

Entity Recognizer in Hungarian Question Processing*

Domonkos Tikk¹, P. Ferenc Szidarovszky²,
Zsolt T. Kardkovács¹, and Gábor Magyar¹

¹ Dept. of Telecom. & Media Informatics,
Budapest University of Technology and Economics,
H-1117 Budapest, Magyar Tudósok krt 2, Hungary
{`tikk`, `kardkovacs`, `magyar`}@tmit.bme.hu
² Szidarovszky Ltd., H-1392 Budapest, POB 283, Hungary
`ferenc.szidarovszky@szidarovszky.com`

Abstract. In our ongoing research and development project, called “In the Web of Words” (WoW), funded by the National R+D Program in Hungary, we aim to create a complex search interface that incorporates—beside the usual keyword-based search functionality—(1) deep web search, (2) Hungarian natural language question processing, (3) image search support by visual thesaurus. This paper focuses on a particular and crucial part of the question processing problem (2): recognition of entities. Entities are expressions that have fixed form, and that are assigned context specific information in the dictionary. Due to the agglutinative feature of Hungarian language they often appear in the text differently as in the dictionary, therefore their detection requires special algorithms at processing.

1 Introduction

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 (NL) 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 a particular problem of question processing task: the entity recognition. Before going into details we give a short overview about the project’s aims.

1.1 The Deep Web

The deep web (DW) is content that resides in searchable and online accessible databases, whose results can only be discovered by a direct query¹. Without the directed query, the database does not publish the result. Result pages are

* This research was supported by NKFP 0019/2002.

¹ <http://www.brightplanet.com>

posted as dynamic web pages as answer to direct queries. Incorporating DW 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 DW is about 400 times larger than that of the surface web that is accessible to traditional keyword-based search engines; (2) the size of DW is growing much faster than the surface web; (3) DW is the category where new information appears the fastest on the web; (4) DW sites tend to be narrower and deeper than conventional surface web sites; (5) DW 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 DW 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.

Initially, we intend to restrict the search space of our deep web searcher (DWS) to the following topics: books, movies, restaurants, and football. DWS accesses information in these topics of contracted databases. It communicates with databases by SQL queries through a mediator layer, which enables database owners to provide the necessary and only the necessary information about their database. Beside that the mediator layer insures feasibility of querying, controls

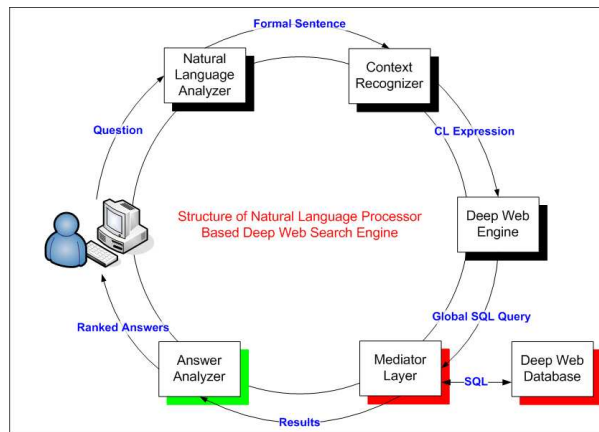


Fig. 1. Structure of our DWS enabling NL querying

authority rights, and assures authenticity and answering facilities. The structure of the system is depicted on Figure 1.

1.2 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 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 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 [4], Ionaut [5], also suffer from this problem.² These pieces of information could be very important when DW 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. DWSs communicate with the DW 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.

There has been a lot of work in recent years on both natural language processing and web-based queries. A non-exhaustive list of projects: Practice[6], START[7,8], NL for Cindi[9], Masque/SQL[10], Chat-80[11] or Team[12] provide solutions for English, Spanish NLQ[13], Sylvia-NLQ[14] for Spanish, Phoenix[15] for Swedish, Edite[16], LIL/SQL[17] for Portuguese, NChiq[18] for Chinese and KID[19] for Korean languages. A bilingual (Danish and Italian) question answering system with ontology harmonization is reported about in [20]. Although several ideas and techniques has been adapted from these systems to WoW, none of them can be applied directly to our purpose, both because of the peculiarity of DW search and that of the language.

