Advancing Hungarian Text Processing with HuSpaCy: Efficient and Accurate NLP Pipelines

György Orosz(✉), Gergő Szabó, Péter Berkecz, Zsolt Szántó, and Richárd Farkas

Institute of Informatics, University of Szeged, 2. Árpád tér, Szeged, Hungary
{gszabo,berkecz,szantozs,rfarkas}@inf.u-szeged.hu,
gyorgy@orosz.link

Abstract. This paper presents a set of industrial-grade text processing models for Hungarian that achieve near state-of-the-art performance while balancing resource efficiency and accuracy. Models have been implemented in the spaCy framework, extending the HuSpaCy toolkit with several improvements to its architecture. Compared to existing NLP tools for Hungarian, all of our pipelines feature all basic text processing steps including tokenization, sentence-boundary detection, part-of-speech tagging, morphological feature tagging, lemmatization, dependency parsing and named entity recognition with high accuracy and throughput. We thoroughly evaluated the proposed enhancements, compared the pipelines with state-of-the-art tools and demonstrated the competitive performance of the new models in all text preprocessing steps. All experiments are reproducible and the pipelines are freely available under a permissive license.

Keywords: Hungarian NLP · spaCy · PoS tagging · lemmatization · dependency parsing · named entity recognition

1 Introduction

Academic research in natural language processing has been dominated by end-to-end approaches utilizing pre-trained large neural language models which are fine-tuned for the particular applications. Although these deep learning solutions are highly accurate, there is an important demand for human-readable output in real-world language processing systems. Industrial applications are frequently fully or partially rule-based solutions, as (sufficient) training data for a pure machine learning solution is not available and each and every real-world application has its own requirements. Moreover, rule-based components provide tight control over the behavior of the systems in contrast to other approaches.

In this paper, we present improvements to a Hungarian text preprocessing toolkit that achieve competitive accuracies compared to the state-of-the-art results in each text processing step. An important industrial concern about large language models is the computational cost, which is usually not worth the accuracy gain. Transformer-based language models require far more computational resources than static word vectors,

G. Orosz, G. Szabó and P. Berkecz—These authors contributed equally to this work.

and their running costs are typically orders of magnitude higher. Furthermore, practical NLP solutions using large language models often only outperform more lightweight systems by a small margin.

In this work, we focus on text processing pipelines that are controllable, resource-efficient and accurate. We train new word embeddings for cost-effective text processing applications and we provide four different sized pipelines, including transformer-based language models, which enable a trade-off between the running costs and accuracy for practical applications. To make our pipelines easily controllable, we implement them in the spaCy[1] framework [9] by extending HuSpaCy [22] with new models.

2 Background

2.1 Specification for Language Processing Pipelines for Industrial Use

Text processing tools providing representation for hand-crafted rule construction should consist of tokenization, sentence splitting, PoS tagging, lemmatization, dependency parsing, named entity recognition and word embedding representation. These solutions have to be accurate enough for real-world scenarios while they should be resource-efficient at the same time. Last but not least, modern NLP applications are usually multilingual and should quickly transfer to a new language. This can be provided by relying on international annotation standards and by the integration into multilingual toolkits.

2.2 Annotated Datasets for Preprocessing Hungarian Texts

According to Simon et al. [24], Hungarian is considered to be one of the best supported languages for natural language processing. In 2004, the Szeged Corpus [4] was created, comprising 1.2 million manually annotated words for part-of-speech tags, morphological descriptions, and lemmata. Subsequently, these annotations were extended [5] with dependency syntax annotations. In 2017, a small section of the corpus was manually transcribed to be a part of the Universal Dependencies (UD) project [18]. Around the same time, the entire corpus was automatically converted from the original codeset to the universal part-of-speech and morphological descriptions [32].

Szeged NER [29], developed in 2006, was the first Hungarian named entity recognition corpus, consisting of 200,000 words of business and criminal news. In recent years, NYTK-NerKor [26] extended the possibilities of training and benchmarking entity recognition systems for Hungarian with a one million word multi-domain corpus.

