WOW: the Hungarian Deep Web Searcher *

Domonkos Tikk†, Zsolt T. Kardkovács, Gábor Magyar

Department of Telecommunication & Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
{tikk,kardkovacs,magyar}@tmit.bme.hu

Abstract

This paper summarizes the goals and presents the results of our ongoing research and development project, called "In the Web of Words" (WOW), funded by the National R+D Program in Hungary. The project aims at creating a complex search interface that incorporates — beside the usual keyword-based search functionality—deep web search, Hungarian natural language (NL) question processing, image search support by visual thesaurus. In this paper we focus on system architecture and NL processing. One of the most crucial part of the system is the transformation of NL 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: NL question processing, context recognition, and SQL transformation.

1 Introduction

The importance of information gathering on the Web and the central and unquestioned role of search engines make them an obvious focus of investigation. However searching on the Internet today can be compared to dragging a net across the surface of the ocean. While a great deal may be caught in the net, there is still a wealth of information that is deep, and therefore, missed. Most of the information accessible through the net is organized and stored in structured databases, and standard search engines never find it. Internet content is considerably more diverse and the volume certainly much larger than commonly understood.

Traditional search engines create their indices by crawling surface Web pages. Traditional search engines can not "see" or retrieve content in the deep Web – those pages do not exist until they are created dynamically as the result of a specific search. Because traditional search engine crawlers can not probe beneath the surface, the deep Web has been hidden.

In 1994, Dr. Jill Ellsworth first used the phrase "invisible Web" to refer to information content that was "invisible" to conventional search engines. Invisible Deep-Web is the part of the Web that can not easily be crawled by traditional search engines.

*This research was supported by NKFP 0019/2002.
†Corresponding author

We use the terminology “deep Web” to the information resources managed in structured (namely SQL) databases, The deep Web is qualitatively different from the surface Web. Deep Web sources store their content in searchable databases that only produce results dynamically in response to a direct request. But a direct query is a “one at a time” laborious way to search. Our search engine aims to use natural language input to produce a series of formal queries.

How does information appear and get presented on the Web? In the earliest days of the Web, there were relatively few documents and sites. It was a manageable task to post all documents as static pages. Because all pages were persistent and constantly available, they could be crawled easily by conventional search engines.

Sites that were required to manage tens to hundreds of documents could easily do so by posting fixed HTML pages within a static directory structure. In the mid 90s database technology was introduced to the Internet and since that a true database orientation is becoming more and more relevant, especially for larger sites. It is now accepted practice that large data producers choose the Web as their preferred medium for commerce and information transfer. What has not been broadly appreciated, however, is that the means by which these entities provide their information is no longer through static pages but through database-driven designs.

It has been said that what cannot be seen cannot be defined, and what is not defined cannot be understood. Such has been the case with the importance of databases to the information content of the Web. And such has been the case with a lack of appreciation for how the older model of crawling static Web pages — today’s paradigm for conventional search engines — no longer applies to the information content of the Internet.

Serious information seekers can no longer avoid the importance or quality of deep Web information. Directed query technology is the only means to integrate deep and surface Web information.

2 Project goals

In WOW 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 the visualized system architecture and secondarily—due to the lack of space—it presents briefly some solutions achieved so far in the area of NL processing, context recognition and SQL parsing [9, 10]. Before going into details we give a short overview about the project’s aims.

2.1 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.

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

Result pages are posted as dynamic web pages being answers of direct queries. Incorporating deep web access in internet search engines is a very important issue. Studies about the internet [4] show 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 [11]. 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.

Form based or other graphical interfaces are the best choices to retrieve information from a single database or to query web sites on a concrete topic. On the one hand, there are differences in structural design and in granularity of data between databases in a multi-database environment. That is why a form based search application needs to semantically restructure user input according to the target databases or it needs to reduce differences in a way. Both are hard tasks if the number of databases are not strongly limited, or if there is no agreement between the site owners. On the other hand, the number of attributes to be queried are not bounded in search engines. Without a strict limitation on the number of the topics, form based applications become unusable or impractical. In addition, natural questions (see examples in the main part) can be displayed in rare or in vague manner. For a more detailed analysis on the differences between NLIDBs, keyword and menu based search engines, see [1].

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 owners of selected databases and incorporate topics of their

databases into scope of WOW. WOW's 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.

