Natural Language Module of a Hungarian Deep Web Searcher

Domonkos Tikk, Zsolt T. Kardkovács, Gábor Magyar
Dept. of Telecom. & Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Hungary
E-mails: {tikk,kardkovacs,magyar}@tmit.bme.hu

Anna Babarczy, István Szakadát
Axelero Rt.
H-1117 Budapest, Hungary,
E-mails: babarczy@eik.bme.hu, syi@axelero.hu

Abstract—**In our ongoing project, called "In the Net of Words" (NoW), we aim at creating a complex search interface that incorporates —beside the usual keyword-based search functionality—deep web search, Hungarian natural language question processing, image search support by visual thesaurus. One of the most crucial part of the system is the transformation of natural language questions to adequate SQL queries that is in accordance with schema and attribute convention of contracted partner databases. This transformation is performed in three steps: natural language question processing, context recognition, and SQL transformation. In this paper we describe in details the first step.**

I. Introduction

In the NoW project our purpose is to create a complex search interface with the following features: search in the deep web content of contracted partners' databases, processing Hungarian natural language questions and transforming them to SQL queries for database access, image search supported by a visual thesaurus that describes in a structural form the visual content of images (also in Hungarian). This paper primarily focuses on question processing and secondarily it presents briefly the idea of the context recognizer. Before going into details we give a short overview about the project's aims.

A. The deep web

The deep web is content that resides in searchable and online accessible databases, whose results can only be discovered by a direct query, as described in BrighPlanet's white paper[1]. Without the directed query, the database does not publish the result. Result pages are posted as dynamic web pages as answer to direct queries. Incorporating deep web access in the internet search engines is a very important issue. Studies about the internet [1] show among other facts that

1) The size of the deep web is about 400 times larger than that of the surface web that is accessible to traditional keyword-based search engines. The deep web contains 7500 terabyte (TB) information, compared to the 19 TB information of the surface web.
2) The size of the deep web is growing much faster than the surface web. The deep web is the category where new information appears the fastest on the web.
3) Deep web sites tend to be narrower and deeper than conventional surface web sites. Deep web sites typically reside topic specific databases, therefore the quality of the information stored on these sites are usually more adequate in the given topic than the one accessible through conventional pages.

Traditional search engines create their catalogs based on crawling web pages that have to be static and linked to other pages. Therefore dynamic web pages, even though they have unique URLs, are not searchable by such engines [2]. However, based on the above listed reasons, it would be highly desirable to make the content of the deep web accessible to search engines, which can be normally accessed only through querying the search surface of the deep web sites. Hence, the user can retrieve his/her information need from the deep web, if s/he knows the appropriate deep web site that stores the sought information and is familiar with the search surface offered by the site. Deep web searchers aim at bridging this gap between the user and deep web sites. The information that resides on deep web site can be efficiently accessed if the structure of the databases is known by the searcher.

In the pilot phase of our project we intend to include only certain topics in the search space of our deep web searcher, namely, books, movies, restaurants, and football. We will contract with the owner of selected databases and incorporate the content of their database into the topics of our deep web search engine. The deep web search engine communicates with databases by SQL queries through a *mediator layer*. Through this layer database owners can provide the necessary and only the necessary information about their database, it insures feasibility of querying, controls authority rights, and assures authenticity and answering facilities.

B. Natural language querying

One of the bottlenecks of traditional search engines is that they keep keyword-based catalogued information about web pages; therefore they retrieve only the keywords from users' queries and display pages with matched ones. This matching method neglects important information granules of a natural language query, such as the focus of the query, the semantic information and linguistic connections between various terms, etc. Traditional searchers retrieve the same pages for "*When*

[1] http://www.brightplanet.com

does the president of the United States visit Russia" and "*Why does the president of Russia visit the United States*", though the requested information is quite different. These pieces of information could be very important particularly when deep web search is concerned, because e.g. the interrogative is not a keyword-like information (the ideal result does not contain it), but it specifies the focus of the query. Deep web searchers communicate with the deep web sites by accessing the residing database using a querying language (e.g. SQL). In the retrieval of the proper answer for the question, hence, the focus of the query should be encoded in the translated SQL query. The user expects different answer for the questions: "*When was J.F.K born*" or "*Where was J.F.K. born*".

