e-magyar – A Digital Language Processing System

Tamás Váradi[1], Eszter Simon[1], Bálint Sass[1], Iván Mittelholcz[1], Attila Novák[2,3], Balázs Indig[2,3], Richárd Farkas[4], Veronika Vincze[4,5]
[1]Research Institute for Linguistics, Hungarian Academy of Sciences
H-1068 Budapest, Benczúr u. 33.
[2]MTA-PPKE Hungarian Language Technology Research Group
[3]Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
H-1083 Budapest, Práter u. 50/a
[4]University of Szeged, Institute of Informatics
H-6720 Szeged, Árpád tér 2.
[5]MTA-SZTE Research Group on Artificial Intelligence
H-6720 Szeged, Tisza Lajos krt. 103.
{varadi.tamas,simon.eszter,sass.balint,mittelholcz.ivan}@nytud.mta.hu,
{novak.attila,indig.balazs}@itk.ppke.hu, {rfarkas,vinczev}@inf.u-szeged.hu

Abstract

e-magyar is a new toolset for the analysis of Hungarian texts. It was produced as a collaborative effort of the Hungarian language technology community integrating the best state-of-the-art tools, enhancing them where necessary, making them interoperable and releasing them with a clear license. It is a free, open, modular text processing pipeline which is integrated in the GATE system offering further prospects of interoperability. From tokenizing to parsing and named entity recognition, existing tools were examined and those selected for integration underwent various amount of overhaul in order to operate in the pipeline with a uniform encoding, and run in the same Java platform. The tokenizer was re-built from ground up and the flagship module, the morphological analyzer, based on the Humor system (Prószéky and Kis, 1999), was given a new annotation system and was implemented in the HFST framework (Lindén et al., 2009). The system is aimed for a broad range of users, from language technology application developers to digital humanities researchers alike. It comes with a drag-and-drop demo on its website: `http://e-magyar.hu/en/`.

Keywords: text analysis, Hungarian pipeline, integrated toolset

1. Introduction

The paper describes e-magyar, a new integrated text processing pipeline for Hungarian. While Hungarian can be considered an under-resourced language it does have an active and cooperating language technology community which has been developing various tools to cover the basic text processing steps. However, earlier fragmented efforts suffered from a number of factors such as the lack of interoperability, openness, clearly defined license conditions and/or have become limited in some technological aspects such as encoding and annotation systems used as well as efficiency and implementation platform. All these reasons served as the motivation for a collaborative effort by key Hungarian language technology partners to overhaul (sometimes quite radically) the existing tools and, more importantly, to make them interoperable so that they can be integrated into a single coherent technological pipeline.
The objectives of the e-magyar system were to serve as an *open, free and modular* text processing toolset that serves the needs of commercial developers and individual researchers in language technology as well as (digital) humanities and social sciences. It is open and free in that the system as a whole and its individual components are typically available for download through Github repositories, freely available often not just for research and development but for commercial use and they certainly come with clear license terms.
Technologically, the aim was to take the state-of-the-art tools available, eliminate their shortcomings either in specifications, functionality or efficiency and integrate them in a single system so that the performance of the individual module in the e-magyar system should be at least equal that of the original tool before its overhaul.
Specific attention was paid to ensure that the toolset was accessible and useful not just for developers but researchers in the social sciences and humanities (SSH). This reflects an increasing awareness within user involvement efforts in the CLARIN community[1] that SSH researchers are less interested in pre-annotated datasets than in toolsets that are capable to process their own data. To serve the needs of non-language technology specialists, a web service was set up for them to process their data in a drag-and-drop fashion and integration in the GATE system, which has a user-friendly graphical interface, which also facilitates the use of the e-magyar toolset.

