Human Language Technologies – The Baltic Perspective
K. Muischnek and K. Müürisep (Eds.)

doi:10.3233/978-1-61499-912-6-166

Neural Morphological Tagging for Estonian

Alexander TKACHENKO [1] and Kairit SIRTS
Institute of Computer Science, University of Tartu, Tartu, Estonia

Abstract. We develop neural morphological tagging and disambiguation models for Estonian. First, we experiment with two neural architectures for morphological tagging – a standard multiclass classifier which treats each morphological tag as a single unit, and a sequence model which handles the morphological tags as sequences of morphological category values. Secondly, we complement these models with the analyses generated by a rule-based Estonian morphological analyser (MA) VABAMORF, thus performing a soft morphological disambiguation. We compare two ways of supplementing a neural morphological tagger with the MA outputs: firstly, by adding the combined analyses embeddings to the word representation input to the neural tagging model, and secondly, by adopting an attention mechanism to focus on the most relevant analyses generated by the MA. Experiments on three Estonian datasets show that our neural architectures consistently outperform the non-neural baselines, including HMM-disambiguated VABAMORF, while augmenting models with MA outputs results in a further performance boost for both models.

Keywords. computational morphology, morphological tagging, morphological disambiguation, neural networks, Estonian language, natural language processing

1. Introduction

We address the problem of morphological tagging of Estonian language. Existing resources for Estonian computational morphological processing include a rule-based morphological analyser VABAMORF [1] which generates all possible morphological analyses for a word. VABAMORF also includes a statistical HMM-based disambiguator which, based on the context, resolves the ambiguities for most of the words [2].

Recently, neural models have been successfully applied to both POS and morphological tagging achieving prominent results in many languages [3,4,5]. In this paper, we focus on neural morphological tagging of Estonian. Among other languages, Heigold et al. [5], and Tkachenko and Sirts [6] also report morphological tagging results for Estonian trained on Universal Dependencies (UD) corpus, reaching 94.25% and 93.30% respectively on different versions of the UD Estonian test set.

We experiment with two neural architectures which differ in the way they model morphological tags. The first architecture is a standard multiclass classifier which treats each morphological tag as a single unit. As a second architecture, we adopt the sequential architecture proposed by Tkachenko and Sirts [6] that models each morphological tag

[1]Corresponding Author: Alexander Tkachenko; E-mail: aleksandr.tkatsenko@ut.ee.

as a sequence of morphological categories. Both architectures use a bidirectional LSTM encoder to compute the representations of words and their contexts.

Morphological disambiguation is a task closely related to morphological tagging. While the goal of morphological tagging is to select the correct label from the set of all possible tags, the task of a morphological disambiguator is to select the correct analysis from a limited set of candidates provided by a morphological analyser (MA) [7,8].

The main drawback of morphological disambiguation is that it totally relies on MA. If the MA output does not contain the correct tag (or in some cases, any tag at all), the model will not be able to make the correct prediction. Furthermore, the straightforward morphological disambiguation is complicated or inapplicable in situations where the available annotated dataset and the MA use different tag annotation schemes.

To address these issues, we experiment with two ways to utilise MA outputs as additional inputs to our neural models, thus performing a soft morphological disambiguation [9]. The first approach, inspired by the one proposed by Inoue et al. [9], combines the embedded representations of the word's morphological analyses and feeds this combined representation, in addition to the representation of the word itself, as input to the neural tagging model. The second architecture again embeds the morphological analyses generated by the analyser but adopts an attention mechanism – a method commonly used in neural machine translation models [10,11] – to focus on the most relevant analyses when making the tagging decisions.

We evaluate our models on three different Estonian morphologically annotated datasets. Both neural models demonstrate high performance, reaching up to 97% of full tag accuracy and consistently outperforming both the rule-based VABAMORF with statistical disambiguation, and a strong CRF-based baseline. By further augmenting our models with the outputs of the VABAMORF MA, we achieve performance gains in the range of 0.46% to 1.15% in the absolute full tag accuracy for all models on all datasets. Furthermore, we show that incorporating the outputs of the MA is beneficial even if the analyser and the training dataset use different annotation schemes.