2.3 Multilingual NLP Toolkits

Thanks to the UD project, it is now possible to easily construct multilingual NLP pipelines. Among the most commonly utilized toolkits are UDPipe [28], Stanza [23], UDify [12], Trankit [30] and spaCy.

[1] https://spacy.io/.

On the one hand, these systems exhibit a high degree of algorithmic diversity. They can be classified into two distinct groups based on their utilization of neural networks. UDPipe, spaCy and Stanza apply older, but faster architectures built on word embeddings employing convolutional and recurrent layers, respectively. On the contrary, UDify and Trankit leverage transformer-based large language models, with the former using multilingual `BERT` [6] while the latter utilizing `XLM-RoBERTa-large` [3].

On the other hand, these frameworks are typically limited by the fact that they rely solely on the Universal Dependencies datasets, which may present a disadvantage in languages such as Hungarian, which have large corpora incompatible with UD. Each of the above-mentioned systems shares this limitation, moreover, spaCy does not offer a Hungarian model at all, due to the restrictive license of the UD-Hungarian corpus. Regarding named entity annotations, Stanza is the only tool supporting NER for Hungarian.

2.4 Hungarian Language Processing Tools

The landscape of the Hungarian text processing systems was similar to that of English before the "industrial NLP revolution". There were a number of standalone text analysis tools [24] capable of performing individual text processing tasks, but they often did not work well with each other.

There were only two Hungarian pipelines that try to serve industrial needs. One of them, `magyarlanc` [33], was designed for industrial applications offering several desirable features such as software quality, speed, memory efficiency, and customizability. However, despite being used in commercial applications in the real world, it has not been maintained for several years and lacks integration with the Python ecosystem. The other pipeline, called `emtsv` [11,25,31], aimed to integrate existing NLP toolkits into a single application, but neither computational efficiency nor developer ergonomics were the main goals of the project. Additionally, while `magyarlanc` natively uses the universal morphosyntactic features, `emtsv` can only do this through conversion. Both pipelines use dependency annotation that is incompatible with Universal Dependencies, furthermore, none of them can utilize word embeddings or large language models, which have become increasingly important in recent years.

In contrast, the development of HuSpaCy placed emphasis not only on accuracy, but also on software ergonomics, while also adhering to the international standards established by Nivre et al. [18]. Moreover, it is built on spaCy, enabling users to access its full functionality with ease. One significant drawback of this tool is the lack of precise annotations for lemmata, entities and dependencies syntax.

To fulfill the industrial requirements of text processing pipelines, this work is built on the Universal Dependencies annotation schema and our models are implemented in spaCy by extending HuSpaCy's text processing model. The detailed documentation, intuitive API, high speed and accuracy of these tools make them an optimal choice for building high-performing NLP models. Additionally, HuSpaCy utilizes non UD compatible corpora as well, which allows for a comprehensive analysis of Hungarian texts.

3 Methods

3.1 HuSpaCy's Internals

HuSpaCy's main strength lies in the clever usage of available Hungarian linguistic resources and its multi-task learning capabilities inherited from spaCy. Its machine learning approach can be summarized as "embed, encode, attend, predict" shown in Fig. 1 and detailed by [10,22]. Tokens are first embedded through the combination of lexical attributes and word vectors, then context encoding is performed by stacked CNN [13] layers[2]. Finally, task specific layers are used parallelly in a multi-task learning setup.