In order to alleviate the shortcomings of keyword based search, in our DWS we enable the user to formulate his/her information need by NL question. Processing of arbitrary NL question is generally an extremely complicated problem for all languages and Hungarian is not an exception. In fact, due to the agglutinative nature of the language even dictionary based search algorithms have to be equipped with intelligent functionality because, e.g.,

1. the stem of words can change;
2. additional suffixes can modify the form of previous suffixes.

In this paper we deal with the natural language module of our DWS with a special focus on the particular important issue of entity recognition. First, a brief

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 did the president of Russia visit the United States?"

description is given about the NL module, then the entity recognition problem is treated in details.

Here we remark that, though, NL processing is always language-dependent but the non-language specific part of the operation, e.g. the structure of our system can directly be adapted for question processing in other languages.

1.3 Description of NL Module

We apply several restrictions on input questions of the system, which are originated from different reasons. First, DWS attempts to answer user queries on the basis of the content of a collection of topic-specific DW sites. The information stored at DW sites are typically factual ones, therefore we do not allow questions focusing on casuality (“*Why did the Allies win WW2?*”), intension (“*Would you like a cup of coffee?*”), subjective (“*How am I?*”) or other type of non-factual information. Second, there are grammatical limitations on input question in order to make NL question processing feasible: 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.

A basic tool that helps the operation of the module is the HunMorph morphological parser (MP) [21].

1.4 Definition of Entity

The knowledge base of the NL module contains various dictionaries storing the lexical information of special tokens (proper names, interrogatives, lists of some significant words, dates, URLs etc.). These tokens are called together *entities*. Our approach is not a (supervised) learning method that is trained to recognize unknown entities in free text. Its purpose is to recognize *known entities* from a collected data base in free text even if they are in inflected form. These entities are stored in a local data base and are extracted from databases of our contracted content providers.

Entities can be single or multi-word expressions and are of two basic types:

1. dictionary based: the entity has a fixed form that is stored in the dictionary (e.g. proper names, interrogatives, special words). Such entities are inserted to the dictionary on the basis of the content of partner DW sites. At insertion context information can also be linked to entities.
2. pattern based: only the possible patterns of the entity is given (e.g. URL, date, e-mail address). A candidate text is matched against the pattern when checking identity. A pattern consists of a set of simple rules, which is given for each pattern manually.

1.5 Operation of NL Module

NL module consists of two submodules: the tokenizer and the bracketing module. The input question is first passed to the *tokenizer*. Its first task is to identify tokens (syntactically relevant units) of the sentence. Tokens are one-word

or multi-word expressions whose internal structures are irrelevant for the syntactic parsing of the sentence. Multi-word tokens are entities, such as personal names, institution, proprietary and company names, titles, addresses, etc. As Hungarian is a highly agglutinative language where major semantic/syntactic relations are signalled by a series of “stackable” suffixes, the identification of entities is a more complex task than simple pattern recognition and requires the support of a morphological parser [21]. It is described in detail in Section 2. MP assigns *part of speech* labels to tokens and provides their morphological analysis. This information is the basis of the subsequent bracketing phase. One of the characteristics of the morphological system of the Hungarian language is that many morphologically complex word forms are ambiguous in terms of their morphological structure. Such ambiguous tokens are disambiguated in parsed alternatives.

The bracketing module groups related tokens in brackets. The module has several submodules for recognizing: (1) adverbs and participles; (2) adjective groups; (3) conjunctions (logical operators); (4) genitive and; (5) postposition structures. Submodules use the morphological annotation of tokens: stem, part of speech, suffixes (entities are labelled by their type as “part of speech”). The operation of bracketing is not detailed in this paper, for further details see [22].