2. Vabamorf

We start by analysing the tagging accuracy of the VABAMORF MA. VABAMORF is a publicly available morphological analyser for Estonian language which is available on its own[2] and is also included in the EstNLTK toolkit[3] [12]. It processes text in two steps. First, it performs rule-based morphological analysis on each individual word in order to identify their possible analyses. Since very often multiple analyses are produced, as a second step, VABAMORF employs a HMM-based disambiguator to identify words' correct analyses based on the context.

However, VABAMORF is not perfect and introduces errors in both steps. According to Kaalep and Vaino [2], for ca 0.4% of words in the texts, the analyser produces a list of analyses that does not contain the correct analysis. Furthermore, for about 13.5% of the cases, the disambiguator purposely avoids resolving ambiguities and outputs multiple analyses. While such behaviour might be appropriate for a corpus linguistic study of the text, it is unsuitable when morphological disambiguation is adopted as an intermediate step in a natural language processing pipeline.

[2]https://github.com/Filosoft/vabamorf

[3]https://estnltk.github.io/estnltk

To assess the magnitude of these errors, we analysed the performance of VABAMORF on one of our experimental datasets – the Estonian morphologically disambiguated corpus (MD). The statistics of this corpus are given in Table 1. We ran VABAMORF with the guesser and proper name resolver on the whole MD dataset. Results show that for 8.4% of words, VABAMORF retains ambiguity on MD dataset.

Since VABAMORF and the MD dataset use different tagsets, it is impossible to evaluate the disambiguation performance of VABAMORF directly. Therefore, we convert VABAMORF outputs to the MD-compatible format using a set of scripts[4] that perform required transformations and on top of that run a constraint grammar parser. As a side effect, such conversion introduces additional ambiguity of 4.22% compared to the VABAMORF's original output. We further post-process the obtained output as well as the MD dataset in order to eliminate some systematic inconsistencies in annotations for pronouns, capitalised words, etc. As a result, 12.62% of words remain ambiguous, 97.07% of which have the correct analysis in the list. In the case of unambiguous words, the accuracy is 97.41%.

In order to assess the overall accuracy, we need to resolve the ambiguous cases. We implemented two simple ways to do so: by choosing either a random or the first analysis out of the list of candidates. This resulted in an overall accuracy of 89.76% and 90.29% respectively. We conclude that unresolved ambiguity significantly affects VABAMORF's overall performance and complicates its direct application to the task of morphological tagging in a NLP pipeline.

3. Morphological Tagging Models

We experiment with two different neural architectures for morphological tagging: a multiclass model which treats morphological labels as a single tag, and a sequence model which generates morphological labels as sequences of feature-value pairs. Both models utilise the general encoder-decoder neural architecture where the encoder computes the representations for the inputs and the decoder performs the labelling based on these input representations. We first describe the encoder architecture which is the same for all models. Then, we describe the decoder architectures.

Encoder We adopt a standard neural sequence tagging encoder architecture for both models. Similar encoder architectures have been applied recently with notable success to morphological tagging [5,4,6] as well as several other sequence tagging tasks [13, 14,15]. It consists of a bidirectional LSTM network that maps words in a sentence into contextual feature vectors using character and word-level embeddings. Character-level word embeddings $\mathbf{w_{char}}$ are constructed with another bidirectional LSTM network and they capture useful information about words' morphology and shape. Word-level embeddings $\mathbf{w}$ are initialised with pre-trained embeddings and fine-tuned during training. The character and word-level embeddings are concatenated and passed as inputs $\bar{\mathbf{w}}$ to the bidirectional LSTM encoder. The resulting hidden states $\mathbf{h}$ capture contextual information for each word in a sentence.

