRDS OF a mile from here".

NI is used in a few special idiomatic number phrases: "I have BETWEEN 5 AND 20 dogs".

NJ is used with NF in idiomatic number expressions involving "of": "He lives two thirds OF a MILE from here".

NN connects number words together in series: "FOUR HUNDRED THOUSAND people live here".

NO is used on words which have no normal linkage requirement, but need to be included in the dictionary, such as "um" and "ah".

NR connects fraction words with superlatives: "It is the THIRD BIGGEST city in China".

NS connects singular numbers (one, 1, a) to idiomatic expressions requiring number modifiers: "I saw him ONE WEEK ago".

NT connects "not" to "to": "I told you NOT TO come".

NW is used in idiomatic fraction expressions: "TWO THIRDS of the students were women".

O connects transitive verbs to their objects, direct or indirect: "She SAW ME", "I GAVE HIM the BOOK".

OD is used for verbs like "rise" and "fall" which can take expressions of distance as complements: "It FELL five FEET".

OF connects certain verbs and adjectives to the word "of": "She ACCUSED him OF the crime", "I'm PROUD OF you".

ON connects the word "on" to dates or days of the week in time expressions: "We saw her again ON TUESDAY".

OT is used for verbs like "last" which can take time expressions as objects: "It LASTED five HOURS".

OX is an object connector, analogous to SF, used for special "filler" words like "it" and "there" when used as objects: "That MAKES IT unlikely that she will come".

P connects forms of the verb "be" to various words that can be its complements: prepositions, adjectives, and passive and progressive participles: "He WAS [ANGRY / IN the yard / CHOSEN / RUNNING]".

PF is used in certain questions with "be", when the complement need of "be" is satisfied by a preceding question word: "WHERE are you?", "WHEN will it BE?"

PP connects forms of "have" with past participles: "He HAS GONE".

Q is used in questions. It connects the wall to the auxiliary in simple yes-no questions ("///// DID you go?"); it connects the question word to the auxiliary in where-when-how questions ("WHERE DID you go").

QI connects certain verbs and adjectives to question-words, forming indirect questions: "He WONDERED WHAT she would say".

R connects nouns to relative clauses. In subject-type relatives, it connects to the relative pronoun ("The DOG WHO chased me was black"); in object-type relatives, it connects either to the relative pronoun or to the subject of the relative clause ("The DOG THAT we chased was black", "The DOG WE chased was black").

RS is used in subject-type relative clauses to connect the relative pronoun to the verb: "The dog WHO CHASED me was black".

RW connects the right-wall to the left-wall in cases where the right-wall is not needed for punctuation purposes.

S connects subject nouns to finite verbs: "The DOG CHASED the cat": "The DOG [IS chasing / HAS chased / WILL chase] the cat".

SF is a special connector used to connect "filler" subjects like "it" and "there" to finite verbs: "THERE IS a problem", "IT IS likely that he will go".

SFI connects "filler" subjects like "it" and "there" to verbs in cases with subject-verb inversion: "IS THERE a problem?", "IS IT likely that he will go?"

SI connects subject nouns to finite verbs in cases of subject-verb inversion: "IS JOHN coming?", "Who DID HE see?"

SX connects "I" to special first-person verbs like "was" and "am".

SXI connects "I" to first-person verbs in cases of s-v inversion.

TA is used to connect adjectives like "late" to month names: "We did it in LATE DECEMBER".

TD connects day-of-the-week words to time expressions like "morning": "We'll do it MONDAY MORNING".

TH connects words that take "that [clause]" complements with the word "that". These include verbs ("She TOLD him THAT..."), nouns ("The IDEA THAT..."), and adjectives ("We are CERTAIN THAT").

TI is used for titles like "president", which can be used in certain circumstances without a modifier: "AS PRESIDENT of the company, it is my decision".

TM is used to connect month names to day numbers: "It happened on JANUARY 21".

TO connects verbs and adjectives which take infinitival complements to the word "to": "We TRIED TO start the car", "We are EAGER TO do it".

TQ is the modifier connector for time expressions acting as fronted objects: "How MANY YEARS did it last".