2. Text Modules

The e-magyar digital text processing system is assembled as follows. Starting from raw text, the first module (called emToken) performs word and sentence segmentation (see Section 2.1.). Then full-fledged morphological analyses and the lemma of the tokens are identified (emMorph and emLem modules, Section 2.2.). Morphological disambiguation is performed by the emTag part of speech (POS) tagger module (Section 2.3.). Syntactic analysis is accomplished in two different ways: con-

[1]`https://www.clarin.eu/`

stituency analysis (emCons module) and dependency analysis (emDep module) are also assigned to sentences (Section 2.4.). Finally, maximal noun phrases (emChunk module, Section 2.5.) and named entities (emNer module, Section 2.6.) are identified as well. The modules are built on one another as Figure 1 shows.

2.1. Tokenizer and Sentence Splitter

A brand new Hungarian tokenizer and sentence splitter has been developed, called quntoken[2], which is based on the Quex[3] lexical analyzer generator and was implemented in C++. This tool was integrated into the e-magyar language processing system under the name emToken.
In its features, it mainly follows HunToken[4], which is a rule-based tokenizer and sentence boundary detector for Hungarian (and English) texts. However, emToken differs in several properties, e.g. its input is plain text in UTF-8 encoding, and its output is the text segmented into sentences and words in XML or JSON format. The output can be detokenized, i.e. the input can be reproduced from the output. It can be run as a standalone program or via an API. It comes with test cases covering all the built-in tokenizing rules.
The evaluation of the performance of emToken was made on the Szeged Treebank (Csendes et al., 2005). As for the sentence segmentation, the emToken failed in 2,131 cases of 81,648, which means a 97.39% accuracy. As for the tokenization, we counted word accuracy, which was 99.27% (10,903 false segmentation for 1,478,300 tokens). Most of the mistakes are due to the differences between the word and sentence segmentation schemes applied in the Szeged Treebank and in emToken.

2.2. Morphological Analyzer and Lemmatizer

The emMorph morphological analyzer (MA) integrated in the system was implemented using the Helsinki Finite-State Transducer (HFST) toolkit (Lindén et al., 2009). The morphological database is primarily based on the Hungarian computational morphology (Novák, 2003; Novák, 2015) originally created for the Humor morphological analyzer (Prószéky and Kis, 1999), which was extended with vocabulary from the morphdb.hu database (Trón et al., 2006). The grammar implemented in the constraint-based Humor formalism was converted to a finite-state description following the procedure described in Novák (2014). The morphological grammar development platform generates descriptions of allomorphs including their features and morph adjacency constraints from morpheme definitions applying a procedural rule system. The word grammar describing well-formed morpheme sequences (including the description of non-local constraints between morphemes) is defined using an extended finite-state automaton. All these data structures are implemented as a single finite-state lexical transducer in the HFST representation of the morphology, including a flag-diacritics-based representation of the word grammar automaton.

[2]https://github.com/dlt-rilmta/quntoken/
[3]http://quex.sourceforge.net/
[4]https://github.com/zseder/huntoken

The hfst-lookup MA engine was extended in several ways to improve its performance. Dynamic FST composition was added to the implementation, so that the FST performing case conversion of capitalized words and those in all caps need not be composed with the lexical transducer off-line. This reduces runtime memory requirement of the MA to 1/3. Moreover, in addition to the lexical form, the MA can now also return the surface form of each morpheme.
The latter is used in the lemmatizer integrated in the system, emLem, which was implemented in Java. Most upstream tools do not need the amount of detail present in the analyses returned by the MA (e.g. compound members, derivational suffixes, alternative equivalent analyses of different granularities of lexicalized polymorphemic stems and their productive analyses). The emLem module merges compound members and derivational suffixes into a single stem using the surface form of non-final stem elements and the lexical form of the stem-final morpheme. It computes the resulting POS, adds the morphosyntactic features exposed by inflections, and discards identical results. The lemmatizer is also capable of optionally returning detailed analyses corresponding to each lemma (including surface forms of each morph) in addition to the lemmatized output. The algorithm implemented in the lemmatizer also handles nontrivial Hungarian word constructions (e.g. when inflectional suffixes are present in a non-word-final position) and returns a correct lemma also in those cases.
Previous morphological analyzers for Hungarian used various *ad hoc* tagsets. In contrast, the tagset used by emMorph and emLem contains tags suggested in the Leipzig Glossing Rules (Comrie et al., 2008) widely used by linguists. The tags in Wikipedia's list of glossing abbreviations[5] were also used and extended to include all Hungarian affixes and POS categories.
The MA was tested on the Hungarian Webcorpus (Halácsy et al., 2004) containing 589M words fully filtered. Word token coverage was found to be 99.36% corresponding to a word type coverage of 85.73%. As for analysis speed, hfst-optimized-lookup is 6.14 times faster than the original Humor implementation. Run-time data segment memory consumption of hfst-optimized-lookup is on the other hand at least 26.11 times higher than that of Humor (148 MB if case conversions are not composed with the lexicon vs. 5.6 MB). The engines themselves consume 4-6 MB memory.