*Multiclass Model (*MC*)* The first decoder employs the standard multiclass classifier used by both Heigold et al. [5] and Yu et al. [4]. It is essentially just a softmax classifier over the full tagset. The inherent limitation of this model is the inability to predict tags that are not present in the training corpus.

[4]https://github.com/EstSyntax/EstCG/blob/master/konverter.sh

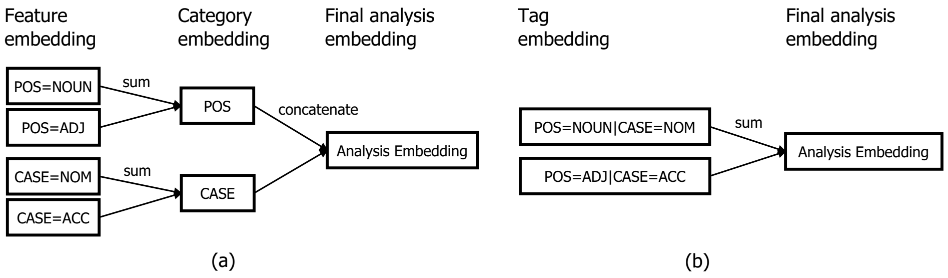

Figure 1. Constructing an analysis embedding for a word with MA analyses [POS=NOUN,CASE=GEN] and [POS=ADJ,CASE=ACC] using (a) category-based and (b) tag-based approach.

*Sequence Model (*SEQ*)* The sequence model, introduced by Tkachenko and Sirts [6], predicts complex morphological labels as sequences of category values. For each word in a sentence, the sequential decoder generates a sequence of morphological feature-value pairs based on the context vector $\mathbf{h}$ and the previous predictions. The decoder is a unidirectional LSTM network. Decoding starts by passing the start-of-sequence symbol as input. At each time step, the decoder computes the label context vector $\mathbf{g}_j$ based on the embedding of the previously predicted category value $\mathbf{f}_{j-1}$, previous label context vector $\mathbf{g}_{j-1}$ and the word's context vector $\mathbf{h}$. The probability of each morphological feature-value pair is then computed with a softmax classifier over all possible feature-value pairs. At training time, we feed correct labels as inputs while at inference time, we greedily emit the best prediction. The decoding terminates once the end-of-sequence symbol is produced.

4. Incorporating Morphological Analyser

We explore two methods to incorporate the outputs generated by the VABAMORF MA. In both cases, we encode the MA outputs using dense vector representations. We make use of the following notation. For a word w, the MA generates M distinct morphological analyses $m_1, \ldots, m_M$. Each of these morphological analyses can be split into a list of C category-value pairs $m = [f_1, \ldots, f_C]$. The embedding vectors corresponding to morphological analyses m are denoted by $\mathbf{m}$. The embedding vectors corresponding to category-value pairs f are denoted by $\mathbf{f}$.

4.1. Analysis Embeddings

The Analysis Embeddings approach is inspired by Inoue et al. [9]. Here, we directly augment the input word representations with the dense representations of the morphological analyses generated by the MA.

We examine two methods to compute the analysis embedding for a word – a tag-based method and a category-based method. In the category-based method, which was introduced by Inoue et al. [9], we decompose morphological analyses into a list of individual key-value features f (e.g. POS=NOUN) and learn vector representations $\mathbf{f}$ for each such feature. For each feature key (e.g. POS), we first compute the category embedding as a sum of the embeddings of the corresponding values contained in the MA analyses (see Figure 1 (a)). The final analysis embedding for a word is obtained by concatenating individual category embeddings in a pre-fixed order. The analysis