TS connects certain verbs that can take subjunctive clauses as complements - "suggest", "require" - to the word that: "We SUGGESTED THAT he go".

TW connects days of the week to dates in time expressions: "The meeting will be on MONDAY, JANUARY 21".

TY is used for certain idiomatic usages of year numbers: "I saw him on January 21, 1990 ". (In this case it connects the day number to the year number.)

U is a special connector on nouns, which is disjoined with both the modifier and subject-object connectors. It is used in idiomatic expressions like "What KIND_OF DOG did you buy?"

UN connects the words "until" and "since" to certain time phrases like "after [clause]": "You should wait UNTIL AFTER you talk to me".

V connects various verbs to idiomatic expressions that may be non-adjacent: "We TOOK him FOR_GRANTED", "We HELD her RESPONSIBLE".

W connects the subjects of main clauses to the wall, in ordinary declaratives, imperatives, and most questions (except yes-no questions). It also connects coordinating conjunctions to following clauses: "We left BUT SHE stayed".

WN connects the word "when" to time nouns like "year": "The YEAR WHEN we lived in England was wonderful".

WR connects the word "where" to a few verbs like "put" in questions like "WHERE did you PUT it?"

X is used with punctuation, to connect punctuation symbols either to words or to each other. For example, in this case, POODLE connects to commas on either side: "The dog, a POODLE, was black."

Y is used in certain idiomatic time and place expressions, to connect quantity expressions to the head word of the expression: "He left three HOURS AGO", "She lives three MILES FROM the station".

YP connects plural noun forms ending in s to "'" in possessive constructions: "The STUDENTS ' rooms are large".

YS connects nouns to the possessive suffix "'s": "JOHN 'S dog is black".

Z connects the preposition "as" to certain verbs: "AS we EXPECTED, he was late".

Appendix B: WordNet

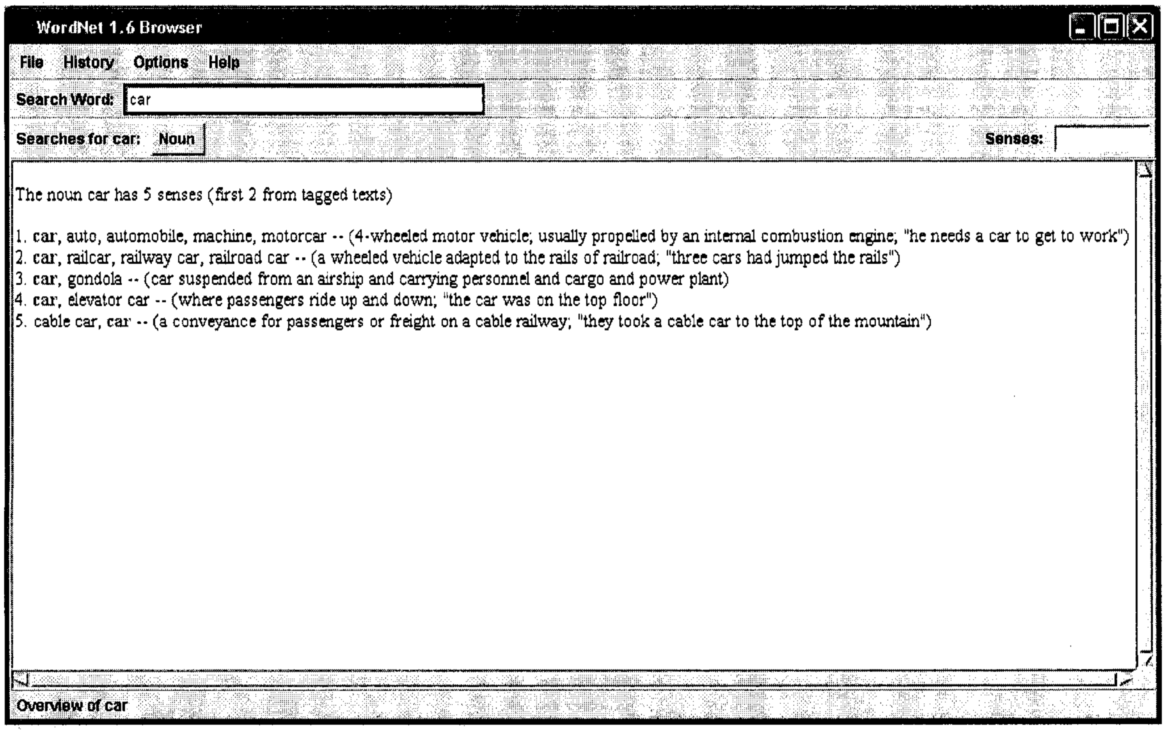
WordNet® is an online lexical reference system whose design is inspired by psycholinguistic theories of human lexical memory (http://www.cogsci.princeton.edu/~wn). English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. WordNet was developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller.