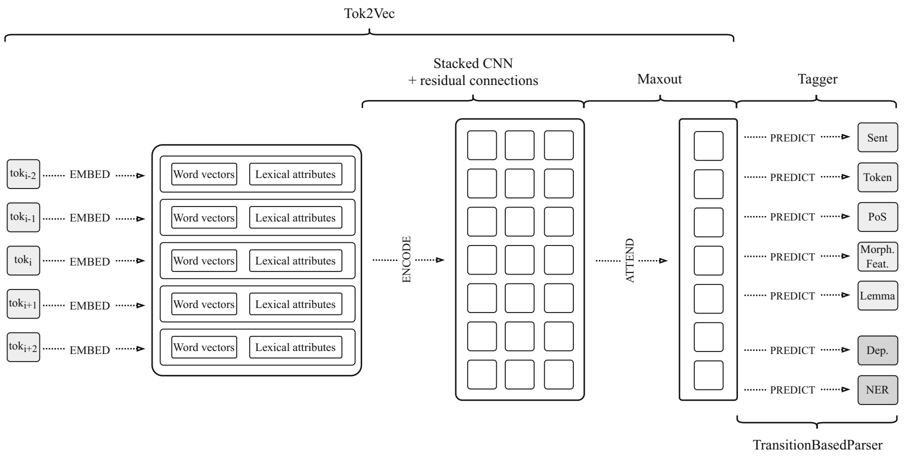

Fig. 1. The "embed, encode, attend, predict" architecture of spaCy

Orosz et al. [22] used a three step approach for fully utilizing annotated Hungarian datasets. First, they pre-train the tagger, the lemmatizer and the sentence boundary detection components on a silver standard UD annotated corpus (cf. [32]). Then, the Tok2Vec layers of this model are reused by both the NER and the parsing components: the dependency parser and the morphosyntactic taggers are fine-tuned on the UD-Hungarian dataset, the lemmatizer is trained on the entire Szeged Corpus, while the entity recognizer is further trained on the combination of the NYTK-NerKor and the Szeged NER datasets.

3.2 Improving on the Underlying Language Models

HuSpaCy' s model is built on `word2vec` [14] word embeddings, which are known to have limitations in providing meaningful representations for out-of-vocabulary words.

[2] These steps are usually referred to as the Tok2Vec layers.

This is particularly problematic for morphology-related tasks in agglutinative languages. To enhance this simple approach, a more fine-grained method that uses subword embeddings can be employed. `fastText` [2] is a widely-used extension of `word2vec` that learns sub-token embeddings. In this study, we utilized `floret`[3] which is a spaCy-compatible fork of `fastText`. To train new word vectors, we used the Hungarian Webcorpus 2.0 [17]. Two sets of word embeddings were constructed: a 100-dimensional and a 300-dimensional one.

In recent years, there has been a growing interest in transformer-based large language models (LLM), as evidenced by their high performance in text processing models (e.g. [8, 17]). With the advent of spaCy' s native support for such architectures and the availability of pre-trained language models for Hungarian, it is now possible to train transformer-based NLP pipelines for Hungarian. Our research is based on two widely used LLMs that provide support for Hungarian. One of these is `huBERT` [17], which has a `BERT`-like architecture and was trained using monolingual data. The other model is `XLM-RoBERTa-large`, which has a much larger capacity compared to the former model and was trained on multilingual corpora.

3.3 Pipeline Component Enhancements

In addition to the use of more powerful language models, we propose fundamental changes to the lemmatization and dependency parsing models, as well as minor improvements to the entity recognizer.

HuSpaCy' s lemmatizer has been replaced by a new edit-tree-based architecture, recently available in the spaCy framework[4]. This new model builds on the foundations laid out by Müller et al. [15] (called the Lemming model), but has minor differences from it. On the one hand, this reimplementation fully utilizes the framework's multi-task learning capabilities, which means that the lemmatizer is not only co-trained with PoS and morphological tagging, but also with sentence boundary detection. On the other hand, spaCy' s version lacks standard support for morphological lexicons which Lemming benefited from.

We have improved this model in two steps. 1. A simple dictionary learning method is put in place to memorize frequent *(token, tag, lemma)* triplets of the training data which are then used at prediction time to retrieve the roots of words. 2. A common weakness of Hungarian lemmatization methods is addressed. Computing the lemmata of sentence-starting tokens can be challenging for non-proper nouns, as their roots are always lowercase. Thus, we force the model to use the true casing of such words. For example, when computing the root of the sentence starting *Ezzel* 'with this' token, our method checks its PoS tag (that is ideally PRON) first, so that it can use the lowercase wordform for generating and looking up edit-trees.