2.2 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 NL 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 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. Solutions that do not deal with the given morphosyntax of the sentence, e.g. Askjeeves [3], AnswerBus [2], Ionaut [7], also suffer from this problem.[2]

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; moreover, interrogatives are usually not used in search engines deemed to be useless word w.r.t. the given 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 NL processing is always language-dependent but the non-language specific part of the operation, i.e. the structure of our system can be directly used for question processing in other languages.

3 System description

We developed a NL querying based deep web searcher that consists of five main modules depicted in Figure 1.

The input of NL module is the user's question. The module specifies the morphological characteristics of each syntactically relevant unit of the question and groups the related units in a bracketing phase. Obviously, NL module requires various knowledge bases such as dictionaries, and ontology. The output is a list of labelled and bracketed expressions. The context identifier determines the context(s) of the question and it creates a list of so-called CL (Context Language) expressions based on the schema and attribute names of the covered topics of the deep web searcher.

[2] To test whether the solution is keyword based or not (even if the input is a NL question) try: "Who is the present King of France?" or "When visited the president of Russia the United States?"

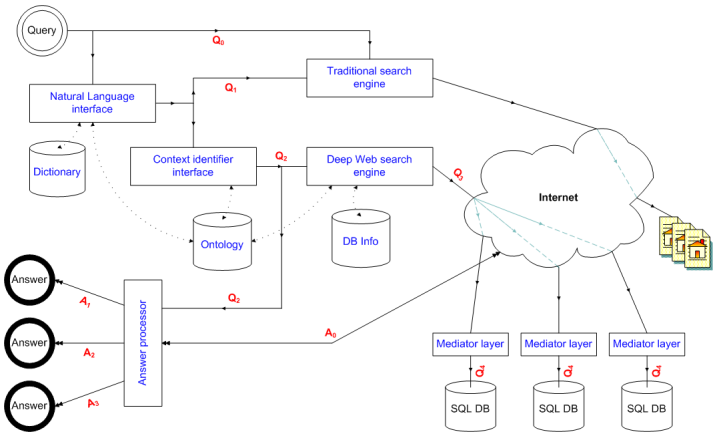

Figure 1: The collaboration of main modules of the NL querying based deep web searcher

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 CL expression(s), it determines databases where 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.

3.1 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 [6], various databases storing the required lexical information (name of entities, interrogatives, patterns for various fixed terms, such as dates, URLs etc.).

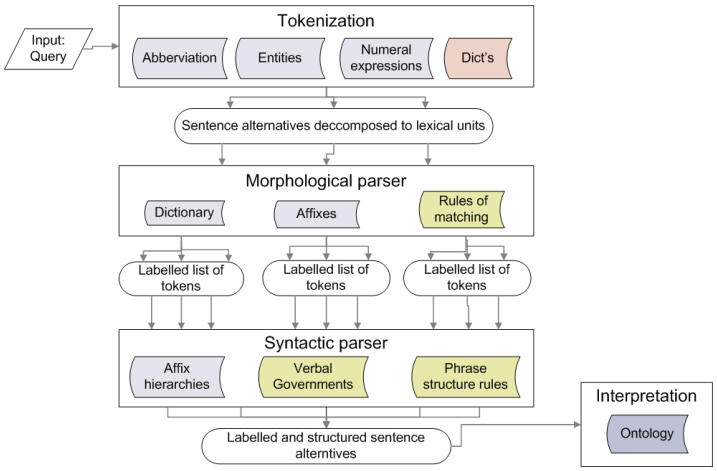

Figure 2: Main steps and means of NL module

Figure 2 depicts main steps and the means of NL module.

NL module consists of two main steps, the tokenization and the syntactic parser. The former determines tokens of the sentence, and annotate them with morfo-syntactic information, the latter identifies syntactic structures in the sentence, and brackets the coherent tokens.

3.1.1 Tokenization

The input question is first passed to tokenizer method. Its task is to determine the syntactically relevant units—termed *tokens* that can consist of several words—of the question and to label with the help of the morphological parser each token with its morphological characteristics (part of speech and affixes). One of the important goals of tokenizer is to determine multi-word tokens like names, titles (movie, book, etc.), institutions, proprietary and company names, etc. We call them together *entities*. Because Hungarian is an agglutinative (morphemes are glued to words forming another words) and highly inflective language [5, 8] the determination of tokens 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 tokens, e.g *ég* [verb] (burn) and *ég* [noun] (sky), are disambiguated in parsed alternatives.