The WordNet system consists of lexicographer files, code to convert these files into a database, and search routines and interfaces that display information from the database. The lexicographer files organize nouns, verbs, adjectives and adverbs into groups of synonyms, and describe relations between synonym groups.

Information in WordNet is organized around logical groupings called synsets. Each synset consists of a list of synonymous words or collocations (e.g. "fountain pen" , "take in"), and pointers that describe the relations between this synset and other synsets. A word or collocation may appear in more than one synset, and in more than one part of speech. The words in a synset are logically grouped such that they are interchangeable in some context.

Two kinds of relations are represented by pointers: lexical and semantic. Lexical relations hold between word forms; semantic relations hold between word meanings. These relations include hypernymy / hyponymy, antonymy, entailment, and meronymy / holonymy.

Nouns and verbs are organized into hierarchies based on the hypernymy / hyponymy relation between synsets. Additional pointers are be used to indicate other relations. Adjectives are arranged in clusters containing head synsets and satellite synsets. Each cluster is organized around antonymous pairs (and occasionally antonymous triplets). The antonymous pairs (or triplets) are indicated in the head synsets of a cluster. Most head synsets have one or more satellite synsets, each of which represents a concept that is similar in meaning to the concept represented by the head synset. One way to think of the adjective cluster organization is to visualize a wheel, with a head synset as the hub and satellite synsets as the spokes. Two or more wheels are logically connected via antonymy, which can be thought of as an axle between the wheels [13].



```
The noun car has 5 senses (first 2 from tagged texts)

1. car, auto, automobile, machine, motorcar -- (4-wheeled motor vehicle; usually propelled by an internal combustion engine; "he needs a car to get to work")
2. car, railcar, railway car, railroad car -- (a wheeled vehicle adapted to the rails of railroad; "three cars had jumped the rails")
3. car, gondola -- (car suspended from an airship and carrying personnel and cargo and power plant)
4. car, elevator car -- (where passengers ride up and down; "the car was on the top floor")
5. cable car, car -- (a conveyance for passengers or freight on a cable railway; "they took a cable car to the top of the mountain")
```

Figure B.1: WordNet Window-based browser Interface

Glossary of lexical & semantic relations in WordNet

Entailment: A verb X entails Y if X cannot be done unless Y is, or has been, done.

Holonym: The name of the whole of which the meronym names a part. Y is a holonym of X if X is a part of Y.

Hypernym: The generic term used to designate a whole class of specific instances. Y is a hypernym of X if X is a (kind of) Y.

Hyponym: The specific term used to designate a member of a class. X is a hyponym of Y if X is a (kind of) Y.

Indirect antonym: An adjective in a satellite synset that does not have a direct antonym has an indirect antonym via the direct antonym of the head synset.

Meronym: The name of a constituent part of, the substance of, or a member of something. X is a meronym of Y if X is a part of Y.

Pertainym: A relational adjective. Adjectives that are pertainyms are usually defined by such phrases as "of or pertaining to" and do not have antonyms. A pertainym can point to a noun or another pertainym.

Subordinate: Same as hyponym.

Superordinate: Same as hypernym.

Synset: A synonym set; a set of words that are interchangeable in some context.

Troponym: A verb expressing a specific manner elaboration of another verb. X is a troponym of Y if to X is to Y in some manner.