NAMED ENTITIES AND THEIR TREATMENT IN A COMPUTATIONAL LINGUISTIC IMPLEMENTATION

CSERNYI GÁBOR
Department of English Linguistics, University of Debrecen, Hungary

Abstract: *The aim of this paper is to provide an overview of the nature of names that are relevant for rule-based (syntactic and morphosyntactic) parsing, and it also presents a method for the treatment of names using the framework of Lexical-Functional Grammar, in connection with a Hungarian treebank project.*

Keywords: *computational linguistics, Hungarian, Lexical-Functional Grammar, named entity*

1. Introduction

The study of *named entities* (or names, in general terms) is a well-known field of *Natural Language Processing* (NLP) and as Nadeau and Sekine (2007) also reports, it has been dealt with in the literature as a distinct area since the 1990s with respect to the recognition and classification of names in raw text (i.e. *Named Entity Recognition*, or *NER*, for short). However, it is not only their identification and categorization that appear to be challenging issues but the treatment of names in syntactic and semantic models, as well, raising serious questions both from the theoretical and the computational linguistic perspective.

This paper, with examples from English and Hungarian, discusses those properties of names that might cause problems in terms of writing computational grammars for parsing, and outlines a computational implementation using the theoretical framework of *Lexical-Functional Grammar*. The need for an adequate, consistent analysis of names is also to be reflected by the results of an experiment carried out on the Hungarian Szeged Treebank.

2. The concept of names

In order to give an overall account of names regarding both recognition and processing (or parsing), it is essential first to give a definition of names, and to specify those characteristics of names that NLP applications should be able to deal with. Nevertheless, it must be pointed out that named entity recognition and parsing structures containing named entities will be distinctly discussed, with special emphasis on the latter issue. This section is concerned with the concept *name*, and with concrete examples, reviews those factors of names that applications of both types have to account for.

2.1. What is a name?

From the theoretical and the computational linguistic aspect, and for the purpose of text processing it is generally desirable to outline what phenomenon should be considered a *name*. When providing a definition the focus of attention, by necessity, should be limited to those cases which pose problems to parsing systems and to computer-based, automated identification that can even allow for tagging.

(1) Bank of England

(2) the center of financial issues

Comparing (1) with (2) in this respect, the field of interest is rather examples like (1) representing proper names (or rigid designators, in Kripke's (1980) words), while examples demonstrated by (2) are to be excluded from this analysis. Phrases as (2) should not be a difficult task to handle syntactically without special lexicons, as opposed to examples like (1) that in the majority of cases require special gazetteers (i.e.: lexicon of names, especially geographical ones).

This paper adopts Simon's (2008) definition, according to which an expression can only be a name if it refers to an entity in a unique way. Consequently, those expressions or phrases that are not unique (though refer to unique entities) are not to be identified as names. Therefore, along these lines, (1) is a name, while (2) is not.

2.2. General characteristics

Other facts characteristics of names are that they can often be multiwords (e.g.: European Union), and that they can appear in text as acronyms (e.g.: UNESCO). Regarding constituency issues, it is also common that they occur with a common noun which describes the type of entity the name itself refers to, like in (3). Such common nouns, again, following Simon (2008), are not to be treated as part of the name, as opposed to counterexamples in which the common noun is "institutionalized" with the name as one, reflected by (4).

(3) Mississippi *river*

(4) Black Sea

(5) *the wonderful* Grand Canyon

Moreover, specifiers and other modifiers of noun phrases containing a name as their head exemplified in (5) are generally not to be taken as parts of the name. It is not the task of named entity recognition to identify names together with other constituents belonging to the same phrase, rather the head alone (that is the name itself). (However, as will be seen, named entity recognizers do not necessarily operate on syntactic categories or presyntactic analyses.) Furthermore, parsers are expected to account for these constituents at the syntactic level of analysis in such a way that the name itself should be understood as one syntactic (and semantic) unit on its own, functioning as a head.

3. Natural language processing and names

3.1. Recognition and classification

In the last decade numerous named entity recognition tasks were carried out to show improving results regarding the identification and even classification of names in raw text. The machine learning oriented, statistics-based models used in the experiments range from *artificial neural networks* (ANN) to *decision trees* and *support vector machines* (SVM). The main idea behind these systems is that after a learning phase in which these systems are exposed to (annotated) corpora especially designed for the purpose of training, they derive generalizations based on which they can recognize names in plain text and sort them into categories as organizations, personal names, etc. A sample comparing the accuracy of such models trained on the named entity subcorpus of the Hungarian Szeged Treebank can be found in Szarvas et al. (2006).