2.3. POS Tagger

The emTag POS tagger is based on PurePOS (Orosz and Novák, 2013), which is the continuation and extension of HunPOS (Halácsy et al., 2007), the first POS tagger written directly to handle agglutination in Hungarian. Even though all the aforementioned systems are based on TnT (Brants, 2000), the well-known HMM tagger, each implementation has its own new features to handle Hungarian and similar highly agglutinating languages better.
The main advantage of emTag lies in its line of predecessors, which were developed with performance in mind with

[5]https://en.wikipedia.org/wiki/List_of_glossing_abbreviations

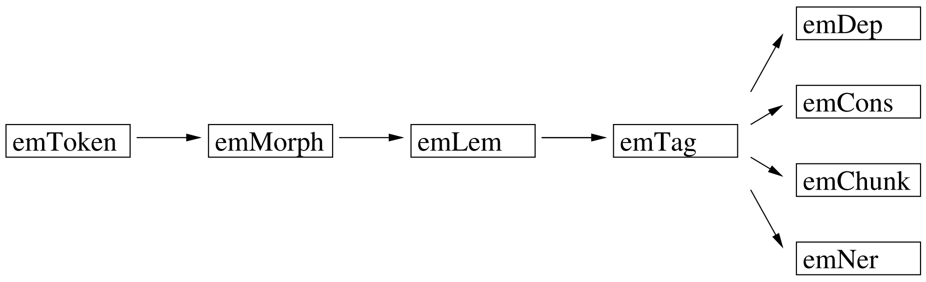

Figure 1: Modules of the `e-magyar` digital language processing system as they are built on one another.

new improved features. PurePOS is the first tagger in the line to add lemmatization support and the possibility to use pre-analyzed input, which essentially improves tagging and lemmatization performance.

From the technical perspective: the *de facto* standard CoNLL-style vertical input and output format is now supported along with the original PurePOS notation. The user is also to set different parameters on lemmatization (e.g. to cut characters from the right side of the token only (suffix) or use a longest common substring-based transformation (circumfixes)) and dump the created model into a human-readable format to check what has been learned. Even though this feature greatly improves the explanatory power of the model, its really rare among mainstream taggers, which makes `emTag` more valuable.

Nonetheless, POS taggers based on supervised learning heavily rely on the quality of the training corpus, the tagging scheme and the morphological analyzer. The performance of `emTag` is on a par with that of PurePOS (Orosz, 2015), which means a 97.58% full disambiguation accuracy on token level when applying lemmatization support as well.

2.4. Constituency and Dependency Parsers

The `e-magyar` toolchain includes constituency and dependency parsers, making it possible to apply two popular approaches for syntactic parsing. The input for both parsers is the output of the previous modules, i.e. tokenized and POS tagged sentences.

Our parser builds heavily on the techniques presented at the workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL), which was dedicated to the parsing of morphologically rich languages, such as Hungarian. An adapted version (Szántó and Farkas, 2014) of the Berkeley Parser (Petrov et al., 2006) – a stochastic context free model – was integrated into the system.