2 The Entity Recognizer

The entire algorithm of tokenizer relies on entity recognition. The entity recognizer (ER) has two main tasks:

- *searching*: determining the entities in the sentence;
- *annotating*: specifying the morphological characters of the entity.

The searching and annotating tasks are usually connected and cannot be performed separately.

In what follows we describe the algorithm of ER for fixed form *dictionary based* entities (see Subsection 1.4). The recognition of the pattern based entities is performed by Smart Tag Detector (STD), and is not detailed here.

Because an entity can consist of several words, theoretically all segment of an input question is a possible candidate. The number of segments in a sentence of size n is $n(n + 1)/2$. The average size of an input sentence is 7–10 words, while the size of the dictionary containing entities can be of order 10^6 . Therefore it is much more efficient to search on the basis of sentence segments than on the basis of dictionary entries. The search of an expression in the dictionary can be optimized by organizing dictionary entries into hash table. Segments of a sentence are checked against the dictionary entries by decreasing size.

Another problem of entity recognition is that an entity can be a part of another one (see e.g. *The New York Times* is a newspaper). While in [23], the Blitz NL processor selects a unique entity among the recognized ones based on confidence values, we intend to recognize all entities of a sentence, and generate different parsed sentence alternatives. Consequently, regardless from the success of a search each subsegment is checked again for shorter entities.

The order of segment checks are the following:

1. We start with the entire sentence: $[1, \dots, n]$.
2. Each subsegment of the sentence is examined by decreasing size: $[1, \dots, j]$, where $j = n - 1, \dots, 1$.
3. Segments starting with the second word are examined: $[2, \dots, j]$, where $j = n, \dots, 2$.
4. Systematically all segments are examined first ordered by the index of the first word and then by the length of the segment: $[i, \dots, j]$, where $j = n, \dots, i$, $i = 3, \dots, n$.

Remark 1. It is obvious that not all the $n(n + 1)/2$ segments are valid entity candidate. Considering that the first word of the sentence must be an interrogative, one may start from segment $[2, \dots, n]$ and thus reducing the number of segments to $n(n - 1)/2$.

In what follows we describe the entity recognition of a particular segment of the input question, called *candidate*. In Hungarian, the stem of a word can be modified when adding certain suffixes to it. In most cases only the last two letters of the word stem may change (*tűz* \rightarrow *tűzet*³; *álmom* \rightarrow *álmom*⁴). Also, the previous suffix can be changed when an additional suffix is put at the end of a word (this case only happens when the entity itself is of inflected form that is further declined in the sentence: *Vissza a jövőbe* \rightarrow *Vissza a jövőbét*⁵). In this case only the last letter may change. These considerations are reflected in the searching phase of ER.

A significant part of the entities are not Hungarian expression, and hence MP is not able to parse them. Nevertheless, when processing such an entity, the ER has to annotate them with morphological characters. We assume that for each entity a substitution word is given that is a morphologically parsable word having the same inflection as the last word of the entity. It is often the last word of the entity (if it is a nominative Hungarian word), or is generated algorithmically when inserting the entity entry into the dictionary. The substitution word plays role in the determination of the entity's suffixes.

We will use the following denotation:

- $\text{last}(x)$ denotes the last word of an expression x .
- $\text{length}(x)$ is the size of the word x in characters.
- $\text{trunc}(x, i)$ the word x truncated by i characters from the end.
- $\text{lchar}(x)$ the last character of x .

Further we use abbreviation C (candidate), S (substitution word) and E (entity). The flowchart of the algorithm is depicted on Figure 2.