Unlike some of these named entity recognizers that are mainly language-dependent, Varga and Simon (2007) outlines a maximum-entropy method that, as claimed, is more flexible concerning language-dependency issues and considers several factors (of name candidates) including positional and morphological ones, making the model work quite efficiently and with high precision. For details, see Varga and Simon (2007).

As for the parsing side, where structures should be morphologically and syntactically (and semantically) adequately processed and analyzed, rule-based systems (i.e.: applications that use phrase-structure rules and operate on categories when parsing) tend to be reliable, better-known and more popular. However, the challenge in these cases, as will be seen, is to prepare the parser to provide analysis of structures possibly containing names (single words, multiwords and acronyms) that are, by default, not part of their core or basic lexicon. The real problem is that, if such a system is not able to analyze fragments (phrase/clause chunks), an unknown expression in the text, like a name, might result in an output without a single parse. The following subsection discusses a Hungarian LFG-approach and its implementational capabilities in relation with names.

3.2. HunGram: an implementational LFG-approach

HunGram (*Hungarian Grammar*) is a computational linguistic project that aims at developing a *Lexical-Functional Grammar* (*LFG*) analysis of the Hungarian language and using it for building a 1.5 million word treebank. HunGram was initiated as part of the *Parallel Grammar* (*ParGram*) research project that is an international cooperation of generative and computational linguists within the theory of LFG. The main idea of the cooperation is to analyze more and more languages and develop large-scale grammars using the framework of LFG. As the theory was also designed for computational application, its main principles are *strong lexicalism*, *modularism*, *rule system based on mathematical formalism* and *parallel architecture*. (For details on LFG, see Bresnan 2001, on ParGram, see http://pargram.b.uib.no; on the *HunGram* project, see Laczkó – Rákosi – Tóth 2010, and http://hungram.unideb.hu)

The *Xerox Linguistics Environment* (*XLE*), the implementational platform (Butt et al. 1997), is a rich system comprising a series of tools that make it possible to parse sentences. The architecture reflects a modular pattern (see *Figure 1*). First the sentence to be parsed is tokenized by finite-state transducers, after which the tokens are sent to the morphological component. Morphological analysis is followed by lexical lookup in the manually created and tagged lexicon that stores subcategorization frames and semantic information. The *lexicon*, with special lexical entries including grammatical functional annotations as well, is a crucial part of the system, this is the module on which the essence of the theory lies. Its importance together with the morphological analyzer is even more emphasized in case of highly inflectional languages like Hungarian.

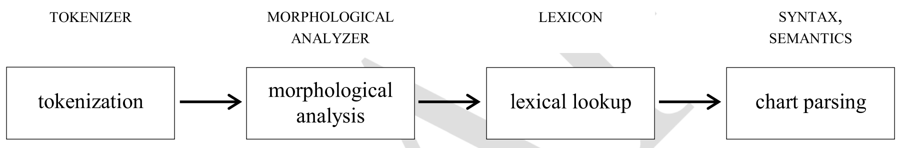


Figure 1: Parsing with XLE

The morphological features the morphological analyzer provides are mapped to lexical entries which, in turn, are used by the chart parser to determine the syntactic structures available. (If no morphological analyzer exists, the tokens are forwarded directly to the lexicon after tokenization. In this case, the lexicon has to be powerful, containing all the forms.)

3.2.1. Names in the model

In relation with names, the task in such a model is either to build a rich lexicon of names, or to prepare the morphological component in that it should produce the relevant morphological analysis for names as well including inflections, plus it should also add a distinct feature tag marking that the particular token is a name. The output of an analysis of this kind is demonstrated by the Hungarian example *Európai Unióban* ‘in the European Union’ in (6), in which the stem is *Európai Unió* and it is in inessive case.

(6) *Európai Unióban*: Európai Unió +Noun +Prop +Sg +Ine

However, as this example suggests, and as it has been claimed in section 2.2, names are often multiwords, which means they are to be treated as one token, a fact to prepare the tokenizer for. Although this component, by default, takes every word as a separate, single token, it is possible to configure XLE through the grammar configuration file not to process multiword tokens as individual items in case they constitute a multiword that is either listed in the lexicon and/or the morphology accounts for it.