Moving on, the dependency syntax annotation component is replaced with a model that has higher accuracy for many languages. Although spaCy' s built-in transition-based parser [10] has high throughput, it falls short on providing accurate predictions. Graph-based architectures are known to have good performance for dependency parsing (e.g. [1]), making such methods good enhancement candidates. Furthermore, a

[3] https://explosion.ai/blog/floret-vectors.

[4] https://explosion.ai/blog/edit-tree-lemmatizer.

spaCy-compatible implementation of Dozat and Manning's model [7] (referred to as the Biaffine parser) has recently been made available, thus we could easily utilize it in our experiments.

Finally, the named entity recognizer has been fine-tuned to provide more accurate entity annotations. This was primarily achieved by using beam-search in addition to the transition-based NER module.

4 Experiments and Results

This section presents the results of several experiments that demonstrate the improvements of our changes and show competitive results compared to well-established baselines. We evaluated pipelines developed on datasets used by the creators of HuSpaCy: the Hungarian part of the Universal Dependencies corpus[5] was utilized to benchmark the sentence boundary detector, the lemmatizer, the PoS and morphological taggers, and the dependency parser, while the entity recognizer is benchmarked on the combination of the NYTK-NerKor and the Szeged NER corpora (similar to [22] and [27]). To account for the instability of spaCy's training process we report the maximum result of three independent runs.

4.1 Evaluation of Architecture Improvements

The lemmatization accuracy of the original model has been greatly improved through a number of steps discussed in Sect. 3.3. As evidenced in Table 1, incorporation of the new neural architecture along with sub-word embeddings produced significant improvements. Furthermore, changing the default behavior of the edit-tree lemmatizer by allowing it to evaluate more than one candidate (see the row `topk=3`) also resulted in a slightly better performance. In addition, the integration of true-casing led to a considerable improvement, and the use of lemma dictionaries also significantly improved lemmatization scores.

Table 1. Lemmatization accuracy on the UD-Hungarian test set of different ablation settings. Rows marked with a "+" indicate a new feature added on top of the previous ones. `topk` is a hyperparameter of the lemmatization model controlling the number of edit-trees considered to be evaluated.

	Lemma Accuracy
HuSpaCy	95.53%
+ Edit-tree lemmatizer	95.90%
+ `floret` 300d vectors	96.76%
+ `topk=3`	97.01%
+ True-casing	97.30%
+ Learned dictionary	97.58%

[5] Experiments are performed at the v2.10 revision.

Entity recognition tasks often encounter a challenge in the form of a considerable number of out-of-vocabulary tokens, leading to decreased performance. However, the utilization of `floret` vectors has proven to be effective in addressing this issue, as indicated by the results in Table 2. Additionally, the use of beam search allowed the model to take prediction history into account, which slightly improved its efficiency.

Table 2. Evaluation of the entity recognition model improvements on the combination of the Szeged NER and NYTK-NerKor corpora. The rows starting with "+" signify the inclusion of a new feature in addition to the existing ones.

	NER F_1-score
HuSpaCy	83.68
+ `floret` 300d vectors	85.53
+ Beam search	85.99

The results in Table 3 indicate that the improved text representations and the new parsing architecture offer substantial improvements over HuSpaCy' s outcomes. However, it is worth noting that spaCy's CNN-based base model is not fully compatible with the Biaffine parser's architecture. Therefore, parsing improvements were benchmarked on top of a transformer-based encoder architecture using `huBERT`. The results show that the use of `floret` vectors is beneficial to predict morphosyntactic characteristics and dependency relations, while the use of `huBERT`-based text representations substantially improves performance across all subtasks. Furthermore, the Biaffine parser significantly outperforms its transition-based counterpart, as evidenced by its better attachment scores.

Table 3. Evaluation of text parsing improvements on the UD-Hungarian test set. "+" indicate a new feature added on top of the existing ones.