The preprocessing toolkit called `magyarlanc` (Zsibrita et al., 2013) also contains a dependency parser, based on the Bohnet parser (Bohnet, 2010), a language independent dependency parser. The model integrated was trained on the Szeged Dependency Treebank (Vincze et al., 2010). This dependency parser was integrated into the `e-magyar` toolkit, with small modifications, one of whose being the different morphological coding systems applied in `e-magyar` and in the Szeged Treebank. As the constituency and dependency parsers exploit the Hungarian version of the morphological coding system of the Universal Dependencies (UD) project (Vincze et al., 2017), the output of the `emTag` module needed to be converted to the UD morphology.

2.5. NP Chunker

The `emChunk` module identifies maximal noun phrases (NPs), i.e. NPs which are not part of any other higher level NPs. Its input is a text that had previously been processed in the toolchain, i.e. they had been segmented into words and sentences, and words are assigned their full morphological analyses. These pieces of information are necessary for the NP chunker module to be effective. The module assigns a tag to every token in the input text. The tag indicates whether the word is part of a maximal NP, and if yes, whether the NP has one or several components. If the latter, it also indicates whether the given word is an initial, medial or final component of the NP. The output keeps the analyses of the previous processing levels and adds the tags of the chunker module.

The `emChunk` module is based on `HunTag3` (Endrédy and Indig, 2015), which is a sequential tagger for several NLP tasks using maximum entropy and HMM. Its predecessor is HunTag[6], which has been used, among others, for Hungarian named entity recognition (Simon, 2013) under the name `hunner` and for shallow syntactic analysis of Hungarian texts (Recski and Varga, 2010) under the name `hunchunk`. Depending on the training data, `HunTag3` can be used for several sequential tagging tasks.

The gold standard dataset used here was a subcorpus of the Szeged Treebank (Csendes et al., 2005), which contains morphological, syntactic and named entity annotation as well. This subcorpus is actually the same as the one published under the name Szeged NER Corpus (Szarvas et al., 2006), which only contained morphological annotation (earlier in MSD then in UD format) and named entity annotation. The NP chunking annotation was generated from the constituency annotation of the Szeged Treebank, while the morphological annotation had to be converted to the format of `emMorph`.

The model was built by using the whole corpus as the training dataset, which comprises more than 220,000 tokens. Since there is no other Hungarian dataset containing all the required annotation layers in the required format to which the output of `emChunk` could be compared, here we

[6]`https://github.com/recski/HunTag`

cannot provide an evaluation of the module's performance. As Recski and Varga (2010) reports, `hunchunk` performs 86.06% F-score when recognizing maximal NPs in Hungarian texts.

2.6. Named Entity Recognizer

The `emNer` module is an automatic Named Entity Recognizer, which identifies named entities (NEs) in running text and assigns them to one of the predetermined categories. We follow the standard NE classes of CoNLL-2002 (Tjong Kim Sang, 2002) tagging person names, organization names, place names and the so called `Miscellaneous` category which mostly comprises everything else falling outside of the main categories.
Similarly to the other modules, the input of `emNer` is a text that had previously been processed in the toolchain, i.e. they had been segmented into words and sentences, and words are assigned their full morphological analyses. These pieces of information are necessary for the NER tagger module to be effective (Simon, 2013).
The module assigns a tag to every token in the text, indicating whether the given word is a named entity, and if yes, what category it belongs to, and whether it has one or several elements, and if the latter, whether the given word has an initial, medial or final position in the named entity. The output keeps the analyses of the previous processing levels and adds the tags of the NER tagger module.
The `emNer` module is also based on `HunTag3`, and the training data is the same as was in the case of `emChunk`. The model was also built by using the whole corpus as the training dataset, therefore here we cannot provide a thorough evaluation of the `emNer` module. What we can provide is the best performance of `hunner` trained on the 90% of the Szeged NER Corpus and tested on the remaining 10% which was 97.87% F-measure. However, cross-corpus and cross-domain evaluation always result in a 20-30% decrease in overall F-measure as reported by Nothman et al. (2008) and Simon (2013) among others, thus the performance of `emNer` highly depends on the input text type given by the user.