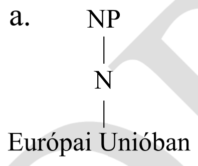
Morphologies relying on finite-state techniques have proved to be quite efficient in the ParGram projects because of the fact that they are quite fast, they can provide multiple analyses in case of ambiguity, and that they can easily be extended by additional transducers like guessers (cf. Kaplan et al. 2004). Embedding names into already built morphologies has the advantage that there is no need for including all the names in all possible forms (concerning inflections), only stems have to be inserted under the appropriate category or group (which can serve as a filter to what inflections can be accepted with the name). In addition, guesser transducers can also promote parsing results. All these, however, require having a rich morphological analyzer of the particular language and more importantly, the base lexicon of this module (that is, by nature, completely different from the lexicon containing entries with grammatical features and predicate-relations as in (7)) should also be accessible.

Another possibility apart from the morphological solution is to put the names into the lexical component (i.e.: build a lexicon for names). While (6) gave an example for how names should be represented in the morphological component, (7) is a counterpart of it in the form of a lexical entry.

(7) Európai Unióban N (↑ PRED) = 'Európai Unió'
(↑ NUM) = sg
(↑ PERS) = 3
(↑ CASE) = ine
@(NSYN proper)
@(NSEM organization).

This lexical entry produces exactly the same analysis as the morphology output in (6). The only additional information in this case is introduced by the template call claiming that the name semantically represents an organization, but as grammatical functional annotations can be assigned to morphological tags in the lexicon, this is not to be understood as a difference.

(8) a.



b.

$$\left[\begin{array}{ll} \text{PRED} & \text{'Európai Unió'} \\ \text{NTYPE} & \left[\begin{array}{l} \text{NSEM } [\text{PROPER } [\text{PROPER-TYPE organization}]] \\ \text{NSYN proper} \end{array}\right] \\ \text{CASE ine NUM sg, PERS 3} & \end{array}\right]$$

Therefore, no matter which method is used in this particular case, *Európai Unióban* parsed as a noun phrase produces the output shown in (8). These two structures represent the core syntactic levels of LFG: the constituent structure (8a) that reflects surface-level properties as word order and constituency, and the functional structure (8b) responsible for language-invariant features like predicate-argument relations and grammatical features. For a detailed explanation, see Bresnan (2001).

3.2.2. Form and orthography

Apart form the fact mentioned in section 2.2 that names can also take the form of acronyms there are several other form-related and orthographically questionable variations among names, to which attention must be paid. This subsection reflects patterns that one might come across when examining data in corpora.

(9) a. *Magyar Nemzeti Bank* vs. *MNB*
Hungarian National Bank

b. *Ady Endre Gimnázium* vs. *Ady Gimnázium*
Ady Endre Grammar School

The examples in (9) represent the problem referred to above. It is not uncommon that within the same text a name takes different forms. In some cases the name is in its full form, while in others it is an acronym, see (9a). Moreover, in informal texts, elements of names (compared to the full name) can also be missing, as (9b) shows. However, it is not only form-variation of this kind that can suggest a sort of inconsistency and make it difficult to cover such cases, but questions related to orthography can also be raised.

(10) a. *iPhone*

b. *Mastercard* vs. *MasterCard*

c. *Noname Ltd.* vs. *Noname Ltd*

d. *Földhitel- és Jelzálogbank* vs. **Földhitel és Jelzálogbank*

As parsing is generally case-sensitive (it does matter, for example, if a lexical entry contains uppercase letters or not), it is of crucial importance to consider instances like (10a) and (10b). The problem with forms like (10b) is that they require inserting all the form-variants into the lexicon. Deciding on the morphological module and not the lexicon to account for the names, lowercasing before determining the appropriate morphological features might turn out to be a solution. Utilizing transducers to normalize input in general (e.g. to remove or add dots in company names and company type markers) might be beneficial to avoid problems of punctuation and minor form-alternations of this kind to keep the representation consistent. Unfortunately, there can still be orthographically ill-formed patterns that are quite difficult to cover (10d).

3.2.3. Morphological issues

Names in some languages, just like other nouns, are open to morphological processes, such as suffixation. In Hungarian, certain inflections are attached to the stem without morphophonological changes (11), others affect the stem, or the inflection itself as in the case of assimilation (12).

(11)	Kovács-*nak*	(Kovács + -*nak*)
	Smith-DAT	
	‘*to Smith*’	
(12)	János-*sal*	(János + -*val*)
	John-with	
	‘with John’	

In Hungarian, according to the orthography, when the derivational suffix -*i* is added to a name stem (to derive an adjective), the initial uppercase letter has to be lowercased (13).