1. If $\text{last}(E)$ is variable in form (i.e. can be inflected), nominative, Hungarian word (i.e. recognized by MP)

³ fire [NOM] \rightarrow fire [ACC]

⁴ dream [NOM] \rightarrow dream [ACC]

⁵ Back to the future [NOM] \rightarrow [ACC]

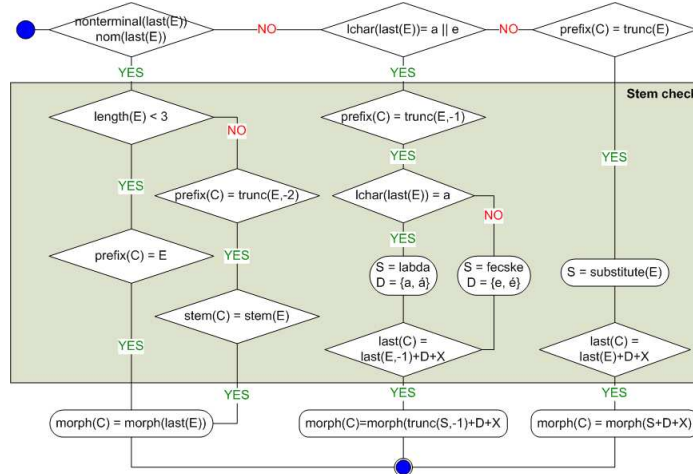


Fig. 2. Flowchart of the algorithm of entity recognizer

- 1.1 searching
 - 1.1.a If $\text{length}(\text{last}(E)) \geq 3$ then it is checked whether $\text{trunc}(E, 2)$ is a prefix of C .
 - 1.1.b If $\text{length}(\text{last}(E)) < 3$ then it is checked whether E is a prefix of C .
- 1.2 stem check: If 1.1.a is successful, that is $\text{trunc}(E, 2)$ is a prefix of C , then it should be examined whether the stem of $\text{last}(C)$ and $\text{last}(E)$ are identical. The reason of this is that after truncation there may be several words that match the prefix of $\text{last}(E)$. This step can be skipped when 1.1.b is valid.
- 1.3 annotation: If the stems in 1.2 are identical then C is the entity E that is annotated with morphological characters of $\text{last}(C)$. If both E and C have some non-terminal morpheme, that is discarded at annotation (see also Example 4).
- 2. If $\text{last}(E)$ does not meet the condition of (1), i.e. it is either not recognized by MP, or invariable in form (i.e. cannot be inflected), or can be inflected but not nominative word.
 - 2.1 searching
 - 2.1.a If $\text{lchar}(\text{last}(E)) = a$ or $= e$ then the prefix matching is performed with $\text{trunc}(E, 1)$.
 - 2.1.b If $\text{lchar}(\text{last}(E)) \neq a$ or $\neq e$ then the prefix matching is performed with E .
 - 2.2 determination of substitution word
 - 2.2.a If 2.1.a is successful and $\text{lchar}(\text{last}(E)) = a$ then $S = \text{labda}$; and when $\text{lchar}(\text{last}(E)) = e$ then $S = \text{fecske}$.
 - 2.2.b If 2.1.b is successful then we take the provided S of the E .
 - 2.3 annotation
 - 2.3.a The last word of the C has the following form: $[\text{trunc}(\text{last}(E, 1))\{a,e\} \text{rest}]$, where rest is the remaining characters (if any) at the end of

$\text{last}(c)$. The following strings are created and passed to MP: $[\text{trunc}(\text{last}(S, 1))\{a, \acute{a}\}\text{rest}]$ ($[\text{trunc}(\text{last}(S, 1))\{e, \acute{e}\}\text{rest}]$) if $\text{lchar}(E) = a$ ($\text{lchar}(E) = e$), resp. Only one of the strings is a valid word, and hence will be recognized by the MP. The annotation of C will be the morphological characters of the valid words.

- 2.3.b The last word of the C has the following form: $[E\text{rest}]$. The following string is created and passed to MP: $[S\text{rest}]$. The annotation of C will be the morphological characters of the compounded word $[S\text{rest}]$.

