Cross Lingual Question Answering using CINDI_QA for QA@CLEF 2007

Chedid Haddad, Bipin C. Desai
Department of Computer Science and Software Engineering
Concordia University
1455 De Maisonneuve Blvd. W.
Montreal, Quebec H3G 1M8, Canada
{c_haddad, bcdesai}@cs.concordia.ca

Abstract

This article presents the first participation of the CINDI group in the Multiple Language Question Answering Cross Language Evaluation Forum (QA@CLEF). We participated in a track using French as the source language and English as the target language. CINDI_QA first uses an online translation tool to convert the French input question into an English sentence. Second, a Natural Language Parser extracts keywords such as verbs, nouns, adjectives and capitalized entites from the query. Third, synonyms of those keywords are generated thanks to a Lexical Reference module. Fourth, our integrated Searching and Indexing component localises the answer candidates from the QA@CLEF data collection provided to us. Finally, the candidates are matched against our existing set of templates to decide on the best answer to return to the user. Two runs were submitted. Having missed using a large part of the corpora, CINDI_QA was only able to generate correct and exact answers for 13% of the questions it received.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; I.2 [**Artificial Intelligence**]: I.2.7 Natural Language Processing

General Terms

Measurement, Performance, Experimentation

Keywords

Question answering, Questions beyond factoids, Bilingual, French, English

1 Introduction

The Concordia Index for Navigation and Discovery on the Internet (CINDI [1]) group has been founded at the Department of Computer Science and Software Engineering of Concordia in the late 1990s. Its purpose is the continuous enhancement of information discovery and retrieval. Hence, CINDI_QA has been created to tackle bilingual question answering within the spectrum of QA@CLEF 2007.

The paper is organized in the following way: we first go through the system overview with an emphasis on architecture. We then list the tools incorporated in CINDI_QA. Afterwards, we take a look at the template matching mechanism that drives the system and its application for the QA@CLEF participation and the results obtained. The conclusion follows where we highlight what we learned and how we intend to improve CINDI_QA's performance for the future editions of QA@CLEF.

2 System Overview

CINDI_QA is made up of one central unit called the Processor, a couple of peripheral components, four integrated tools and a template module. These are illustrated in the figure below and elaborated on in the next section.