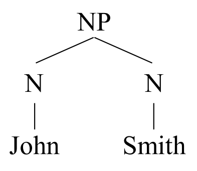
(13)	Vietnám + -*i*	→	vietnámi
	Vietnam + -*ese*	→	Vietnamese

If it is the morphological component of the grammar that is responsible for the lexical-level analysis of forms, then these problems can be easily overcome: as names can only receive those inflectional suffixes that common nouns can have, the task is just to configure those suffix continuation classes to the names in the morphological analyzer that common nouns are set to. When it is not possible to extend the morphology module with names, the grammar writer needs to list all the relevant forms of a name that s/he wants the parser to analyze.

3.2.4. Further issues: syntax and semantics

An even more challenging question that does not seem to have been covered in detail in implementational models is the syntactic and semantic representation of names. While single-word name forms are inserted under a nominal node in the constituent structure, multiword names allow for exploiting as many different (sub)categories under a branching nominal node as many tokens the name consists of. According to this view, the tokens are considered distinct lexical units. However, this also entails that one of the parts of the name has to be the head of the whole name. An example from the English ParGram is demonstrated in (14).

(14) a.



b.

```
| PRED        'Smith'                                                                   |
|                                                                                       |
| NAME_MOD  | PRED    'John'                                                          | |
|           | NTYPE | NSEM | PROPER |PROPER-TYPE name, NAME-TYPE first_name | |  |   | |
|           |       | NSYN proper                                              |   | |
|           | GEND-SEM male, HUMAN +, NUM sg, PERS 3                              | |
|                                                                                       |
| NTYPE     | NSEM | PROPER |PROPER-TYPE name, NAME-TYPE last_name | | |               |
|           | NSYN proper                                          |                   |
|                                                                                       |
| HUMAN +, NUM sg, PERS 3                                                               |
```

As the constituent structure in this example illustrates, in the name *John Smith*, each part of the name is inserted under a distinct node *N*, each of which is a daughter of the *NP* node (see 14a). The functional structure (14b) also shows that in this analysis the last name is the head, and the first name is its modifier. In Hungarian, on the other hand, it is the last name that comes first in sequential order and inflections are always attached to the first name. This means that it is the first name that, from the morphological perspective, calls for treating it as the head, which in turn does not permit to take the last name as the head. One possibility to avoid this problem is to represent the full name as a single lexical unit. In the English ParGram there are also examples for this type of analysis, like *Trafalgar Square* in (15). (In highly inflectional languages, like Hungarian, inflectional endings are attached to the whole lexical complex in this approach.)

```
(15)  a.        NP             b.  | PRED      'Trafalgar Square'                                     |
                 |                 |                                                                  |
                 N                 | NTYPE    | NSEM |PROPER |PROPER-TYPE  location| | |              |
                 |                 |          | NSYN proper                              |              |
          Trafalgar Square         | NUM sg, PERS 3                                                   |
```

The need for a unified and consistent analysis is even more manifested by instances of brand + type (+subtype) multiword name constructions (e.g.: 'Ford Model T'). These patterns might have a preference for taking each token of multiword names, in general, as separate lexical items, but this approach still appears to be difficult to implement as reflected by the treatment of Hungarian full names in this way. On the other hand, a less economical way of storing whole multiwords, each as a single lexical item, might require a lot of work on the part of the grammar writer, resulting in large lexicons, nevertheless, this option poses fewer problems computationally.

3.3. A case study: names in the Szeged Treebank

The evidence for the high frequency of names in corpora is proved by the following experiement inteded to extract all the names from the Szeged Treebank for statistics and for lexicon building purposes. This treebank is based on a 1.2 million word corpus, in which the texts are categorized thematically involving written texts collected from a variety of sources ranging from business papers to essays of students aged 14 to 16 (for details, see Csendes et al. 2005). The results of this study, shown in *Figure 2*, demonstrate that 25.904 out of 82.099 sentences (total number) contain at least one name, which also entails that there is one in every fourth sentence.

subcorpus	**Ss**	**Ss with NEs**	**NEs**	**NE lemmas**	**multiword NE lemmas**
/business-news/newsml.xml	9 574	6 735	13 650	4 214	2 810
/compositions/10elb.xml	9 541	1 316	1 659	753	179
/compositions/10erv.xml	7 604	160	202	147	37
/compositions/8oelb.xml	7 575	995	1 219	599	156
/computer/cwszt.xml	6 676	3 219	5 699	2 410	1 373
/computer/win200.xml	3 083	1 254	2 550	735	430
/fiction/1984.xml	6 658	1 729	2 140	185	56
/fiction/pfred.xml	6 485	1 421	1 692	225	78
/fiction/utas.xml	5 415	1 946	2 634	307	92
/law/gazdtar.xml	5 734	1 657	2 119	79	31
/law/szerzj.xml	3 544	1 166	1 764	311	200
/newspapers/hvg.xml	2 369	1 286	2 623	1 469	843
/newspapers/mh.xml	2 435	954	1 692	971	509
/newspapers/np.xml	4 107	1 497	2 570	1 452	732
/newspapers/nv.xml	1 299	569	1 079	729	385
whole corpus (all included):	**82 099**	**25 904**	**43 292**	***12 546***	***7 386***

