Initial Experiments in Data-Driven Morphological Analysis for Finnish

Miikka Silfverberg
University of Colorado Boulder
Department of Linguistics
miikka.silfverberg@colorado.edu

Mans Hulden
University of Colorado Boulder
Department of Linguistics
mans.hulden@colorado.edu

Abstract

This paper presents initial experiments in data-driven morphological analysis for Finnish using deep learning methods. Our system uses a character based bidirectional LSTM and pretrained word embeddings to predict a set of morphological analyses for an input word form. We present experiments on morphological analysis for Finnish. We learn to mimic the output of the OMorFi analyzer on the Finnish portion of the Universal Dependency treebank collection. The results of the experiments are encouraging and show that the current approach has potential to serve as an extension to existing rule-based analyzers.

Tiivistelmä

Esittelemme kokeita aineistolähtöisellä syväoppimismenetelmiin perustuvalla suomen kielen morfologisella analysaattorilla. Esittelemämme järjestelmä perustuu merkkipohjaisiin LSTM-malleihin ja esiopetettuihin sanaupotuksiin. Järjestelmämme oppii matkimaan OMorFi-jäsennintä, joka on suomen kielen morfologinen analysaattori. Teemme kokeita Universal Dependency -puupankkikokoelman suomenkielisellä osuudella. Kokeemme osoittavat, että koneoppimismenetelmät tarjoavat lupaavan lähestymistavan suomen kielen morfologiseen analyysiin.

1 Introduction

The task of morphological analysis consists of providing a word form with the complete set of morphological readings it can attain (see Figure 1). It is a cornerstone in the development of natural language processing (NLP) utilities for morphologically complex languages such as the Uralic languages. It is a necessary preprocessing task because of the high type-to-token ratio, which is prevalent in morphologically complex languages. Additionally, phenomena like compounding and derivation, which frequently produce previously unseen lexemes, necessitate the use of morphological analyzers.

Proceedings of the 4th International Workshop for Computational Linguistics for Uralic Languages (IWCLUL 2018), pages 100–107,
Helsinki, Finland, January 8–9, 2018.

```
tunne   tunne    Noun+Sg+Nom
        tuntea   Verb+Act+Impv+Sg2
        tuntea   Verb+Act+Indv+Pres+Con
```

Figure 1: A complete set of morphological readings for the Finnish word *tunne*.

Hand-crafted analyzers (Koskenniemi, 1983) are the gold standard for morphological analysis. Creation of such analyzers is, however, a labor intensive process and requires expertise in linguistics, the target language and the rule formalisms used to create these analyzers. Moreover, analyzers need to be continuously updated with new lexemes in order to maintain high coverage on running text.

In this paper, we investigate an alternative to hand-crafted analyzers, namely, data-driven morphological analyzers which are learned from annotated training data. In our case, the training data consists of words and complete sets of analyses. During test time, the system takes a Finnish word such as *kisaan* ('into the competition' or 'I am competing') as input and gives a set of analyses

{`Noun+Sg+Ill`, `Verb+Act+Indv+Pres+Sg1`}

as output.

We present experiments in data-driven morphological analysis of Finnish. We learn to mimic the OMorFi analyzer (Pirinen et al., 2017) on the Finnish portion of the Universal Dependency treebank collection (Pyysalo et al., 2015). The data sets and OMorFi analyzer are further discussed in Section 3. We use a deep learning model encompassing a character-level recurrent model, which maps words onto sets of analyses as explained in Section 4. Our results, described in Section 5, show that this line of research is encouraging. We present related work in Section 2 and present concluding remarks in Section 6.

2 Related Work

The task of data-driven morphological analysis has received far less attention than morphological tagging and disambiguation which aim at producing exactly one analysis, which is correct in a given sentence context. Because hand-crafted morphological analyzers have been shown to improve the performance of neural taggers (Sagot and Martínez Alonso, 2017), the task of data-driven morphological analysis is, nevertheless, important.

The task explored in this paper is closely related to the construction of morphological guessers (Lindén, 2009), where the aim is to guess the inflectional type of a word. To the best of our knowledge deep learning methods have, however, not been applied to this task. In contrast, there is a growing body of work on deep learning for word form generation (Cotterell et al., 2017, 2016). In word form generation, or morphological reinflection, the aim is to generate word forms given lemmas and morphological analyses. Therefore, it can be seen as a natural counterpart to morphological analysis. Our work is inspired by the encoder-decoder models commonly applied in morphological reinflection (for example Kann and Schütze (2017)) but the task at hand is naturally quite different.