In this paper we describe the operation of our system to process Hungarian language questions and to transform them to SQL queries. Although, natural language processing is always language-dependent but the non-language specific part of the operation, e.g. the structure of our system can be directly used for question processing in other languages. Moreover, our concept of processing natural language questions—to be outlined in the next—can be carried over to other languages with appropriate modifications on handling the important grammatical structures.

II. System description

We developed a natural language querying based deep web searcher that consists of five main modules depicted in Figure II.

The input of the natural language (NL) module is the user's question. The module specifies the morphological characteristics of each syntactically relevant unit of the question (see details in subsection II-A) and groups the related units in a bracketing phase. The output is a list of bracketed expressions. The context recognizer determines the context(s) of the question and it creates a list of so-called CL (Context Language – see the Appendix) expressions based on the schema and attribute names of the covered topics of the deep web searcher.

The deep web search engine stores the database (DB) info about partner deep sites. Based on the context of the query that is encoded in the CL expression(s), it determines databases where the CL expression should be sent. It also includes an SQL parser that translates CL expressions into local SQL queries. The word local here refers to the fact that the naming conventions of schemas and attributes in these SQL queries have local validity. Because each partner deep web site has different database structures, and applies different terminology, the searcher uses its own one and the domestication of SQL queries for the mediator layer.

The mediator layer transfers local SQL queries according to the naming convention and structure of the database of the deep site. The administrator of the database should fill in a form before the database is connected to the searcher; by this form the administrator maps the names of schemas and attributes of the deep web searcher to the local convention. Finally, the appropriately transferred SQL queries are passed to the database of the deep web site and the answer is returned in the form of URLs. The answer analyzer orders the URLs and displays them to the user.

A. The NL module

The input questions should fulfill several requirements. The system accepts only simple (only one tensed verb is allowed), well-formulated and -spelled interrogative sentences starting with a question word from a given list. It is not designed to answer questions focusing on causality ("*Why did the Allies win WW2?*"), intension ("*Would you like a cup of coffee?*"), non-factual ("*How am I?*") information. The system's task is to answer questions based on information stored in partner's databases.

The module applies several tools for its operation: morphological parser [3], various databases storing the required lexical information (proper names, interrogatives, lists of some significant words, patterns for various fixed terms, such as dates, URLs etc.).

1) Identification of SRUs: The input question is first passed to unit parser (UP) method. Its task is to determine the syntactically relevant units (SRU — can consist of several words) of the question and to label with the help of the morphological parser each SRU with its part of speech. One of the important goals of UP is to determine multi-word SRUs like names, titles (movie, book, etc.), institutions, proprietary and company names, etc. We call them together *individuums*. Because Hungarian is an agglutinative (morphemes are glued to words forming other words) and highly inflective language [4], [5], [6], [7] the determination of SRUs is a more complex task than simple pattern recognition, and therefore requires the support of the morphological parser. One of the characteristics of the morphological system of the Hungarian language is that there are many ambiguous word forms (same spelling, but different morphological parsing). Such ambiguous word forms (SRUs) are disambiguated in parsed alternatives.

In UP method five modules collaborate: the expression dismemberer (ED), the individuum detector (ID), the smart tag detector (STD), the abbreviation detector (AD), and the morphological parser (MORPH). ED decreases the size of its input expression by cutting off words from it. ID detects if its input expression is listed among individual SRUs. STD checks whether its input is of special form (date, URL, e-mail, etc.) or on the list of special terms. AD checks whether its input is an abbreviation with a given definition. The algorithm of UP is the following.

Input: full question represented as array of words.

Output: lists of sentence alternatives represented in a special data model. Elements of lists are SRUs labelled with their morphological information.

1) Begin ED(T); T is an array of words.
2) If length(T) > 0
3) T is passed to ID, STD and AD, it is processed simultaneously.
   a) Begin ID(T)
   b) T is compared to each entries of the individuum database.