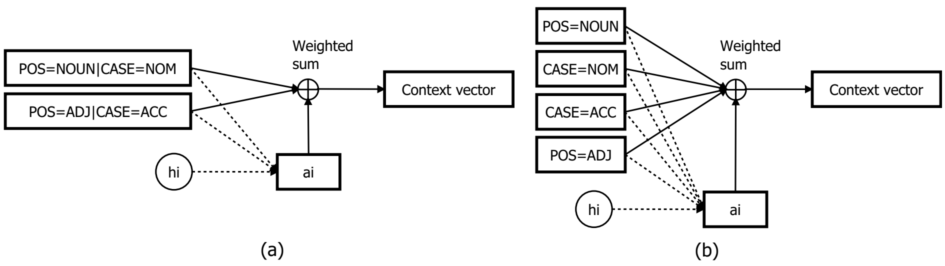

Figure 2. Constructing an attentional context vector for a word with analyses [POS=NOUN,CASE=GEN] and [POS=ADJ,CASE=ACC] using (a) tag-based and (b) category-based approach.

embedding is then concatenated to the word and character-based embedding to obtain an input representation fed into the encoder.

The category-based method assumes that the morphological tags must be given as a list of category-value pairs, or at least is should be possible to convert the tags into such format relatively easily. While this is the case for the Estonian resources we are using in this work, this assumption might not necessarily hold for morphological analysers or lexicons in other languages. Thus, for the sake generality, we also explore a simpler method that does not make any assumptions about the format of the morphological tags. In the tag-based method, we treat each MA analysis as a single combined tag m and simply encode them using dense vector representations $\mathbf{m}$. The analysis embedding for a word is then computed as a sum of individual MA tag embeddings (see Figure 1 (b)).

4.2. Attention Embeddings

In the Analysis Embeddings approach, the augmented word representations are obtained by combining all MA analyses, both correct and wrong, with equal weights. Alternatively, an attention-based method [11,10] would enable to weigh MA analyses differently, such that the most relevant analyses will have greater contribution to the final tagging decision. To the best of our knowledge, the attempt to utilise attention to guide the neural morphological tagging/disambiguation is novel.

Similarly to Analysis Embeddings method, attentional context vectors can be computed in two distinct ways – based on the full MA analysis tag and based on individual categories as illustrated in Figure 2. In the tag-based approach, the attention is computed over analysis embedding vectors $\mathbf{m}$. The unnormalised attention scores between the ith word and the jth tag are computed according to the Luong's global attention [11]. The scores are normalised via softmax and the context vector $\mathbf{c}_i$ is computed as a weighted average over the tag embedding vectors. The context vector $\mathbf{c}_i$ is concatenated with the latent feature vector $\mathbf{h}_i$, passed through a dense layer with a non-linear activation to obtain an attention-informed feature vector $\bar{\mathbf{h}}_i$. The attention-informed feature vector $\bar{\mathbf{h}}_i$ is passed through a linear layer and then through a softmax to compute the predictive probabilities of the tags. The category-based attention is applied similarly. However, instead of the analysis embeddings $\mathbf{m}$, the attention is applied over the set of distinct category-value embeddings $\mathbf{f}$ obtained from all morphological analyses generated to the word by the MA.

The application of the attention to the MA outputs differs for the MC and SEQ models. In the case of the MC model, we compute attention alignments with respect to the encoder outputs $\mathbf{h}_i$. In the case of the SEQ model, the attention is applied at each step when

Table 1. Summary statistics for our datasets.

	MD			EDT			UD		
	train	test	dev	train	test	dev	train	test	dev
Tokens	499 273	62 655	62 649	345 463	43 058	43 145	287 859	41 273	37 219
Types	77 275	19 354	18 982	68 714	15 949	16 000	58 942	12 613	12 972
Tags	385	299	315	719	471	474	918	559	580

generating a label to a word. Specifically, we compute the alignments with respect to MA analyses at each decoding step when predicting each morphological category-value pair.