Several approaches have been explored for returning one analysis, or a small set of possible analyses, for a word form in context. For example, Kudo et al. (2004) apply

Conditional Random Fields for morphological analysis of Japanese but their system only returns one tokenization for a sentence and one analysis per token. This is not the same task as the one we are exploring, where the objective is to return the complete set of possible analyses. Similar in spirit is the work on Kazakh morphological analysis by Makhambetov et al. (2015). Their system, based on Hidden Markov Models, returns a subset of the analyses of a token which could plausibly occur in a given context. Sequence models are a natural choice when the aim is to generate one analysis for each word form but they are not suitable for our needs because we want to generate complete sets of analyses.

3 Data and Resources

We conduct experiments on the Finnish part of the Universal Dependency treebank collection (`UD_Finnish`) (Pyysalo et al., 2015). We analyze corpus tokens using the OMorFi[1] morphological analyzer (Pirinen et al., 2017) which is a high coverage Finnish open-source morphological analyzer capable of analyzing compounds and derivations.

Because we are learning to mimic the output of the OMorFi analyzer, we have to filter out tokens which are not recognized by OMorFi from the training, development and test set (approximately 3% of tokens in `UD_Finnish` are not recognized by OMorFi).

We slightly transform the analyses provided by OMorFi by removing lemma information since this paper does not investigate lemmatization.[2] Consequently, we conflate analyses which only differ with regard to the lemma.

Table 1 describes the Finnish UD treebank analyzed by OMorFi. The partition into training, development and test set follows the standard split provided by version 2.0 of the `UD_Finnish` treebank.

Table 1: Description of data sets used in experiments. Average ambiguity refers to the average count of distinct analyses for tokens recognized by OMorFi.

	Tokens	Tokens recognized by OMorFi	Avg. Ambiguity
Train	162,827	157,317 (96.6%)	1.82
Devel.	18,311	17,762 (97.0%)	1.76
Test	21,070	20,447 (97.0%)	1.84

As explained in Section 4, we use pretrained word vectors to initialize word embeddings. These were trained using the `word2vec` implementation in the `gensim` toolkit (Řehůřek and Sojka, 2010) on approximately 71M words of Finnish newsgroup data from the Suomi24 corpus[3]. The corpus contains texts available from the discussion forums of the Suomi24 online social networking website between years 2001 and 2015.

[1] `https://github.com/flammie/omorfi/releases/tag/20170515`

[2] We did not investigate lemmatization because it can easily be treated as a reinflection task using existing methods.

[3] Aller Media ltd. (2014). The Suomi 24 Corpus (2015H1) (text corpus). Kielipankki. Retrieved from `http://urn.fi/urn:nbn:fi:lb-201412171`

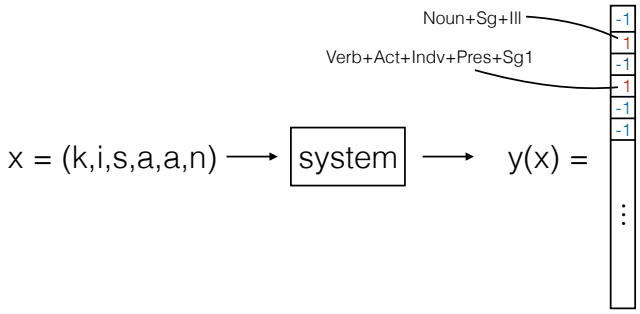

Figure 2: The system gets a Finnish word, *kisaan* ('I am competing' or 'into the competition'), as input. It then outputs the set of valid morphological analyses for the input word. For example, *kisaan* has two valid morphological analyses `Noun+Sg+Ill` and `Verb+Act+Indv+Pres+Sg1`. The input word is fed to the system as a sequence of letters $x = (k, i, s, a, a, n)$. The output $y(x)$ is a vector in $\{-1, 1\}^N$, where each index corresponds to a morphological analysis. The entry at index i is 1, iff i corresponds to a valid morphological analysis for the input word. Otherwise, it is -1.