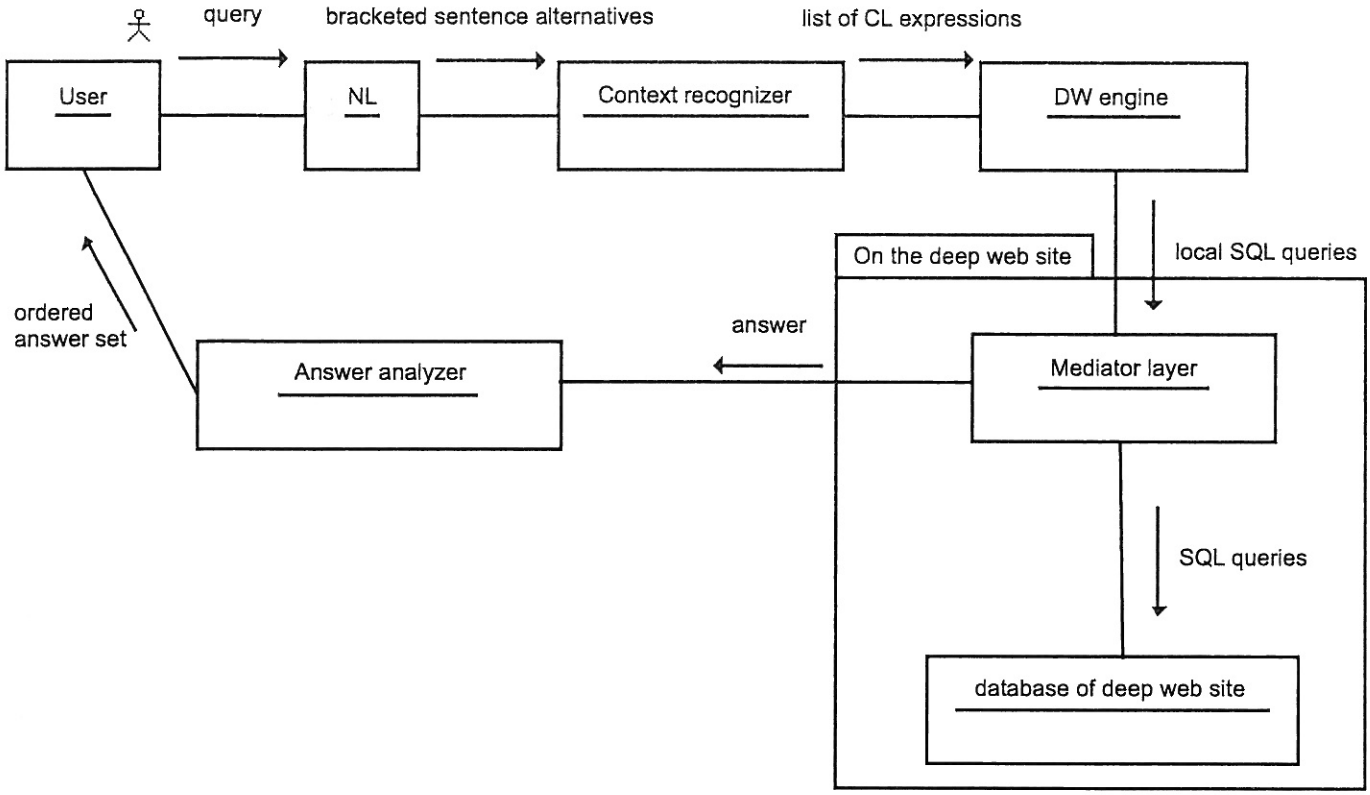

Fig. 1. The collaboration of main modules of the natural language querying based deep web searcher

c) If T is identical with any of the entries
   i) T is labelled as individuum with suffix *none*;
   ii) Remove the last word of T and call ID recursively.
d) If T is not identical with any of the entries.
   i) MORPH (T) is called;
   ii) If there is at least one suffix (S) on the last word of T
      A) Remove the suffix from T; $T := T - S$
      B) Administrate the removed suffix S.
   iii) If there is no suffix on the last word of T
      A) Remove the last word of T;
   iv) Calls ID recursively with the new T.
e) End ID

