Neural Morphological Analysis: Encoding-Decoding Canonical Segments

Katharina Kann
Center for Information and Language Processing
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Canonical morphological segmentation aims to divide words into a sequence of standardized segments. In this work, we propose a character-based neural encoder-decoder model for this task. Additionally, we extend our model to include morpheme-level and lexical information through a neural reranker. We set the new state of the art for the task improving previous results by up to 21% accuracy. Our experiments cover three languages: English, German and Indonesian.

1 Introduction

Morphological segmentation aims to divide words into morphemes, meaning-bearing sub-word units. Indeed, segmentations have found use in a diverse set of NLP applications, e.g., automatic speech recognition (Afify et al., 2006), keyword spotting (Narasimhan et al., 2014), machine translation (Clifton and Sarkar, 2011) and parsing (Seeker and Çetinoğlu, 2015). In the literature, most research has traditionally focused on *surface segmentation*, whereby a word w is segmented into a sequence of substrings whose concatenation is the entire word; see Ruokolainen et al. (2016) for a survey. In contrast, we consider *canonical segmentation*: w is divided into a sequence of standardized segments. To make the difference concrete, consider the following example: the surface segmentation of the complex English word *achievability* is *achiev+abil+ity*, whereas its canonical segmentation is *achieve+able+ity*, i.e., we restore the alterations made during word formation.

Canonical versions of morphological segmentation have been introduced multiple times in the literature (Kay, 1977; Naradowsky and Goldwater, 2009; Cotterell et al., 2016). Canonical segmentation has several representational advantages over surface segmentation, e.g., whether two words share a morpheme is no longer obfuscated by orthography. However, it also introduces a hard algorithmic challenge: in addition to segmenting a word, we must reverse orthographic changes, e.g., mapping *achievability*↦*achieveableity*.

Computationally, canonical segmentation can be seen as a sequence-to-sequence problem: we must map a word form to a canonicalized version with segmentation boundaries. Inspired by the recent success of neural encoder-decoder models (Sutskever et al., 2014) for sequence-to-sequence problems in NLP, we design a neural architecture for the task. However, a naïve application of the encoder-decoder model ignores much of the linguistic structure of canonical segmentation—it cannot directly model the individual canonical segments, e.g., it cannot easily produce segment-level embeddings. To solve this, we use a neural reranker on top of the encoder-decoder, allowing us to embed both characters and entire segments. The combined approach outperforms the state of the art by a wide margin (up to 21% accuracy) in three languages: English, German and Indonesian.

2 Neural Canonical Segmentation

We begin by formally describing the canonical segmentation task. Given a discrete alphabet Σ (e.g., the 26 letters of the English alphabet),

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 961–967,
Austin, Texas, November 1-5, 2016.

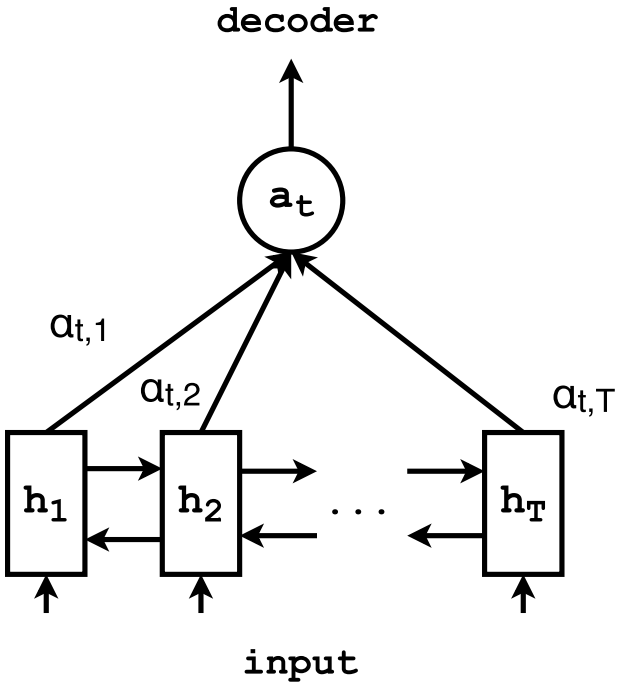

Figure 1: Detailed view of the attention mechanism of the neural encoder-decoder.

our goal is to map a word $w \in \Sigma^*$ (e.g., w=*achievability*), to a canonical segmentation $c \in \Omega^*$ (e.g., c=*achieve+able+ity*). We define $\Omega = \Sigma \cup \{+\}$, where $+$ is a distinguished separation symbol. Additionally, we will write the segmented form as $c=\sigma_1+\sigma_2+\ldots+\sigma_n$, where each segment $\sigma_i \in \Sigma^*$ and n is the number of canonical segments.

We take a probabilistic approach and, thus, attempt to learn a distribution $p(c \mid w)$. Our model consists of two parts. First, we apply an encoder-decoder recurrent neural network (RNN) (Bahdanau et al., 2014) to the sequence of characters of the input word to obtain candidate canonical segmentations. Second, we define a neural reranker that allows us to embed individual morphemes and chooses the final answer from within a set of candidates generated by the encoder-decoder.