	PoS Acc.	Morph. Acc.	UAS	LAS
HuSpaCy	96.58%	93.23%	79.39	74.22
HuSpaCy + `floret` 300d vectors	96.55%	93.93%	80.36	74.89
HuSpaCy + `huBERT`	98.10%	96.97%	89.95	83.94
+ Biaffine parser	98.10%	96.97%	90.31	87.23

4.2 Comparison with the State-of-the-Art

In addition to parsing and tagging correctness, resource consumption is an important consideration for industrial NLP applications. Therefore, following the approach of Orosz et al. [22] we conducted a benchmark study to compare both the accuracy and memory usage as well as the throughput of our models with text processing tools available for Hungarian.

Table 4. Text parsing accuracy of the novel pipelines compared to HuSpaCy, Stanza, UDify, Trankit and `emtsv`. Results for non-comparable models are shown in italics.

	Sent. F_1-score	PoS Acc.	Morph. Acc.	Lemma Acc.	UAS	LAS	NER F_1-score
`emtsv`	*98.11*	*89.19%*	*87.95%*	*96.16%*	–	–	**92.99**
Trankit	*98.00*	*97.49%*	*95.23%*	*94.45%*	***91.31***	***87.78***	–
UDify	–	96.15%	90.54%	88.70%	88.03	83.92	–
Stanza	97.77	96.12%	93.58%	94.68%	84.05	78.75	83.75
HuSpaCy	97.54	96.58%	93.23%	95.53%	79.39	74.22	83.68
`md`	97.88	96.26%	93.29%	97.38%	79.25	73.99	85.35
`lg`	98.33	96.91%	93.93%	97.58%	79.75	74.78	85.99
`trf`	99.33	**98.10%**	**96.97%**	98.79%	90.31	87.23	91.35
`trf_xl`	**99.67**	97.79%	96.53%	**98.90%**	90.22	86.67	91.84

Table 5. Resource usage (All benchmarks are run on the same environment having AMD EPYC 7F72 CPUs and NVIDIA A100 GPUs) of the new models and state-of-the-art of text processing tools available for Hungarian. Throughput is measured as the average number of processed tokens per second, while memory usage columns records the peak value of each tool.

	Throughput		Memory Usage (GB)
	CPU	GPU	
`emtsv`	113	–	3.9
Trankit	434	2119	3.7
UDify	129	475	3.2
Stanza	30	395	5.3
HuSpaCy	1525	6697	3.5
`md`	2652	3195	1.4
`lg`	847	3128	3.2
`trf`	273	2605	4.8
`trf_xl`	82	2353	18.9

First of all, an important result of this study is a base model (referred to as `lg`), which achieves a good balance between accuracy and resource usage as seen in Tables 4 and 5. This pipeline is built on top of the 300d `floret` vectors and incorporates all the enhancements described above, except for the new parser. Evaluation data demonstrates that the `lg` pipeline consistently outperforms Stanza in all tasks except syntactic dependency relation prediction, which can be explained by the superior parsing model of the latter tool.

We present the results of a medium-sized model (`md`) as well that is a reduced version of the `lg` pipeline utilizing the smaller (100d) word embeddings. Surprisingly, the `md` pipeline delivers performance similar to that of the larger model. Furthermore, the medium-sized model achieves scores comparable to or higher than those of HuSpaCy, despite requiring half the memory and exhibiting much higher throughput on CPU.

Transformer-based pipelines using the graph based dependency parser have the highest scores across all language analysis tasks. Remarkably, despite its smaller capacity, the model based on `huBERT` (`trf`) achieves the highest attachment scores for dependency parsing, while the one using `XLM-RoBERTa-large` (`trf_xl`) provides slightly more accurate PoS tags and named entities.

It is important to consider that not all third-party pipelines in Table 4 are directly comparable to our results, due to differences in the versions of the UD-Hungarian dataset used to train and evaluate their models. To ensure a fair comparison, Stanza and UDify have been retrained. On the other hand, we obtained the results of Trankit from [30] since it would be a demanding task to fine-tune this model. Furthermore, the results of `emtsv`' s text parsing components [19–21] cannot be deemed reliable either (cf. [22]), since its components use a different train-test split of the Szeged Corpus. However, this tool's entity recognition module (`emBERT` [16]) was evaluated by Simon et al. [27] using the same settings as in our paper, thus we rely on their assessment. Additionally, state-of-the-art results are also shown in Table 4. With regard to highest dependency parsing scores, the results of the multilingual Trankit system are produced by a parsing model similar to that of ours. As for named entity recognition, `emBERT` attains the best F_1 scores by utilizing a Viterbi encoder that eliminates invalid label sequences from the outputs of the underlying model.