Remark 2. One can observe that the first case of the algorithm is more complicated at searching, because the matching is more complex when word of variable form are concerned. In contrast, the second case of the algorithm is more complicated at annotating because the suffixes of the entity can be determined only by a proper substitution word.

Remark 3. The search form of each entity can be stored in the dictionary and generated at the insertion of the entity entry by means of MP. It can save considerable time in the searching phase.

Remark 4. At 2.3 if $\text{length}(\text{rest}) = 0$ then one can omit calling MP, because it means that there are no suffixes on the entity, so it can be considered a nominative noun.

Remark 5. We apply a semi-heuristic algorithm when assigning the substitution word S to each entity of case 2.2.b. S is selected from a table based on the last consonant and the vowels in the last of word of the entity. It is not 100% perfect but assigns good words in the vast majority of case (over 98%).

Remark 6. At 2.3 if $\text{length}(\text{rest}) = 0$ then one can omit calling MP, because it means that there are no suffixes on the entity, so it can be considered a nominative noun.

2.1 Examples

Here we present some example to illustrate the algorithm of ER. The questions are taken from the sample inputs used for testing NL module.

Example 1. See also Figure 3.

*Milyen költők vannak Arany Jánostól József Attiláig?*⁶

$E = \text{József Attila}$, $\text{last}(E)$ is recognized by MP as

$\text{Attila}[\text{noun_prs}] + [\text{NOM}]$

therefore 1) is the actual case. The matching is performed with search form “*József Atti*”, and is successful when matched against segment $C = \text{József Attiláig}$

⁶ What poets are between János Arany and Attila József? (e.g. concerning alphabetical order).

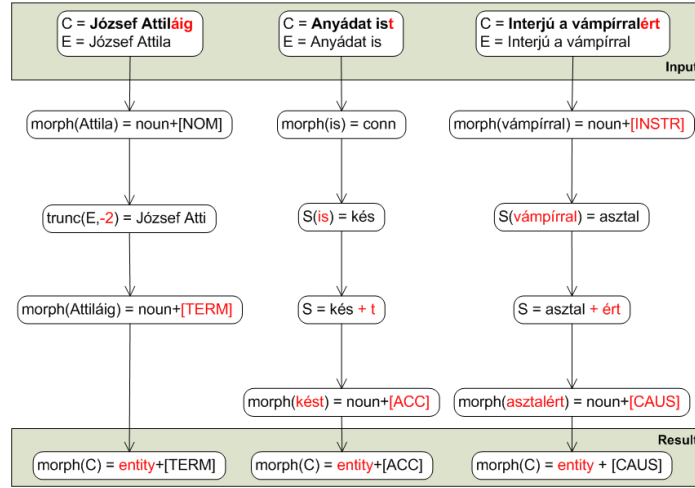


Fig. 3. Illustration of ER's algorithm on examples 1–3

(because the right choice of C is trivial in these examples, we will not specify it explicitly in the next). The result of MP for $\text{last}(C)$ is

$\text{Attila}[\text{noun_prs}] + [\text{TERM}]$

It means that the entity E is found in C and it is annotated as $[\text{TERM}]$.

Example 2. See also Figure 3.

*Ki rendezte az Anyádat ist?*⁷

$E = \text{Anyádat is}$. This is the case 2 (b), because the part of speech of is conjunction that cannot be inflected. Let S be $kés$ ⁸, hence $kést$ is passed to MP. The result is $kés[\text{noun}] + [\text{ACC}]$ so the recognized entity is: $\text{Anyádat is}_{\text{entity}} + [\text{ACC}]$.

Example 3. See also Figure 3.

*Mennyit kell fizetnem az Interjú a vámpírralért?*⁹

$E = \text{Interjú a vámpírral}$. This is also case 2, because $\text{last}(E)$ is an inflected word: $vámpír[\text{noun}] + [\text{INSTR}]$. Let S be $asztal$ ¹⁰, and so $asztalért$ is passed to MP that returns: $asztal[\text{noun}] + [\text{CAUS}/\text{FIN}]$. The final result of ER will be $\text{Interjú a vámpírral}_{\text{entity}} + [\text{CAUS}/\text{FIN}]$.