3. GATE Integration

The integration of the `e-magyar` modules described in Section 2. into a unified text analysis toolchain has been implemented in the GATE framework (Cunningham et al., 2011). During the integration, the main task was to enable the modules to take their input from the form corresponding to the *offset*-based annotation model of GATE and produce their output in this form as well. For this reason, we have created a GATE wrapper for each module that performs the required data conversions. It was also necessary to fit the non-Java language tools into the Java language framework: we have solved this by calling directly the binary of the non-Java modules.
Modules are built on one another as Figure 1 shows. The fixed basic processing chain consisting of a tokenizer, a morphological analyzer, a lemmatizer and a morphological disambiguator is followed by additional tools which utilizes the output of the fixed chain and which can be run independently of each other.
The modules are complemented by additional facilities. First, a human-readable format from the detailed morphological analysis is produced. Second, separated verbal particles and verbs are combined together based on the dependency analysis, so providing the full verbal form. Third, the `IOB`-type (specifically `BIE-1`) encoding provided by the `emChunk` and `emNer` modules are converted into a more convenient standalone (NP and NE) annotation format.
The processing chain can be used in four ways. On the website (see Section 4.), the user can simply copy-paste a short text and analyze this text by running the full processing chain with just a mouse click. For more serious text analysis tasks or for digital humanities research, the GUI of the GATE system called *GATE Developer* is recommended, into which the `e-magyar` chain can be easily installed. Installation instructions are available at `https://github.com/dlt-rilmta/hunlp-GATE` along with the entire system. If needed, the user can improve the system with adding custom built-in modules to the chain. For processing larger corpora, using GATE command line access is recommended, which is also available via the URL provided above. As a fourth method, the so-called *gate-server* can be used. This is another command line technology, which operates behind the `e-magyar` web service.

4. Online Interface

The project's objectives included the ability to access and use the text processing chain by users who are not necessarily familiar with the field of language technology. This demand is addressed by the online text analysis service[7] of `e-magyar`, which allows the user to easily test each of the analytic modules or even the entire tool chain via a web interface without using any other software than the web browser.
The text analyzer is based on a web service that uses GATE software libraries. It takes the text and the list of analytical modules that the user wants to run as input and provides GATE generated XML containing the annotations as output. The site processes the output XML and displays the data in an easily interpretable, visualized form.
The analyzer interface consists of two main parts: an input panel and an output panel. The text box on the input panel lets the user specify the text to be analyzed (currently, the length of the text is limited to 6000 characters), then the user can piece together the list of modules wanted to run on the text.
The result of the processing can be seen on the output panel. The analysis can be displayed in two different layouts: 'text' view and 'list' view. In 'text' view, tokens follow each other sequentially, annotations for each token show up in a small bubble when moving the cursor over a given token or clicking on it. In the case of separated verbal particles, the main verb is also highlighted. In 'list' view, each token has its own row in a tabular form, while annotations added by the text analyzer are placed in successive columns. This layout is more suitable for displaying lots of annotations together. In 'list' view, it is possible to filter