Regarding computational requirements, Table 5 presents findings that demonstrate how `floret` embeddings can effectively decrease the memory usage of models without compromising their accuracy and throughput. However, it is apparent that enhancing pipeline accuracy frequently results in slower processing speed, as can be observed from the `lg`, `trf` and `trf_xl` models. Additionally, our tests also showed that most of the readily available NLP pipelines are not adequately optimized to handle large workloads, which is evident from their low throughput values.

5 Conclusion

This paper has introduced new industrial-grade text processing pipelines for Hungarian and presented a thorough evaluation showing their (close to) state-of-the-art performance. We have shown that new architectures for lemmatization and dependency parsing and the use of improved text representation models significantly improve the accuracy of HuSpaCy. The presented models have not only demonstrated high performance in all text preprocessing steps, but the resource consumption of three of our models' (`md`, `lg`, `trf`) makes them suitable for solving practical problems. All of our experiments are reproducible and the models are freely available under a permissive license.

For future work, we consider the following areas of improvement. 1. Transformer-based pipelines are optimized for accuracy but this could limit their usability due to reduced computational efficiency. We would like to investigate optimizing their size to enhance their resource usage. 2. We would like to include more silver standard data to further improve the parsing and tagging scores, as the corpus used to train and evaluate text parsing components is limited in size. 3. Our models are mostly trained on news-related corpora, which makes user-generated text processing a difficult task. In order to address this challenge, we intend to integrate automatic data augmentation into the training process as a solution.

Acknowledgments. The authors would like to thank Gábor Berend for his valuable suggestions. HuSpaCy research and development is supported by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory.

References

1. Altıntaş, M., Tantuğ, A.C.: Improving the performance of graph based dependency parsing by guiding bi-affine layer with augmented global and local features. Intell. Syst. Appl. **18**, 200190 (2023)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
3. Conneau, A., et al.: Unsupervised cross-lingual representation learning at scale. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8440–8451 (2020)
4. Csendes, D., Csirik, J., Gyimóthy, T.: The szeged corpus: a POS tagged and syntactically annotated Hungarian natural language corpus. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2004. LNCS (LNAI), vol. 3206, pp. 41–47. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30120-2_6
5. Csendes, D., Csirik, J., Gyimóthy, T., Kocsor, A.: The szeged treebank. In: Matoušek, V., Mautner, P., Pavelka, T. (eds.) TSD 2005. LNCS (LNAI), vol. 3658, pp. 123–131. Springer, Heidelberg (2005). https://doi.org/10.1007/11551874_16
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, pp. 4171–4186 (2019)
7. Dozat, T., Manning, C.D.: Deep biaffine attention for neural dependency parsing. In: International Conference on Learning Representations (2017)
8. Enevoldsen, K., Hansen, L., Nielbo, K.: DaCy: a unified framework for Danish NLP. arXiv preprint arXiv:2107.05295 (2021)
9. Honnibal, M.: Introducing spaCy (2015). https://explosion.ai/blog/introducing-spacy
10. Honnibal, M., Goldberg, Y., Johnson, M.: A non-monotonic arc-eager transition system for dependency parsing. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning, pp. 163–172. Association for Computational Linguistics, Sofia (2013)
11. Indig, B., Sass, B., Simon, E., Mittelholcz, I., Vadász, N., Makrai, M.: One format to rule them all - the `emtsv` pipeline for Hungarian. In: Proceedings of the 13th Linguistic Annotation Workshop, pp. 155–165. Association for Computational Linguistics, Florence (2019)
12. Kondratyuk, D., Straka, M.: 75 languages, 1 model: parsing universal dependencies universally. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2779–2795 (2019)
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)