Example 4. See also Figure 4.

*Ki rendezte Az én kis mosodámat?*¹¹

⁷ Who directed And Your Mother Too?
⁸ knife.
⁹ How much I should pay for the Interview with the Vampire?
¹⁰ table.
¹¹ Who directed My beautiful launderette?

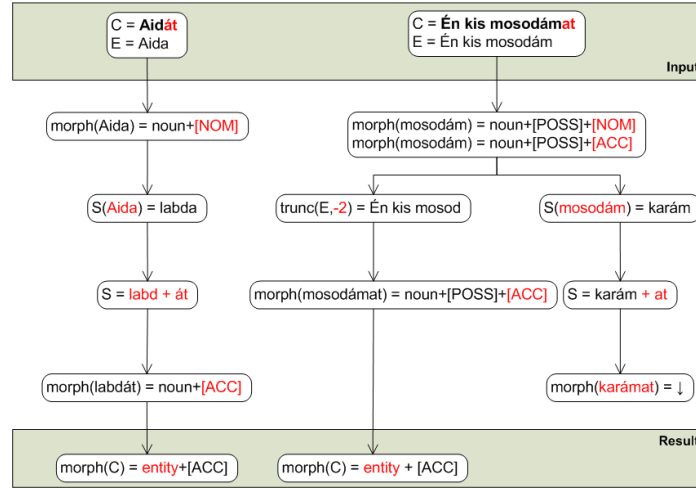


Fig. 4. Illustration of ER's algorithm on examples 4–5

The title of the movie in Hungarian is $E = Az\ \acute{e}n\ kis\ mosodám$, where the stem of the last word (*mosoda*¹²) has a possessive suffix (-*m*). This word (*mosodám*) is further receives the accusative suffix (-*(a)t*). Observe that the entity has to be annotated only with accusative suffix. This entity activates both cases because the result of MP is:

mosoda [noun] + [POSS_SG_1] + [ACC]
mosoda [noun] + [POSS_SG_1] + [NOM]

The first row infers case 2. Let S be *karám*¹³; we create the string *karámat* based on the algorithm that is an invalid word, so this branch does not find any entity. The second row calls case 1. The stem of $last(E) = mosodám$ and $last(C) = mosodámat$ are identical, thus E is a prefix of C . Finally, the morphological characters are obtained as the difference of the morphosyntax of $last(C)$ and $last(E)$: as $\acute{E}n\ kis\ mosodám_{entity} + [ACC]$.

Example 5. See also Figure 4.

*Hol játsszák az Aidát?*¹⁴

$E = Aida$. This is case 2 (a), because $last(E)$ is not recognized by MP. Let S be *labda*¹⁵, and so *labdát* is passed to MP that returns: *labda* [noun] + [ACC]. The final result of ER will be $Aida_{entity} + [ACC]$.

¹² laundry.

¹³ hurdle.

¹⁴ Where Aida is played?

¹⁵ ball.

2.2 Algorithm of the Tokenizer

The algorithm of the tokenizer is straightforward based on ER. It processes each segment of the question in the order specified above. If there are no entities starting with a given word W of the question, then a token is created from W itself and annotated with the respective output of MP. Whenever a single word is not recognized by MP, it is treated as a nominative noun.

3 Conclusions

In this paper we presented some results of the “In the web of words” (WoW) project that aimed to create 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 with special focus on the entity recognizer algorithm. Its goal is to recognize known entities (based on dictionary) in all possible inflected form. By means of a morphological parser we also determine morphological characteristics of recognized entities.