5. Experimental Setup

Datasets We evaluate our models on three datasets: Morphologically Disambiguated corpus (MD)[5], Estonian Dependency Treebank (EDT)[6] and Estonian Universal Dependencies version 2.2 (UD)[7]. We transform morphological annotations in the MD and the EDT datasets into a UD-like key-value format by organising individual morphological attributes into groups of morphological categories. For model training and evaluation purposes, we split the MD and EDT datasets sentence-wise into training, dev and test sets with a ratio of 80%, 10% and 10%, while the UD dataset comes with the pre-defined split. Table 1 provides summary statistics for all datasets.

Vabamorf We use VABAMORF morphological analyser to obtain candidate analyses for soft neural disambiguation. We run VABAMORF with a guesser, proper name resolver and disambiguator. Since some of our models utilise VABAMORF's outputs on category-level, we convert them to the key-value format as explained above.

Word Embeddings We use *fastText* word embeddings[8] [16]. The dataset contains 329 987 300-dimensional word vectors trained on Estonian Wikipedia. Although these embeddings are uncased, our model still captures case information by means of character-level embeddings. Words in a training set with no pre-trained embeddings are initialised with random embeddings. At test time, words with no pre-trained embedding are assigned a special UNK-embedding. We train the UNK-embedding by randomly substituting singletons in a batch with the UNK-embedding with a probability of 0.5.

Training and Parametrisation We train all neural models using stochastic gradient descent for up to 400 epochs and stop early if there has been no improvement on development set within 50 epochs. The batch size is set to 5 for SEQ models and 20 for MC models. The initial learning rate is set to 1 for both models. For MC model, we decay the learning rate by a factor of 0.98 after every 2500 batch updates. We represent input words with 300-dimensional word-level embeddings and 150-dimensional character-level embeddings. The encoder employs a single-layered biLSTM with a 400-dimensional hidden state. We apply a dropout of 0.5 on both input layer and encoder outputs. In the Analysis Embeddings approach, we use 50-dimensional tag-embeddings and 10-

[5]http://www.cl.ut.ee/korpused/morfkorpus
[6]https://github.com/EstSyntax/EDT
[7]https://github.com/UniversalDependencies/UD_Estonian-EDT/
[8]https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Table 2. Morphological tagging accuracies for our neural models (a) and baselines (b). The best results are marked in bold.

(a) Neural models

MODEL	MD	EDT	UD
MC	97.00	95.60	94.43
MC+EMB-TAG	98.02	96.01	95.01
MC+EMB-CAT	**98.03**	**96.06**	95.02
MC+ATN-TAG	97.80	95.61	94.59
MC+ATN-CAT	97.82	95.61	94.72
SEQ	96.85	95.37	94.32
SEQ+EMB-TAG	98.00	95.92	**95.19**
SEQ+EMB-CAT	97.96	95.93	**95.19**
SEQ+ATN-TAG	97.94	95.75	94.78
SEQ+ATN-CAT	97.95	95.67	94.71

(b) Baselines

MODEL	MD	EDT	UD
VABAMORF	90.22	79.17	-
MARMOT	95.23	93.01	92.03
MC [5]	-	-	94.25
MC [6]	-	-	93.28
SEQ [6]	-	-	93.30

dimensional category-embeddings, while in the Attention Embeddings approach, both category- and tag-embeddings are 50-dimensional. We initialise biases with zeros and parameter matrices using Xavier uniform initialiser.

6. Results

Table 2 (a) reports full-tag accuracies on the MD, EDT and UD test sets. In addition to our neural models, we also report results for VABAMORF MA[9] and for MARMOT, a CRF-based morphological tagger [17] (b). We also include recent results by Heigold et al. [5] and Tkachenko and Sirts [6] which were, however, obtained on older and smaller versions of the UD dataset, and thus are not directly comparable to ours. Results indicate that both our basic neural models – MC and SEQ – outperform MARMOT and VABAMORF baselines[10] by a large margin while the MC model demonstrates consistent improvement over the SEQ model on all datasets. The latter finding contradicts Tkachenko and Sirts [6] who argue in favour of the SEQ model. This can be explained by the fact that in [6], the authors train models using a significantly smaller dataset which contains a larger number of infrequent and OOV tags to which the MC model is especially susceptible.