Figure 2: Named entities in the Szeged Treebank

This index is even lower in the case of business news (where, on average, every second sentence has a name in it) and computer texts and newspaper articles (in which every third sentence contains a name). Statistics also indicate that the number of lemmas extracted concerning the whole corpus is 12.546, which appear in 43.292 different forms, and out of this considerably high number of name lemmas 7.386 are multiword ones.

Taking a look at the data based on the type of texts might help decide on the type of source for testing a computational grammar described in section 3.2 without preparing it for analyzing names. But even those parts of the corpus in which the number of sentences with names is quite representative, and except for compositions of students, the parser will still not provide analysis for at least 21% of the sentences (see line */fiction/pred.xml* in *Figure 2*).

4. Conclusion

This article demonstrated the basic problems that computational LFG grammar writers face in connection with parsing structures containing (proper) names. Augmenting grammar implementations of this kind with a statistics-based machine learning approach that identifies and extracts names from the texts to be parsed in a preprocessing phase to build named entity lexicons, either for morphological analyzers or as part of the lexicons of the system, might improve parsing efficiency; however, a detailed analysis of the syntactic inner structure of names is still desirable if it can reduce the size of such lexicons and introduce a consistent language independent treatment of names, especially multiword ones, reflecting their syntactic complexity.

Notes on the author

Gábor Csernyi is a Ph.D. student with special interest in computational linguistics and with general theoretical interest in generative grammar, mainly Lexical-Functional Grammar.

Acknowledgements

The authors acknowledge that the research reported here is supported, in part, by OTKA (Hungarian Scientific Research Fund), grant number: K 72983; by the TÁMOP-4.2.1/B-09/1/KONV-2010-0007 project, which is implemented through the New Hungary Development Plan co-financed by the European Social Fund and the European Regional Development Fund; and by the TÁMOP-4.2.2/B-10/1-2010-0024 project, which is co-financed by the European Union and the European Social Fund.

References

Bresnan, J. 2001. *Lexical-Functional Syntax*. Oxford: Blackwell.

Butt, M.; King, T. H.; Niño, M. E. and Segond, F. 1997. *A Grammar Writer's Cookbook*. Stanford: CSLI Publications.

Csendes, D.; Csirik, J.; Gyimóthy, T.; Kocsor, A. 2005. 'The Szeged Treebank' in *Lecture Notes in Computer Science: Text, Speech and Dialogue*. V. Matousek, P. Mautner and T. Pavelka (eds.). Berlin: Springer, pp. 123-131.

Kaplan, R.; Maxwell, J.; King, T. H. and Crouch, R. 2004. 'Integrating Finite-state Technology with Deep LFG Grammars' in *Proceedings of the ESSLLI'04 Workshop on Combining Shallow and Deep Processing for NLP*, pp. 11-20.

Kripke, S. 1980. *Naming and Necessity*. Oxford: Blackwell.

Laczkó, T.; Rákosi, Gy. and Tóth, Á. 2010. 'HunGram vs. EngGram in ParGram' in *CrosSections Volume 1: Selected Papers in Linguistics from the 9th HUSSE Conference.* I. Hegedűs and S. Martsa (eds.). Pécs: Institute of English Studies, Faculty of Humanities, University of Pécs, pp 81-95.

Nadeau, D. and Sekine, S. 2007. 'A survey of named entity recognition and classification' in *Linguisticae Investigationes*. 30/1, pp. 3-26.

Simon, E. 2008. 'Nyelvészeti problémák a tulajdonnév-felismerés területén' in *LingDok 7. Nyelvész-doktoranduszok dolgozatai*. B. Sinkovics (ed.). Szeged: Doctoral School in Linguistics, University of Szeged, pp 181-196.

Szarvas, Gy.; Farkas, R.; Felföldi, L.; Kocsor, A. and Csirik, J. 2006. 'A highly accurate Named Entity corpus for Hungarian' in *Proceedings of LREC (2006)*, pp. 1957-1960.

Varga, D. and Simon, E. 2007. 'Hungarian named entity recognition with a maximum entropy approach' in *Acta Cybernetica*. 18/2, pp. 293-301.