In the tokenizer five modules collaborate: the expression dismemberer (ED), entity detector (ENT), 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. ENT detects if its input expression is listed in among entities. 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 recognition of multi-word tokens is performed based on decreasing size of expression. If an expression is not found in any of token 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 token is recognized by any of ENT, STD, or AD, the last word of the recognized token is removed and remaining expression is passed to the three detectors again. This is because a token may include internal tokens. By 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. The detailed algorithm of the tokenizer can be found in [9].

3.1.2 Syntactic parser

The syntactic parser, the second part of NL module groups related tokens in brackets. The module has several sub-modules that are connected sequentially in the following order (for more detailed discussion see [9, 10]: 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 is each token is annotated with. Note that special tokens that are recognized by any of ENT, SDT or AD are labelled by their recognized type (e.g. individuum, type of tag, abbreviation).

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:

- adjectives that 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.
- 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 tokens around a conjunction and forms a group from the three tokens if their part of speech and suffixes are identical.

4. RPS works based on the genitival and possessive suffixes of tokens. It linearizes genitival structures in the order

$$[_1 \ldots [_n \text{genitival possessed}]_n \ldots \text{possessed}]_1,$$

i.e. subsequent genitival structures are embedded. 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+ɔnak/nek), otherwise the suffix of the possessor is not indicated.

Based on the above listed possessive morphological characters of tokens 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.

3.2 The Context Identifier

The context identifier (CI) gets bracketed sentence alternatives as input. CI determines the context of the query and creates a list of CL expressions based on the schema and attribute names of the covered topics of the deep web searcher. The context is determined in the following steps (see Figure 3, not detailed here):

- Recognition of fixed elements occurring in the questions; schema and attribute names represent type and property information, respectively. These features determine the segment of the information space the question refers to.
- Procession of linguistic structures recognized in NL module, as depicted on Figure 3.
- Identification of question's focus and roles of entities and verbs. This is based on the interrogative of the question, the type of entities occurring in the question, and the verb and its complements in the question (as the query should be simple, only one tensed verb is allowed), respectively.
- A filtering algorithm that matches the results of previous steps with stored context information, finds the possible non-ambiguous interpretations, and translates the question into CL expression(s).

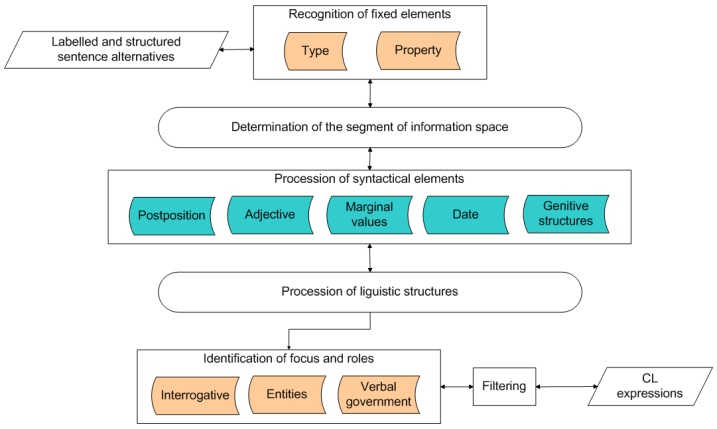

Figure 3: Main steps of CI's algorithm

3.3 Example

Let us illustrate the steps of question processing on the sentence: "*Mikor játsszák a Mátrixot Budapesten?*" (When Matrix is played in Budapest). Only those steps are indicated that are effective related to the example.

1. NL module; ENT (entity identifier) finds tokens *Mátrix* and *Budapest* in the appropriate entity dictionary.
2. NL module; morphological parsing; each word (token) is annotated with its morphological characteristics, e.g. *játsszák* = *játszik* (play) [verb] + present, indicative, plural, 3, or *játszik* [verb] + present, imperative, plural, 3; *Mátrixot* = *Mátrix* [entity] + accusative; *Budapesten* = *Budapest* [entity] + superessive.