4) Remark: the operation of STD and AD is analog with ID, the difference is only in the compared entries.
5) If ID (or SDT or AD) recognizes an array of word, T as SRU, then ED continues from the word after the last word of T.
6) If none of ID, SDT and AD recognizes T, then (1) it consists of a single word, and therefore (2) MORPH assign it its part of speech and suffixes; It is labelled by this information and ED continues from the next word.
7) End ED

The recognition of multi-word SRUs is performed based on decreasing the size of the expression. If an expression is not found in any of the SRU lists the last suffix (if exists) or the last word (if no suffix) is removed from the expression, and ED is continued recursively. The loop terminates when the expression contains only a single word. When a multi-word SRU is recognized by any of ID, STD, or AD, the last word of the recognized SRU is removed and remaining expression is passed to the three detectors again (Step 3c-ii). This is because an SRU may have internal SRUs. In this way we create different alternative interpretations of the original questions. The number of the alternative interpretation can be increased if the morphological parser or the detector methods assigns several solutions to an expression.

Let us illustrate the above algorithm by means of an example. *Ki rendezte Az én szép kis mosodámat?* (Who directed My beautiful laundrette). The title of the movie in Hungarian is „*Az én szép kis mosodám*", where the stem of the last word (*mosoda*=laundry) has a possessive suffix (-m). This word (mosodám) further receives the accusative suffix (-(a)t). The two suffixes have to be removed from the word, in order to be able to recognize the title of the movie correctly.

1) ED start with the full question.
2) Because there are no special SRU starting with the first (*Ki*), and the second (*rendezte*) words, they are labelled by the output of the morphological parser: *Ki* (affix; interrogative: two alternatives), *rendezte* (verb, past tense + suffix).

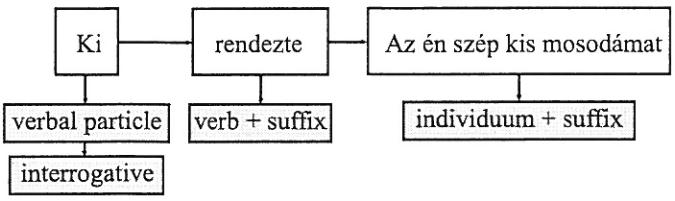

Fig. 2. The result of UP of the question: *Ki rendezte Az én szép kis mosodámat?*

3) T = *Az én szép kis mosodámat*. None of the three detectors recognizes T. Therefore it is passed to MORPH.
4) MORPH removes the accusative suffix (-(a)t) from T, and the control return to ID (SDT, AD).
5) ID recognizes the expression *Az én szép kis mosodám*.
6) No other special SRU starts with the remaining part of the expression, therefore no other alternative is generated.
7) Because no word remains unprocessed in the input question, ED finishes its operation.

The result of UP is depicted on Figure II-A.1

2) Bracketing phase: The second part of the NL module groups related SRUs in brackets. The module has several submodules that are connected sequentially in the following order:

1) recognizer of adverbs and participles (RAP);
2) recognizer of adjective groups (RAG);
3) recognizer of conjunctions (logical operators) (RC);
4) recognizer of possessive structures (RPS);
5) recognizer of postposition structures (RPPS).

Each sub-module works based on the morphological information (stem [part of speech], suffixes) that is determined by the morphological parser and that each SRU is annotated with. Note that special SRUs that are recognized by any of ID, SDT or AD are labelled by their recognized type (e.g. individuum, type of tag, abbreviation).

We define the *bracketing level* for an SRU, which is the depth of embeddedness, i.e. the number of brackets that is around the given SRU. E.g. $[A\ [B\ [C\ D]\ [E]]]\ F$, $\mathrm{bl}_A = 0$, $\mathrm{bl}_{[C\ D]} = 1$, and $\mathrm{bl}_{[B\ [C\ D]\ [E]]} = 2$, etc.

1. Adverbs and participles are removed from input questions, because they typically do not represent such information that is stored in databases.