References

1. Bergman, M.K.: The deep web: surfacing hidden value. *Journal of Electronic Publishing* **7** (2001) <http://www.press.umich.edu/jep/07-01/bergman.html>.
2. Winkler, H.: Suchmaschinen. metamedien im internet? In Becker, B., Paetau, M., eds.: *Virtualisierung des Sozialen*, Frankfurt/NY (1997) 185–202 (In German; English translation: http://www.uni-paderborn.de/~timwinkler/suchm_e.html).
3. <http://www.askjeeves.com/>.
4. <http://www.answerbus.com/>.
5. <http://www.ionaut.com:8400/>.
6. Popescu, A.M., Armanasu, A., Etzioni, O., Ko, D., Yates, A.: Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In: *Proc. of Int. Conf. on Computational Linguistics (COLING04)*, Geneva, Switzerland (2004)
7. <http://www.ai.mit.edu/projects/infolab/>.
8. Katz, B., Lin, J.L.: Start and beyond. In: *Proc. of World Multiconference on Systemics, Cybernetics and Informatics (SCI02)*. Volume XVI. (2002)
9. Stratica, N., Kosseim, L., Desai, B.C.: A natural language processor for querying Cindi. In: *Proc. of SSGRR 2002*, L’Aquila, Italy (2002)
10. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Masque/SQL – an efficient and portable natural language query interface for relational databases. In: *Proc. of IEA/AIE 93 Conference*, Edinburgh (1993) 327–330
11. Warren, D., Pereira, F.: An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics* **8** (1982) 110–122
12. Grosz, B.J., Appelt, D.E., Martin, P.A., Pereira, F.C.: Team: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence* **32** (1987) 173–243

13. Rangel, R.A.P., Gelbukh, A.F., Barbosa, J.J.G., Ruiz, E.A., Mejía, A.M., Sánchez, A.P.D.: Spanish natural language interface for a relational database querying system. In Sojka, P., Kopecek, I., Pala, K., eds.: Proc. of TSD 2002. Volume 2448 of Lecture Notes in Computer Science. Springer-Verlag, Brno, Czech Republic (2002) 123–130
14. <http://www.l11f.uam.es/proyectos/sylvia.html>.
15. Cedermark, P.: Swedish noun and adjective morphology in a natural language interface to databases. Master thesis, Uppsala University, Department of Linguistics (2003)
16. Reis, P., Matias, J., Mamede, N.: Edite: A natural language interface to databases – a new perspective for an old approach. In: Proc. of ENTER'97. Information and Communication Technologies in Tourism, Edinburgh (1997) 317–326
17. Filipe, P.P., Mamede, N.J.: Databases and natural language interfaces. In: Proc. of 5th Jornada de Engenharia de Software e Bases de Dados (JISBD), Valladolid (2000) 321–332
18. Meng, X., Wang, S.: Overview of a chinese natural language interface to databases: Nchiql. *International Journal of Computer Processing of Oriental Languages* **14** (2001) 213–232
19. Lee, H., Park, J.C.: Interpretation of natural language queries for relational database access with combinatory categorial grammar. *International Journal of Computer Processing of Oriental Languages* **15** (2002) 281–303
20. Pazienza, M.T., Stellato, A., Henriksen, L., Paggio, P., Zanzotto, F.M.: Ontology mapping to support ontology-based question answering (2005)
21. Hunmorph: (2004) <http://www.szoszablya.hu/termekek/hunmorph/>.
22. Tikk, D., Kardkovács, Z.T., Andriská, Z., Magyar, G., Babarczy, A., Szakadát, I.: Natural language question processing for hungarian deep web searcher. In: Proc. of IEEE Int. Conf. on Computational Cybernetics (ICCC04), Wien, Austria (2004) 303–309
23. Katz, B., Yuret, D., Lin, J., Felshin, S., Schulman, R., Ilik, A.: Blitz: A preprocessor for detecting context-independent linguistic structures. In: Proc. of the 5th Pacific Rim Conference on Artificial Intelligence (PRICAI '98), Singapore (1998)