[7] `http://e-magyar.hu/en/parser`

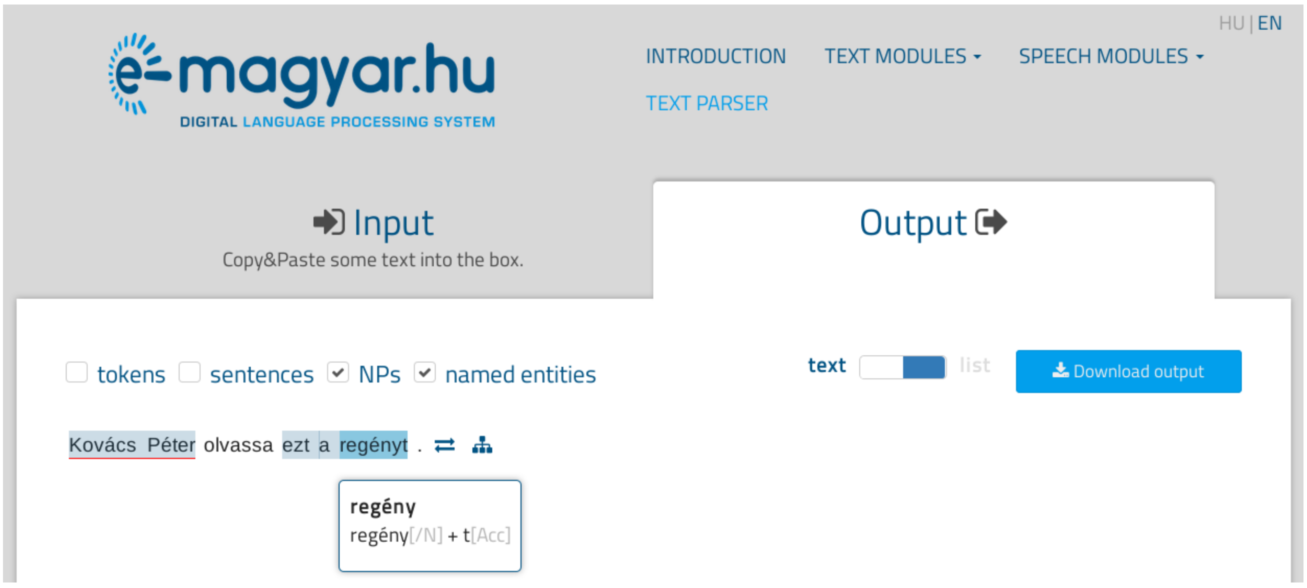

Figure 2: A screenshot of the 'text' view of the e-magyar online interface. The example sentence is *Kovács Péter olvassa ezt a regényt.* 'Kovács Péter reads this novel.' Lemma and morphological analysis is shown in a bubble for a token. Here we see that *regényt* is the accusative form of *regény* 'novel' which is a noun. Noun phrases are highlighted, and named entities are underlined.

the tokens based on different criteria: the user can filter for a word form, for elements of the morphological analysis, for POS tags or for several grammatical functions. In both views, it is possible to highlight certain segments (of one or more tokens) created by some analyzer modules: tokens, sentences, noun phrases, and named entities. An illustration can be seen in Figure 2. The results of the syntactic analyses can be accessed in the 'text' view by clicking on the icons next to each sentence: a dependency tree and a constituent tree diagram, the output of the dependency parser and the constituency parser, respectively.
The result of the analysis – the text with all added annotations – can be downloaded for further use. The downloaded zipped file contains three files: raw text sent for processing as a plain text file, the GATE generated output XML file, and an extract of the 'list' view in tsv format.

5. Acknowledgements

The work reported in this paper was supported by the Hungarian Academy of Sciences (Infrastructure Development Support Programme 2015/2).

6. Bibliographical References

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of Coling 2010*, pages 89–97.

Brants, T. (2000). T'n'T: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics.

Comrie, B., Haspelmath, M., and Bickel, B. (2008). The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses.

Csendes, D., Csirik, J., Gyimóthy, T., and Kocsor, A. (2005). The Szeged Treebank. In *Lecture Notes in Computer Science: Text, Speech and Dialogue*, pages 123–131. Springer.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M. A., Saggion, H., Petrak, J., Li, Y., and Peters, W. (2011). *Text Processing with GATE (Version 6).*

Endrédy, I. and Indig, B. (2015). HunTag3: a general-purpose, modular sequential tagger – chunking phrases in English and maximal NPs and NER for Hungarian. In *7th Language & Technology Conference, Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC '15)*, pages 213–218, Poznań, Poland, November. Poznań: Uniwersytet im. Adama Mickiewicza w Poznaniu.