2. In Hungarian there are two main types of adjective group:

1) adjectives that are inflected from a noun by suffixes: -ú, or -ű (e.g. *célú, című*). The structure of the adjective group is: [[[noun phase] adjective] noun], where the noun phase is typically an individuum, and the adjective expresses a property of the noun that is defined by the noun phase. E.g.: *Mátrix című film* (movie entitled Matrix); here the noun *film* (movie) is specified by its title property, and the value of the title is Matrix.
2) Other adjective groups have the structure: [adjective noun]. The adjective can be arbitrary but with -ú, -ű suffix.

3. We process numerous conjunctions that have the meaning *and, or*. The point of the recognizer method is that it checks the two neighboring SRUs around a conjunction and forms a group from the three SRUs if their part of speech and suffixes are identical. Namely, there are two distinct cases:

1) If either of the neighboring SRUs are already in a bracketed group then we simply form a new group and label with the morphological characters of the second SRU. For example:

$$\textit{könyvet}_{\text{noun; ACC}}\ \textit{vagy}_{\text{conj}}\ \textit{filmet}_{\text{noun; ACC}}$$

(book or movie); they form a conjunctive group: [könyvet vagy filmet]$_{\text{noun; ACC}}$
2) If any of the neighboring SRUs are already in a bracketed group then we take SRU with the lower bracketing level, and compares its morphological characters with the ones of the first SRU at the same level in the other bracketed grouped. If they are identical a conjunctive group is created. Example: *piros és [fehér zászló]* (red and white flag) where the last two words are grouped by RAG; $\mathrm{bl}_{\text{piros}} = 0$; the corresponding SRU to be compared with is *fehér*. They share morphological characters (adjectives) so a conjunctive group is created as: *[[piros és fehér] zászló]*. Analogously, *[április 13.] és [24.]* (April 13th and 24th) is grouped as *[április [13. és 24.]]*

4. RPS works based on the possessive suffixes of SRUs. The method linearizes possessive structure in order to have equally the structure

$$[_1 \ldots [_n \text{possessor property}]_n \ldots \text{property}]_1$$

i.e. subsequent possessive structures are embedded in the above way. In Hungarian there are several ways to express the possessive relation between words, and the order of the structure is not fixed.

- simple: *a könyv szerzője* (the author of the book); the possessor (*könyv*) gets zero suffix and the property (*szerző*) has possessive suffix.
- indicated: *a filmnek a rendezője* (director of the movie); the possessor gets (-nak/-nek) suffix and a definite article precedes the suffixed property (*rendező*). This is always the case when the order of the words is opposite, e.g.: *Ki **a rendezője a** Mátrix című **filmnek?*** (Who is the director of the movie entitled Matrix?)
- multiple: *a producer filmje rendezőjének a felesége* (the wife of the director of the movie's producer). Only the *last* word (*rendező*) that is both in the possessor and in the property role receives both suffixes (-je+-nak/nek), otherwise the suffix of the possessor is not indicated.

Based on the above listed possessive morphological characters of SRUs in the question the RPS method specifies the linearized structure of the possessive structures.

5. RPPS has the simple task to group postposition with the preceding noun. Examples: *1997 óta* (since 1997) is grouped

as *[1997 óta]*; *[április [13. és 15.]] között* (between 13th and 15th of April) becomes *[[április [13. és 15.]] között]*.

The last four sub-modules of bracketing phase is performed iteratively until new grouping can be done in a cycle.

B. The context recognizer

The context recognizer gets bracketed sentence alternatives as input. It determines the context of the query and creates a list of CL expressions (see the Appendix) based on the schema and attribute names of the covered topics of the deep web searcher. The context is determined based on the following information (not detailed here):

- schema and attribute names occurring in the question;
- the type of individuums occurring in the question;
- the interrogative of the question;
- the verb and its complements in the question (as the query should be simple, only one tensed verb is allowed);
- superlatives representing extreme values with respect to attributes;
- a matching algorithm that compares the results of the above-listed sources, finds the possible non-ambiguous interpretations, and translates the question into CL expression(s).

We illustrate the result of the context recognizer through an example:

Mikor lesz a legközelebbi Arsenal-MU meccs?