Incorporating MA outputs further boosts performance for both models from 0.46% (MC-EMB-CAT on EDT) to 1.15% (SEQ-EMB-TAG on MD). We hypothesised that Attention Embeddings would perform better than Analysis Embeddings because by using Attention Embeddings, the model can learn different weights to input analyses depending on context. Interestingly, this hypothesis did not hold and the Analysis Embeddings method performs consistently better in all cases. The difference between the category- and tag-based approach is, however, negligible, suggesting that it is not necessary to convert the MA analyses into category-value format.

[9]Ambiguous words are resolved by always picking the first analysis.

[10]The UD annotation is very different from the annotation used by VABAMORF and thus a reliable evaluation of VABAMORF on UD corpus could not be made.

Table 3. Performance of MC and MC+EMB-CAT models on individual features reported as macro-averaged F1-scores. Table rows are sorted by feature frequency in the MD dataset.

	MD			EDT			UD		
Feature	MC	+EMB-CAT	diff	MC	+EMB-CAT	diff	MC	+EMB-CAT	diff
POS	93.34	95.12	+1.78	90.54	91.17	+0.63	87.83	88.18	+0.35
Number	98.73	99.16	+0.43	98.42	98.86	+0.44	98.06	98.52	+0.46
Case	96.05	98.31	+2.26	95.25	95.52	+0.27	94.46	95.26	+0.80
NounType	97.38	98.40	+1.02	97.15	97.62	+0.47	-	-	-
Punct	100.00	100.00	0.00	96.49	96.60	+0.11	-	-	-
VerbType	96.96	97.39	+0.43	97.26	97.37	+0.11	-	-	-
Voice	97.62	98.02	+0.40	96.33	96.80	+0.47	98.68	98.88	+0.20
Tense	97.44	97.56	+0.12	96.82	97.10	+0.28	93.62	93.70	+0.08
Polarity	99.03	99.29	+0.26	99.15	99.32	+0.17	99.30	99.22	-0.08
Mood	95.93	99.07	+3.14	93.64	95.80	+2.16	87.82	89.99	+2.17
Person	97.65	98.56	+0.91	97.22	97.80	+0.58	96.04	96.06	+0.02
Degree	96.98	98.41	+1.43	95.90	96.02	+0.12	95.39	95.93	+0.54
VerbForm	97.41	98.07	+0.66	96.82	97.00	+0.18	98.53	98.59	+0.06
NumForm	97.33	97.91	+0.58	89.68	91.94	+2.26	84.20	87.08	+2.88
NumType	97.87	98.53	+0.66	95.59	96.22	+0.63	91.25	91.69	+0.44
Abbr	95.78	95.34	-0.44	97.58	98.22	+0.64	89.94	89.58	-0.36
PronType	-	-	-	95.35	97.28	+1.93	89.97	89.25	-0.72
AdpType	-	-	-	-	-	-	89.28	88.94	-0.34
Connegative	-	-	-	-	-	-	99.10	98.76	-0.34
Poss	-	-	-	-	-	-	95.87	95.44	-0.43
Reflex	-	-	-	-	-	-	93.68	92.55	-1.13
Hyph	-	-	-	-	-	-	99.03	98.00	-1.03
Average	97.22	98.07		95.83	96.51		93.79	93.98	

Table 3 compares the performance of the MC and MC+EMB-CAT models on individual features. On MD and EDT corpora, the prediction accuracy improves for all categories. The only exception is the abbreviation category, for which the F1-score drops a little. For MD corpus, adding MA outputs helps most to improve the prediction performance of MOOD, CASE and DEGREE categories. The accuracy of the POS as well as the NOUNTYPE (which essentially also encodes POS information) also improves considerably. For EDT corpus, the categories NUMFORM, MOOD and PRONTYPE improve the most. The UD corpus gains the least from the addition of MA outputs. In fact, the macro-averaged F1-scores of 8 categories actually decrease. One reason for these results might be that the UD annotations differ the most from VABAMORF annotations. More detailed analysis of these results is left for future work. However, we note that even supplying MAs with very different annotations, the neural tagger is able to improve the overall tagging performance on the UD corpus as well.