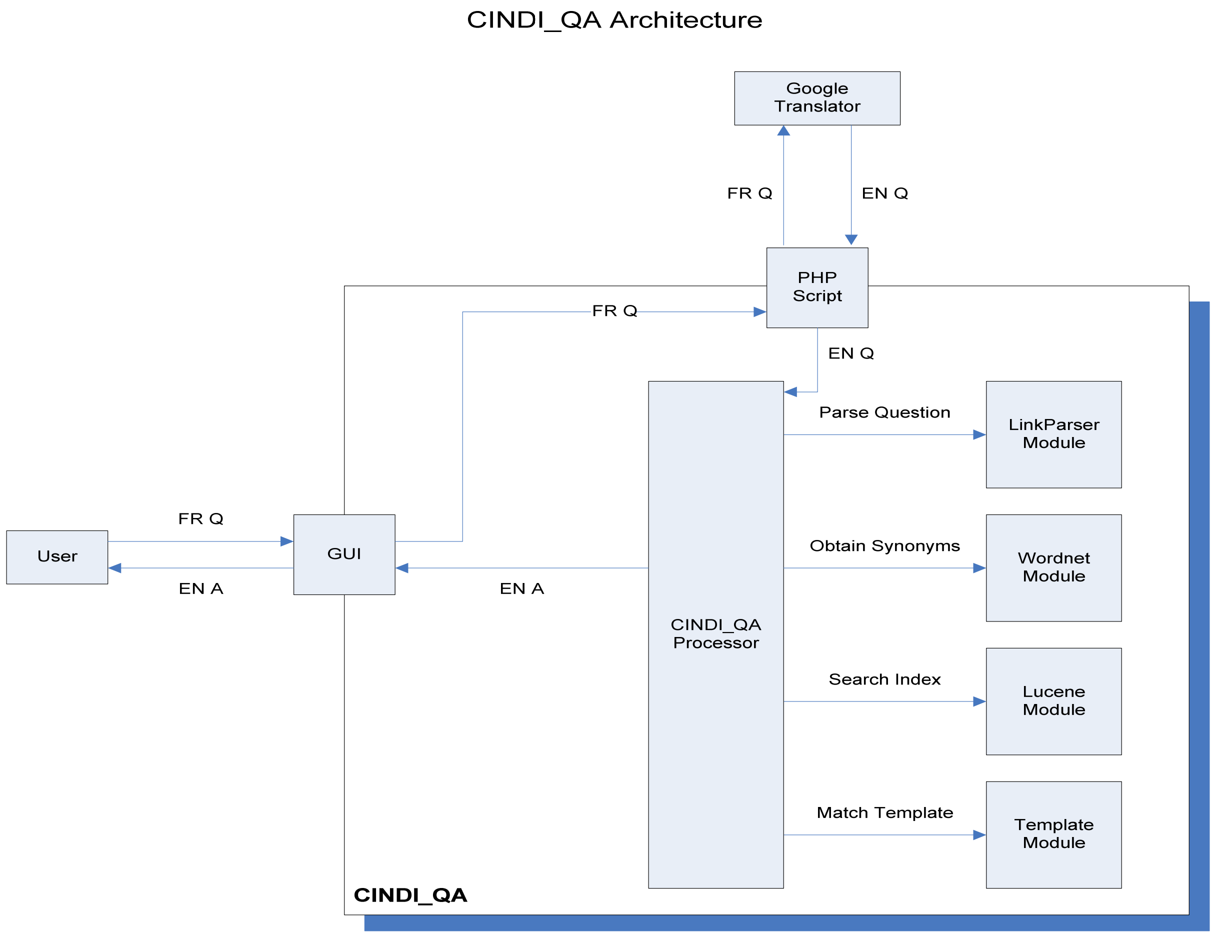

Figure 1: CINDI_QA Architecture

The system's "brain" is located in the CINDI_QA Processor where all the logical inferences are made. Since CINDI_QA communicates with several tools, the information it retrieves from those modules - which is meaningless on its own - is analyzed in the Processor in a structured way that helps build the eventual answer. This process is done in a pre-defined order, with the Processor getting data from one module, sorting it out then using it to probe the next module.

One of the peripheral components is the PHP Script that acts as an interface between the Online Translator and the Processor. Its purpose is to send the French question to the Online Translator and bring its English equivalent back. The other peripheral unit is the Graphical User Interface (GUI) which is the façade of our system for the user; it will prompt for a French question and return the English answer.

In a typical scenario, the question inputted in French by the user is translated to English then parsed to extract the keywords. Afterward, the synonyms of the keywords are obtained to form an internal query sent to the Search engine that already has the CLEF data indexed. The answer candidates are localized at which point they are matched against a pre-existing set of templates that enables the selection of the best answer. That answer, which is in English, is sent back to the user.

In order to improve performance, CINDI_QA allows user interaction to direct its flow of operations. Actually, CINDI_QA can be run in two modes. In the automatic mode, it acts as a complete black box by only taking a question and returning an answer. In the feedback mode, it has the possibility of disambiguating the query by prompting the user after each stage. For example, since certain synonyms of a word often don't suit a particular context, the user can choose to eliminate them and only retain relevant ones.

The process flow of CINDI_QA in the feedback mode is shown in the following diagram.

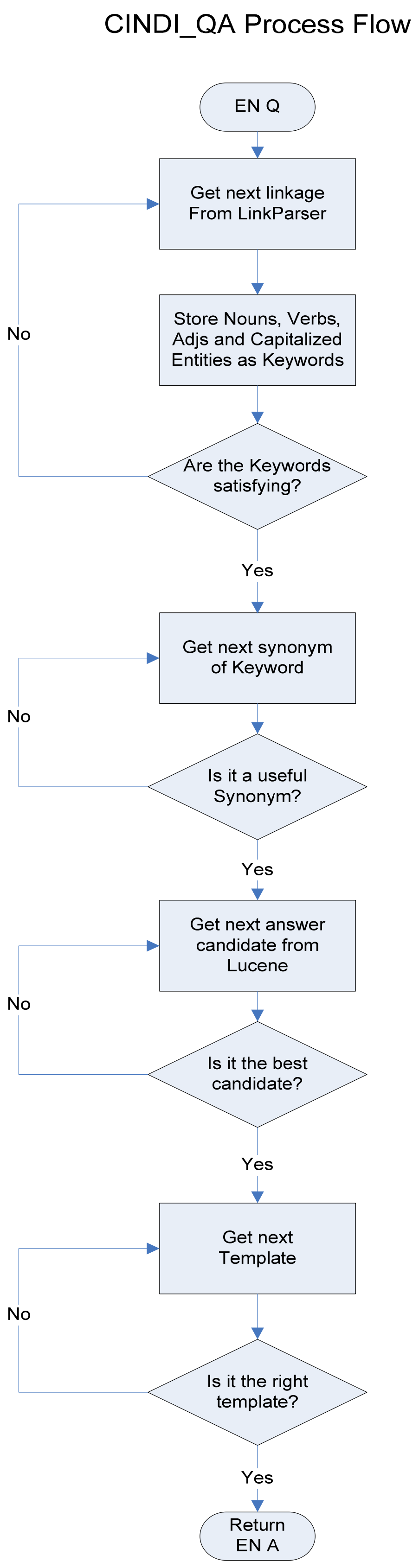
CINDI_QA Process Flow
EN Q
Get next linkage From LinkParser
Store Nouns, Verbs, Adjs and Capitalized Entities as Keywords
Are the Keywords satisfying?
No
Yes
Get next synonym of Keyword
Is it a useful Synonym?
No
Yes
Get next answer candidate from Lucene
Is it the best candidate?
No
Yes
Get next Template
Is it the right template?
No
Yes
Return EN A