15. Müller, T., Cotterell, R., Fraser, A., Schütze, H.: Joint lemmatization and morphological tagging with lemming. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2268–2274. Association for Computational Linguistics, Lisbon (2015)
16. Nemeskey, D.M.: Egy `emBERT` próbáló feladat. In: XVI. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2020), pp. 409–418. Szeged (2020)
17. Nemeskey, D.M.: Natural language processing methods for language modeling. Ph.D. thesis, Eötvös Loránd University (2020)
18. Nivre, J., et al.: Universal dependencies v2: an evergrowing multilingual treebank collection. In: Proceedings of the Twelfth Language Resources and Evaluation Conference, pp. 4034–4043. European Language Resources Association, Marseille (2020)
19. Novák, A.: A new form of humor – mapping constraint-based computational morphologies to a finite-state representation. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014), pp. 1068–1073. European Language Resources Association (ELRA), Reykjavik (2014)
20. Novák, A., Siklósi, B., Oravecz, C.: A new integrated open-source morphological analyzer for Hungarian. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 1315–1322. European Language Resources Association (ELRA), Portorož (2016)
21. Orosz, G., Novák, A.: PurePos 2.0: a hybrid tool for morphological disambiguation. In: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013, pp. 539–545. INCOMA Ltd., Shoumen, BULGARIA, Hissar (2013)
22. Orosz, G., Szántó, Z., Berkecz, P., Szabó, G., Farkas, R.: HuSpaCy: an industrial-strength Hungarian natural language processing toolkit. In: XVIII. Magyar Számítógépes Nyelvészeti Konferencia (2022)
23. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: a Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020)
24. Simon, E., Lendvai, P., Németh, G., Olaszy, G., Vicsi, K.: A Magyar Nyelv a Digitális Korban - the Hungarian Language in the Digital Age. Georg Rehm and Hans Uszkoreit (Series Editors): META-NET White Paper Series. Springer, Heidelberg (2012)
25. Simon, E., Indig, B., Kalivoda, Á., Mittelholcz Iván, S.B., Vadász, N.: Újabb fejlemények az `e-magyar` háza táján. In: Berend, G., Gosztolya, G., Vincze, V. (eds.) XVI. Magyar Számítógépes Nyelvészeti Konferencia, pp. 29–42. Szegedi Tudományegyetem Informatikai Tanszékcsoport, Szeged (2020)
26. Simon, E., Vadász, N.: Introducing NYTK-NerKor, a gold standard Hungarian named entity annotated corpus. In: Ekštein, K., Pártl, F., Konopík, M. (eds.) TSD 2021. LNCS (LNAI), vol. 12848, pp. 222–234. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83527-9_19
27. Simon, E., Vadász, N., Lévai, D., Dávid, N., Orosz, G., Szántó, Z.: Az NYTK-NerKor több szempontú kiértékelése. XVIII. Magyar Számítógépes Nyelvészeti Konferencia (2022)
28. Straka, M.: UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pp. 197–207. Association for Computational Linguistics, Brussels (2018)
29. Szarvas, György., Farkas, Richárd, Kocsor, András: A multilingual named entity recognition system using boosting and C4.5 decision tree learning algorithms. In: Todorovski, Ljupčo, Lavrač, Nada, Jantke, Klaus P.. (eds.) DS 2006. LNCS (LNAI), vol. 4265, pp. 267–278. Springer, Heidelberg (2006). https://doi.org/10.1007/11893318_27
30. Van Nguyen, M., Lai, V., Veyseh, A.P.B., Nguyen, T.H.: Trankit: a light-weight transformer-based toolkit for multilingual natural language processing. EACL **2021**, 80 (2021)

31. Váradi, T., et al.: E-magyar - a digital language processing system. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki (2018)
32. Vincze, V., Simkó, K., Szántó, Z., Farkas, R.: 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, pp. 356–365. Association for Computational Linguistics, Valencia (2017)
33. Zsibrita, J., Vincze, V., Farkas, R.: magyarlanc: a toolkit for morphological and dependency parsing of Hungarian. In: Proceedings of Recent Advances in Natural Language Processing 2013, pp. 763–771. Association for Computational Linguistics, Hissar (2013)