7. Conclusion

In this work, we focused on Estonian morphological tagging using neural sequence tagging framework. We experimented with two neural architectures which differ in the way they model morphological tags. Our results show that both neural models demonstrate high performance, consistently outperforming both rule-based VABAMORF and a CRF-based baseline on three datasets. We further explored multiple ways to utilise word analyses produced by a morphological analyser as additional inputs to our neural models. We found that in all cases, the augmented models achieved a consistently superior performance on our datasets, including UD which uses a considerably different tagset than the VABAMORF MA. The highest improvement for both models was achieved by incorporating morphological analyser outputs in the form of category-level embeddings as inputs to the tagger encoder.

Acknowledgments

This work was supported by the Estonian Research Council (grants no. 2056 and 1226).

References

[1] H.-J. Kaalep, "An Estonian morphological analyser and the impact of a corpus on its development," *Computers and the Humanities*, vol. 31, no. 2, pp. 115–133, 1997.

[2] H.-J. Kaalep and T. Vaino, "Complete morphological analysis in the linguist's toolbox," *Congressus Nonus Internationalis Fenno-Ugristarum Pars V*, pp. 9–16, 2001.

[3] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," in *Proceedings of ACL*, vol. 2, pp. 412–418, 2016.

[4] X. Yu, A. Falenska, and N. T. Vu, "A general-purpose tagger with convolutional neural networks," in *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pp. 124–129, 2017.

[5] G. Heigold, G. Neumann, and J. van Genabith, "An extensive empirical evaluation of character-based morphological tagging for 14 languages," in *Proceedings of EACL*, vol. 1, pp. 505–513, 2017.

[6] A. Tkachenko and K. Sirts, "Modeling composite labels for neural morphological tagging," in *Proceedings of CoNLL*, 2018.

[7] Q. Shen, D. Clothiaux, E. Tagtow, P. Littell, and C. Dyer, "The role of context in neural morphological disambiguation," in *Proceedings of COLING*, pp. 181–191, 2016.

[8] N. Zalmout and N. Habash, "Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic," in *Proceedings of EMNLP*, pp. 704–713, 2017.

[9] G. Inoue, H. Shindo, and Y. Matsumoto, "Joint prediction of morphosyntactic categories for fine-grained arabic part-of-speech tagging exploiting tag dictionary information," in *Proceedings of CoNLL*, pp. 421–431, 2017.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[11] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of EMNLP*, pp. 1412–1421, Association for Computational Linguistics, 2015.

[12] S. Orasmaa, T. Petmanson, A. Tkachenko, S. Laur, and H.-J. Kaalep, "EstNLTK - NLP toolkit for Estonian," in *Proceedings of LREC*, (Paris, France), 2016.

[13] G. Lample, M. Ballesteros, K. Kawakami, S. Subramanian, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of NACL*, 2016.

[14] J. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *TACL*, vol. 4, pp. 357–370, 2016.

[15] W. Ling, C. Dyer, A. W. Black, I. Trancoso, R. Fermandez, S. Amir, L. Marujo, and T. Luis, "Finding function in form: Compositional character models for open vocabulary word representation," in *Proceedings of EMNLP*, pp. 1520–1530, Association for Computational Linguistics, 2015.

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *TACL*, vol. 5, pp. 135–146, 2017.

[17] T. Müller, H. Schmid, and H. Schütze, "Efficient higher-order CRFs for morphological tagging," in *Proceedings of EMNLP*, pp. 322–332, 2013.