Figure 2: CINDI_QA Process Flow

The next section lists the tools used to build CINDI_QA.

3 Tools Integration

3.1 Online Translation

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 the Google [2], Babel Fish [3] and Systran [4] translators appear to be powered by the same engine because they offer the same result when asked to translate from French to English.

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

3.2 Natural Language Parsing

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 [5], 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. This tool is written in generic C code, but since CINDI_QA has been programmed in Java, we used the Java Native Code Link Grammar Interface [6].

The Link Parser is plugged into our system to generate linkages of the 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. Remember that the user has the option of choosing the most appropriate linkage.

3.3 Lexical Reference

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 [7] is a lexical database for the English language developed at Princeton University and has been used in other CINDI related projects [10] 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* [8].

After defining the keywords using the Link Parser, we queried 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 to discard them and select only the appropriate ones.

3.4 Document Indexing and Searching

We needed a tool that could not only index all the documents that make up QA@CLEF's 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 [9], a high-performance, full-featured text search engine library written entirely in Java.

CINDI_QA makes extensive use of Lucene. As mentioned before, it is used in concordance with WordNet to get synonyms of keywords. Lucene also creates the CLEF index and ranks the results found. The following features are of great importance to our system.

3.4.1 Lucene 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 spread out in a paragraph. This is done by adding the tilde character '~' and a number to the end of the query.

3.4.2 Highlighting query terms in Answer Candidates

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 take advantage of the Lucene Highlighter, a wonderful tool that actually displays snippets of text 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* [8].

4 Template Matching

CINDI_QA's template module comes into play in the final stages, between the identification of the answer candidates and the return of an answer to the user. Indeed, after the Lucene component's job is done, we are left with a set of sentences one of which holds the actual answer. To determine which one to choose, we match them against our set of pre-defined templates. We chose to use templates because a previous project [10] done by a member of our group was pretty successful at parsing English questions using mainly templates.

According to the CLEF Working Notes [11], there are three different types of questions: Factoid, Definition and List. Factoid and Definition kinds are further broken down into sub-categories: Person, Time, Location, Organization, Measure, Count, Object and Other for the former and Person, Organization, Object and Other for the latter. Our templates take advantage of this approach to differentiate among the questions that are inputted in our system, and also of the keywords we identified in the Link Parser module. The capitalized entity, made up of a word whose first letter is uppercase, identifies proper nouns. It is an important keyword because of its high frequency of occurrence; this is due to the fact that most questions at QA@CLEF ask about famous people, locations or events.

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

This is without a doubt the easiest template to consider. To discover that a question is of that type, it must start with the word "who", have the verb "is" or its derivative "was" and must be followed by a capitalized entity X. X could be one word or a composition of words; as long as adjacent words are capitalized, CINDI_QA will define them as belonging to one and the same capitalized entity. The following illustrates the query given to CINDI_QA, the candidate returned by the Highlighter of the Lucene Module and the answer returned after template matching.

Example:	- query:	*Who is Robert Altman?*
	- candidate:	Robert Bernard Altman (February 20, 1925 – November 20, 2006) was an American film director known for making films that are highly naturalistic, but with a stylized perspective.
	- answer:	American film director.

4.2 The Count Factoid Template: *How many N V?*

This template is selected if the question starts with the words "how many" and has at least one verb V and one noun N, capitalized entities being optional. In the following example, had the expression "Solar System" not been capitalized, the Link Parser module would have identified "solar" as an adjective and "system" as a noun, yet the template would still be selected because "planets" is a noun.