4 Model

Morphological analysis can be formulated as a multi-label classification task, that is, the objective is to return a set of analyses for each input. We accomplish this by predicting an output vector for each input example. The vector contains one element for each morphological analysis type (for example `Noun+Sg+Nom`) and its values encode which of the analyses are active for a given input example (see Figure 2).[4] We structure the task in the following way. Each input token $x = x_1...x_n \in \Sigma^*$ (where Σ is the Finnish alphabet) is mapped into a vector $\mathbf{y}(x) \in \{-1, 1\}^{|A|} \subset \mathbb{R}^{|A|}$, where A is the set of morphological analyses found in the training data. The value $\mathbf{y}(x)_i = 1$ if the analysis corresponding to index i is a valid analysis for token x. Otherwise, $\mathbf{y}(\mathbf{x})_i = -1$.

Our system is based on word embeddings $\mathbf{e}(x)$ and character-based embeddings $\mathbf{B}(x_1, ..., x_n)$ using a bidirectional LSTM network (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). We use the final cell state of the bidirectional LSTM as our character-based embedding (that is, we do not employ an attention mechanism). The word embedding $\mathbf{e}(x)$ and the character-based embedding $\mathbf{B}(x_1, ..., x_n)$ are summed and fed into a single-layer linear perceptron network whose output is the vector $\mathbf{y}(x)$.[5]

We initialize word embeddings using pretrained word vectors as explained in Section 5. For OOV tokens, which are not found in the training set and which additionally are not present in the pretrained word embedding, we use a special unknown word embedding. During training time we randomly replace the embeddings of training words with the unknown word embedding in order to train it.

When training the system, we optimize the L1-loss of the prediction vector $\mathbf{y}(x)$ given the gold standard analysis vector $\mathbf{y} \in \{-1, 1\}^{|A|}$ as shown in Equation 1. It is

[4]In the case of Finnish, this leads to a high dimensional vector because there are thousands of possible morphological analysis types.

[5]In addition to summing the character-based embedding and pretrained word embedding vector, we also experimented with concatenating the vectors. Unfortunately, this did not improve the accuracy of the system. However, it did increase training time. Therefore, we opted for summing vectors.

easily seen that the loss is minimized when the predicted vector exactly equals $\mathbf{y}$.

$$\mathcal{L}(\mathbf{y}(x), \mathbf{y}) = \sum_{i=0}^{|A|} |\mathbf{y}(x)_i - \mathbf{y}_i| \quad (1)$$

In order to analyze a token x, we first generate the vector $\mathbf{y}(x)$ and then return all analyses corresponding to indices i for which $\mathbf{y}(x)_i > 0$. For words in the test set, which are present in the training set, we give the set of analyses found in the training set. This substantially improves performance of the system in the early stages of training but has little effect after the system is fully trained.

5 Experiments and Results

We perform experiments on the `UD_Finnish` treebank as explained above. We train the system on the training data and report performance on the held-out test set.

The system was implemented using the Dynet toolkit (Neubig et al., 2017). We set all hyper-parameters using the development set and optimize the network using Adam (Kingma and Ba, 2014) with learning rate 0.0001 and beta values $\beta_1 = 0.9; \beta_2 = 0.999$. We train the system for 50 epochs.

We use word embeddings and character-based embeddings of dimension 200. Character-based embeddings are computed in the following way: We set the hidden state dimension of the character-based LSTM to 100 and use a single layer bidirectional LSTM network. We concatenate the final 100 dimensional cell states of the forward and backward component of the bidirectional LSTM. This gives us one 200 dimensional character-based embedding vector for the input word. As explained in Section 4, the word embedding and character-based embedding are then summed.

During training, we employ 50% dropout on recurrent connections in the character-based LSTM networks. We use pretrained word vectors to initialize word embeddings. These were trained using the `word2vec` (Mikolov et al., 2013) implementation in the `gensim` toolkit (Řehůřek and Sojka, 2010). In order to train the unknown word embedding discussed in Section 4, we randomly replace word embeddings during training with the unknown word embedding with probability 2%.

The system is evaluated with regard to accuracy for full analysis sets as well as recall, precision and f-score of analyses. Full analysis accuracy defined as C/A, where C is the number of test set tokens, which received exactly the correct set of analyses, and A is the count of all tokens in the test set. Recall is defined $r = TP/T$, where TP is the amount of correct analyses that the system recovered and T is the total amount of correct analyses in the gold standard test set. Similarly, precision is defined as $p = TP/P$, where P is the total amount of analyses returned by the system. As familiar, f-score is defined as $2pr/(p + r)$.