(When will be the next Arsenal-MU match?) generate the following CL expressions:

```
Context = Match  | Context = Match
Date = ?         | Date = ?
Date             | Date >= today; MIN
team1 = Arsenal  | team1 = MU
team2 = MU       | team2 = Arsenal
```

The question generates two CL expressions because based on the verb and its complements (*van* + zero suffix; *lesz* is the future tense form of *van*) we cannot decide which football team has to be substituted to attributes `team1` and `team2`. This ambiguity is resolved when the complements of the verb clarify the roles of the two teams:

Mikor játszik legközelebb az Arsenal az MU-val?

(When will Arsenal play next with MU?), when a single CL-expression is generated as:

```
Context = Match
Date = ?; >= today; MIN
team1 = Arsenal
team2 = MU
```

Here we use that the structure *játszik* + zero suffix + *-val/vel* suffix (sy/sg play with sy/sg) uniquely determines the role of the two teams.

III. Conclusions

In this paper we presented some results of the "In the net of words" (NoW) project that aimed at creating a complex search interface that incorporates deep web search, Hungarian natural language question processing, image search support by visual thesaurus. The paper was focused on the processing of Hungarian questions and introduced a few algorithms for determining syntactically relevant units (SRUs) with their morphological characters, and for determining related groups of SRUs. The set-up of the system can be readily applied for deep web search applications on other languages. The concept of the described language processing method can be carried over to other languages with appropriate modifications on handling grammatical structures.

IV. Acknowledgements

This research was supported by NKFP 0019/2002.

Appendix

Here we give the definition of CL (Context Language) in BNF.

```
<sentence> ::== ( <context>; <logical expr.> )
  | <sentence> <sentence op.> <sentence>
<sentence op.> ::== UNION | MINUS | INTERSECT
<context> ::= Context: <text constant>
<logical expr.> ::== <attribute name>: <value expr.>
  | <logical expr.>; <logical expr.>
<attribute name> ::== <text constant>
<value expr.> ::== <return expr.> | <logical cond.>
<logical cond.> ::== MIN | MAX
  | COUNT <logical op.> <numeral constant>
  | <logical op.><logical value>
  | <logical op.><attribute name> <sentence>
  | <logical cond.>, <logical cond.>
<return expr.> ::== ? | COUNT ?
<logical op.> ::== = | != | < | > | LIKE | NOT LIKE
<logical value> ::== <text constant>
  | <numeral constant> | <date constant>
<text constant> ::== <string>
<numeral constant> ::== <real number>
  | <natural number>
<date constant> ::== [ <date> ] | NOW
  | <date> + <natural number>
```

References

[1] M. K. Bergman, "The deep web: surfacing hidden value," *Journal of Electronic Publishing*, vol. 7, no. 1, August 2001, http://www.press.umich.edu/jep/07-01/bergman.html.

[2] H. Winkler, "Suchmaschinen. metamedien im internet?" in *Virtualisierung des Sozialen*, B. Becker and M. Paetau, Eds., Frankfurt/NY, 1997, pp. 185–202, (In German; English translation: http://www.uni-paderborn.de/~timwinkler/suchm_e.html).

[3] L. Németh, P. Halácsy, A. Kornai, A. Rung, and I. Szakadát, "Leveraging the open source ispell codebase for minority language analysis," 2004, to appear in Proc. of SALTMIL 2004; http://www.szoszablya.hu/.

[4] J. Tompa, Ed., *The System of the Modern Hungarian. Descriptive Grammar.*, 2nd ed. Budapest: Akadémiai Kiadó, 1970, (In Hungarian; original title: A mai magyar nyelv rendszere. Leíró nyelvtan.).

[5] K. É. Kiss, F. Kiefer, and P. Siptár, *New Hungarian Grammar*, 2nd ed. Budapest: Osiris, 1999, (In Hungarian; original title: Új magyar nyelvtan.).

[6] F. Kiefer, Ed., *Structural Hungarian Grammar. Syntax.* Budapest: Akadémiai Kiadó, 1992, (In Hungarian; original title: Strukturális magyar nyelvtan. Mondattan.).

[7] ——, *Structural Hungarian Grammar. Morphology.* Budapest: Akadémiai Kiadó, 2000, (In Hungarian; original title: Strukturális magyar nyelvtan. Morfológia.).