   Remark: due to word *játsszák* has two morphological parsings, two sentence alternatives are generated.
3. NL module; results of bracketing: (Mikor) (játsszák) (a Mátrixot) (Budapesten).
4. CI module; determination of entities' role: identification of *Mátrix* as movie and *Budapest* as city
5. CI module; focus determination; interrogative *Mikor* relates to date or time or moment.
6. CI module; procession of verbal government; matching of structure *játszik* + object + locativus, and determination of governments role: object = movie, function, role, . . . ; locativus = city, country, ...
7. CI module; filtering and generation of CL expression:

```
Context = Event
Time    = ?
Event   = ( Context  = Program
            Title    = Matrix
            Location = ( Context = Cinema
                         City    = Budapest ))
```

4 DW Searcher and connection with content providers

The deep web search engine of WOW attends to the followings:

1. Relevance identification: determines the sources based on the context of CL expression and the valid property fields, which are theoretically capable of answering the question. It uses the Relevance Table from DB Info.
2. DWL conversion: converts CL expression to DWL query that is a special SQL dialect.
3. Procession of standard units: unifies the form of data elements that are stored different ways in databases (names, dates, etc.)
4. Authentication, identification: controls communication with partner databases, such as authentication of the database, question–answer pairing.

The deep web searcher communicates with the deeps web sites through the mediator layer placed on the site of deep web content provider partners. DWL queries are also transferred to databases via this layer.

The tasks of the mediator layer are the joining of deep web site to the deep web searcher, authorization of queries, authentication, and assurance of answering. Its competence is restricted exclusively to tasks connected with querying; it neither stores data, nor starts processes.

The mediator layer is initialized by the administrator of DW site when the DW content provider is connected to WOW. Then the administrator selects topics of his/her site from a list, and determines the part of the database to be shared publicly via the deep web search engine. These data are registered in DB Info and used by the relevance identifier when selecting databases for question answering. DB Info can be updated on an administration interface.

The data transferred to the mediator layer consists of three parts: question ID, DWL query and authentication elements. After checking the authentication information of WOW and DW site, the mediator layer substitutes in DWL query the schema and attribute names to be in accordance to the naming convention of the given DW site. These information are provided by the DW site administrator when the mediator layer is initialized.

The answering is implemented analogously. The answer must contain the question ID, number of valid answers — if it does exceed the limit, the answers themselves, and finally the authentication elements.

The answer processor module collects and orders the incoming answers from different sources. Based upon our former investigations, data obtained from different sources are heterogenous: either records, set of records, or only a URL. Therefore, WOW returns to the user a page containing links to the answer pages of deep web sites. DW searcher is not competent in judging the soundness of DW sites' answers, therefore the answers are only grouped and ordered w.r.t. the following aspects

- If the original question is ambiguous, the answers are grouped by the different interpretations of the questions;
- based on source, and based on schema within a source;
- based on time-stamp of incoming answers;
- based on use profile;
- based on the evaluation of user habits; this can be applied to set up a source–context relevance matrix.

The actual ordering method can be chosen by the user when customizing the user interface of WOW.

5 Conclusions

In this paper we presented some results of WOW 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 system architecture and NL processing.

References

[1] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases – an introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995.

[2] `http://www.answerbus.com/`.

[3] `http://www.askjeeves.com/`.

[4] M. K. Bergman. The deep web: surfacing hidden value. *Journal of Electronic Publishing*, 7(1), August 2001. http://www.press.umich.edu/jep/07-01/bergman.html.

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

[6] Hunmorph, 2004. http://www.szoszablya.hu/termekek/hunmorph/.

[7] `http://www.ionaut.com:8400/`.

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

[9] D. Tikk, Zs. T. Kardkovács, Z. Andriska, G. Magyar, A. Babarczy, and I. Szakadát. Natural language question processing for hungarian deep web searcher. In *Proc. of the IEEE International Conference on Computational Cybernetics (ICCC'04)*, page 5 p., Vienna, Austria, 2004.

[10] D. Tikk, Zs. T. Kardkovács, G. Magyar, A. Babarczy, and I. Szakadát. Natural language module of a hungarian deep web searcher. In *Proc. of the 4th IEEE Int. Conf. on Intelligent Systems Design and Applications (ISDA'04)*, pages 73–77, Budapest, Hungary, August 2004.

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