Halácsy, P., Kornai, A., and Oravecz, C. (2007). Hunpos: An open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 209–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I., and Trón, V. (2004). Creating open language resources for Hungarian. In *Proceedings of the 4th international conference on Language Resources and Evaluation (LREC2004)*, pages 1201–1204.

Lindén, K., Silfverberg, M., and Pirinen, T. (2009). HFST tools for morphology – an efficient open-source package for construction of morphological analyzers. In Cerstin Mahlow et al., editors, *State of the Art in Computational Morphology*, volume 41 of *Communications in Computer and Information Science*, pages 28–47. Springer

Berlin Heidelberg.

Nothman, J., Curran, J. R., and Murphy, T. (2008). Transforming Wikipedia into Named Entity Training Data. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 124–132.

Novák, A. (2014). A New Form of Humor – Mapping Constraint-Based Computational Morphologies to a Finite-State Representation. In Nicoletta Calzolari, et al., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1068–1073, Reykjavik, Iceland, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1207.

Novák, A. (2003). Milyen a jó Humor? [What is good Humor like?]. In *I. Magyar Számítógépes Nyelvészeti Konferencia [First Hungarian conference on computational linguistics]*, pages 138–144, Szeged. SZTE.

Novák, A. (2015). *A Model of Computational Morphology and its Application to Uralic Languages*. Ph.D. thesis, Roska Tamás Doctoral School of Sciences and Technology, Pázmány Péter Catholic University.

Orosz, Gy. and Novák, A. (2013). PurePos 2.0: a hybrid tool for morphological disambiguation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, pages 539–545, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Orosz, Gy. (2015). *Hybrid algorithms for parsing less-resourced languages*. Ph.D. thesis, Roska Tamás Doctoral School of Sciences and Technology, Pázmány Péter Catholic University.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440.

Prószéky, G. and Kis, B. (1999). A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 261–268, Stroudsburg, PA, USA. Association for Computational Linguistics.

Recski, G. and Varga, D. (2010). A Hungarian NP Chunker. *The Odd Yearbook. ELTE SEAS Undergraduate Papers in Linguistics*, pages 87–93.

Simon, E. (2013). *Approaches to Hungarian Named Entity Recognition*. Ph.D. thesis, PhD School in Cognitive Sciences, Budapest University of Technology and Economics.

Szántó, Zs. and Farkas, R. (2014). Special techniques for constituent parsing of morphologically rich languages. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 135–144, Gothenburg, Sweden, April. Association for Computational Linguistics.

Szarvas, G., Farkas, R., Felföldi, L., Kocsor, A., and Csirik, J. (2006). A highly accurate Named Entity corpus for Hungarian. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, pages 1957–1960. ELRA.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In Dan Roth et al., editors, *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Trón, V., Halácsy, P., Rebrus, P., Rung, A., Vajda, P., and Simon, E. (2006). Morphdb.hu: Hungarian lexical database and morphological grammar. In *Proceedings of LREC 2006*, pages 1670–1673.

Vincze, V., Szauter, D., Almási, A., Móra, Gy., Alexin, Z., and Csirik, J. (2010). Hungarian Dependency Treebank. In *Proceedings of LREC 2010*, Valletta, Malta, May. ELRA.

Vincze, V., Simkó, K., Szántó, Z., and Farkas, R. (2017). Universal Dependencies and Morphology for Hungarian - and on the Price of Universality. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 356–365, Valencia, Spain, April. Association for Computational Linguistics.

Zsibrita, J., Vincze, V., and Farkas, R. (2013). magyarlanc: A toolkit for morphological and dependency parsing of Hungarian. In *Proceedings of RANLP*, pages 763–771.