Example:	- query:	*How many planets does the Solar System have?*
	- candidate:	The Solar System or solar system consists of the Sun and the other celestial objects gravitationally bound to it: the eight planets, their 165 known moons, three currently identified dwarf planets (Ceres, Eris, and Pluto) and their four known moons, and billions of small bodies.
	- answer:	Eight.

4.3 The Time Factoid Template: *What year V1 X V2?* / *When V1 X V2?*

This template is selected if the following occurs: the question starts with the words "what year" or "when", then has two verbs, V1 and V2 as well as one capitalized entity X. The Link Parser module will identify two separate verbs even if they belong to the participate form of the same verb, i.e. it will flag "was" as V1 and "murdered" as V2, even though technically they both define the verb "murder". In the following example, notice how the word "assassinate", synonym of the inputted verb "murder", is used to return the correct answer.

Example:	- query:	*What year was Martin Luther King murdered?*
	- candidate:	On April 4, 1968, King was assassinated in Memphis, Tennessee.
	- answer:	1968.

In the feedback mode, the user will be prompted one last time to approve the template chosen by CINDI_QA before being shown an answer. In the black-box mode, the system chooses the least costly linkage from the Link Parser, retains only the first two synonyms of a keyword, uses the best ranked document from Lucene with the most highlighted sentence and constructs the answer based on the first template matched.

5 Results

The CINDI group participated in the FR to EN track of the QA@CLEF edition of 2007. We produced two runs that were sent: cind071fren and cind072fren. The two runs ended up with the same overall accuracy value of 13%. The following figure details the different assessments both runs obtained.

	Right	**Wrong**	**Inexact**	**Unsupported**	**Unassessed**	**Accuracy**
cind071fren	26	171	1	2	0	**13%**
cind072fren	26	170	2	2	0	**13%**

Figure 3: CINDI_QA Results at QA@CLEF 2007

In addition to the usual news collections, articles from Wikipedia were also considered as answer source for the first time this year. But we became aware of that very late in our answering process and were only able to use a small part of the corpora available. This is the main reason for our system's low performance.

Since our two runs are equivalent, the following figure shows the accuracy by question type only of cind072fren. CINDI_QA's score on definition questions is much higher than any other type of question. This result is attributed to the robustness of template 4.1.

	Factoids	**Lists**	**Definitions**
Total	161	9	30
Right	18	1	7
Wrong	140	8	22
Unsupported	2	0	0
Inexact	1	0	1
Accuracy	**11.18%**	**11.11%**	**23.33%**

Figure 4: CINDI_QA Results by Question Type

6 Conclusion and Future Works

Although the approach we used to tackle bilingual question-answering at CLEF looks good in theory, the results obtained are a bit disappointing, as is shown by our 13% success rate. Because this is our first participation in QA@CLEF and the CINDI group only invested 1 man/year on the project, we have high hope for the future, especially since we missed a large part of the source corpora.

We learned that because CINDI_QA relies on so many external tools, it is only as strong as its weakest link. For instance, if from the start, the translation of the question isn't a successful one, there is nothing the system can do after that stage to come up with a correct answer.

Future works include the addition of new templates to handle the multitude of question sub-categories as well as a mechanism to identify questions whose answer isn't located in the CLEF data collection and return NIL for those questions. We hope we can have these enhancements ready for the next edition of QA@CLEF.

References

[1] The CINDI System, http://cindi.encs.concordia.ca/about_cindi.html

[2] Google Translate, http://translate.google.com/translate_t

[3] Babel Fish Translation, http://babelfish.altavista.com/

[4] Systran Box, http://www.systransoft.com/

[5] The Link Grammar Parser, http://www.link.cs.cmu.edu/link

[6] Java Native Code Link Grammar Interface, http://chrisjordan.ca/projects

[7] WordNet, a lexical database for the English language, http://wordnet.princeton.edu/

[8] O. Gospodnetic and E. Hatcher, *Lucene in Action*, Manning, 2005.

[9] Lucene, http://lucene.apache.org/

[10] N. Stratica, *NLPQC: A Natural Language Processor for Querying CINDI*, Master Thesis, Concordia University, 2002.

[11] B. Magnini, D. Giampiccolo, P. Former, C. Ayache, V. Jijkoun, P. Osenova, A. Peñas, P. Rocha, B. Sacaleanu and R. Sutcliffe, *Overview of the CLEF 2006 Multilingual Question Answering Track*, in: Cross Language Evaluation Forum: Working Notes for the CLEF 2006 Workshop (CLEF 2006), Alicante, Spain, 20-22 September 2006.