Results of experiments are shown in Table 2. We present results separately for all tokens in the test set and OOV tokens, which were not present in the training set.

6 Discussion and Conclusions

All in all the results seem encouraging when taking into account that the proposed system is very straightforward. It is clear that performance drops drastically when the system is applied on words not occurring in the training set, however, almost half of OOV words still get the correct morphological analysis set from the system.

Table 2: Results of experiments.

	Accuracy	Recall	Precision	F-Score
All words	87.24	89.66	94.03	91.79
OOV words	44.18	43.56	58.37	49.89

Roughly 27% of errors involve mix-ups between proper nouns and common nouns. At first glance, this might seem weird because Finnish proper nouns almost always start with an upper-case letter whereas common nouns do not. However, words in sentence initial position also start with an upper-case letter. Because the current system does not employ any contextual information, it can therefore not rely on capitalization when determining the distinction between common and proper nouns. It is noteworthy, that many of these erroneous analyses only differ from the gold standard with regard to part-of-speech (`Noun` versus `Proper`). The additional inflectional information, such as case and number, are frequently correct.

Another problem, which complicates the analysis of proper nouns, is that they are often missing from the pretrained word embedding which might otherwise provide good clues toward a proper noun interpretation. Word embeddings utilizing sub-word information such as fastText embeddings (Bojanowski et al., 2016) might improve accuracy for proper nouns. Incorporating sub-word information to pretrained embeddings remains future work at the present time.

Other common errors include assigning noun or adjective analyses to participles. For example, OMorFi gives *taitava* (skillful) both a participle and an adjective reading but it does not give an adjective reading to *sanova* (a participle form of 'to say'). The distinction is mainly a matter of convention and cannot be reliably determined from the orthography or distribution of the word. In addition to these common error types, there are a substantial amount of less frequent errors but a more thorough error analysis is required to interpret these and to offer a solutions for them.

It is clear that the precision of the system is greater than its recall. When applying an analyzer to a task such as morphological disambiguation or morphological tagging, this may be problematic because the disambiguation system cannot find the correct analysis in a given context if the morphological analyzer does not suggest it. It may, however, be possible to improve the recall of the system while reducing precision. As explained in Section 4, the analyzer outputs a label corresponding to index i if $\mathbf{y}(x)_i > 0$, where $\mathbf{y}(x)$ is the output vector for example x. By replacing this formulation with $\mathbf{y}(x)_i > TH$, where TH is an adjustable hyperparameter, it is possible to create a trade-off between precision and recall. This remains future work at the current time.

We proposed a simple system for data-driven morphological analysis. The system is based on a character-based bidirectional LSTM network and utilizes pretrained word embeddings. The system is directly optimized to produce complete sets of morphological analyses. We presented experiments on the Finnish Universal Dependency treebank. The experiments show that the system is clearly capable of learning to analyze unseen word forms but there is still room for substantial improvement.

Acknowledgments

We want to thank the anonymous reviewers for their valuable comments.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. *arXiv preprint arXiv:1706.09031* .

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 10–22.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Katharina Kann and Hinrich Schütze. 2017. The LMU system for the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection. In *CoNLL Shared Task*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Kimmo Koskenniemi. 1983. Two-level model for morphological analysis. In *IJCAI*. volume 83, pages 683–685.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*. Association for Computational Linguistics, Barcelona, Spain, pages 230–237.

Krister Lindén. 2009. Guessers for finite-state transducer lexicons. *Computational Linguistics and Intelligent Text Processing* pages 158–169.

Olzhas Makhambetov, Aibek Makazhanov, Islam Sabyrgaliyev, and Zhandos Yessenbayev. 2015. *Data-Driven Morphological Analysis and Disambiguation for Kazakh*, Springer International Publishing, Cham, pages 151–163.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Tommi A Pirinen, Inari Listenmaa, Ryan Johnson, Francis M. Tyers, and Juha Kuokkala. 2017. Open morphology of finnish. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11372/LRT-1992.

Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal dependencies for finnish. In *NODALIDA*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. `http://is.muni.cz/publication/884893/en`.

Benoît Sagot and Héctor Martínez Alonso. 2017. Improving neural tagging with lexical information. In *Proceedings of the 15th International Conference on Parsing Technologies*. Association for Computational Linguistics, Pisa, Italy, pages 25–31. http://www.aclweb.org/anthology/W17-63 4.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.