2.1 Neural Encoder-Decoder

Our encoder-decoder is based on Bahdanau et al. (2014)'s neural machine translation model.[1] The **encoder** is a bidirectional gated RNN (GRU) (Cho et al., 2014b). Given a word $w \in \Sigma^*$, the input to the encoder is the sequence of characters of w, represented as one-hot vectors. The **decoder** defines a conditional probability distribution over $c \in \Omega^*$ given w:

$$\begin{aligned} p_{\text{ED}}(c \,|\, w) &= \prod_{t=1}^{|c|} p(c_t | c_1, \ldots, c_{t-1}, w) \\ &= \prod_{t=1}^{|c|} g(c_{t-1}, s_t, a_t) \end{aligned}$$

where g is a nonlinear activation function, s_t is the state of the decoder at t and a_t is a weighted sum of the $|w|$ states of the encoder. The state of the encoder for w_i is the concatenation of forward and backward hidden states $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ for w_i. An overview of how the attention weight and the weighted sum a_t are included in the architecture can be seen in Figure 1. The attention weights $\alpha_{t,i}$ at each timestep t are computed based on the respective encoder state and the decoder state s_t. See Bahdanau et al. (2014) for further details.

2.2 Neural Reranker

The encoder-decoder, while effective, predicts each output character in Ω sequentially. It does not use explicit representations for entire segments and is incapable of incorporating simple lexical information, e.g., does this canonical segment occur as an independent word in the lexicon? Therefore, we extend our model with a reranker.

The reranker rescores canonical segmentations from a candidate set, which in our setting is sampled from p_{ED}. Let the sample set be $\mathcal{S}_w = \{k^{(i)}\}_{i=1}^{N}$ where $k^{(i)} \sim p_{\text{ED}}(c \mid w)$. We define the neural reranker as

$$p_\theta(c \,|\, w) = \frac{\exp\Big(u^\top \tanh(W v_c) + \tau \log p_{\text{ED}}(c \,|\, w)\Big)}{Z_\theta}$$

where $v_c = \sum_{i=1}^{n} v_{\sigma_i}$ (recall $c = \sigma_1+\sigma_2+\ldots+\sigma_n$) and v_{σ_i} is a one-hot morpheme embedding of σ_i with an additional binary dimension marking if σ_i occurs independently as a word in the language.[2] The partition function is $Z_\theta(w)$ and the parameters are $\theta = \{u, W, \tau\}$. The parameters W and u

[1] github.com/mila-udem/blocks-examples/tree/master/machine_translation

[2] To determine if a canonical segment is in the lexicon, we check its occurrence in ASPELL. Alternatively, one could ask whether it occurs in a large corpus, e.g., Wikipedia.

are projection and hidden layers, respectively, of a multi-layered perceptron and τ can be seen as a temperature parameter that anneals the encoder-decoder model p_{ED} (Kirkpatrick, 1984). We define the partition function over the sample set $\mathcal{S}_w$:

$$Z_\theta = \sum_{k \in \mathcal{S}_w} \exp\left(u^\top \tanh(W v_k) + \tau \log p_{\text{ED}}(k \mid w)\right).$$

The reranking model's ability to embed morphemes is important for morphological segmentation since we often have strong corpus-level signals. The reranker also takes into account the character-level information through the score of the encoder-decoder model. Due to this combination we expect stronger performance.

3 Related Work

Various approaches to morphological segmentation have been proposed in the literature. In the unsupervised realm, most work has been based on the principle of minimum description length (Cover and Thomas, 2012), e.g., LINGUISTICA (Goldsmith, 2001; Lee and Goldsmith, 2016) or MORFESSOR (Creutz and Lagus, 2002; Creutz et al., 2007; Poon et al., 2009). MORFESSOR was later extended to a semi-supervised version by Kohonen et al. (2010). Supervised approaches have also been considered. Most notably, Ruokolainen et al. (2013) developed a supervised approach for morphological segmentation based on conditional random fields (CRFs) which they later extended to work also in a semi-supervised way (Ruokolainen et al., 2014) using letter successor variety features (Hafer and Weiss, 1974). Similarly, Cotterell et al. (2015) improved performance with a semi-Markov CRF.

More recently, Wang et al. (2016) achieved state-of-the-art results on surface morphological segmentation using a window LSTM. Even though Wang et al. (2016) also employ a recurrent neural network, we distinguish our approach, in that we focus on *canonical* morphological segmentation, rather than *surface* morphological segmentation.

Naturally, our approach is also relevant to other applications of recurrent neural network transduction models (Sutskever et al., 2014; Cho et al., 2014a). In addition to machine translation (Bahdanau et al., 2014), these models have been successfully applied to many areas of NLP, including parsing (Vinyals et al., 2015), morphological reinflection (Kann and Schütze, 2016) and automatic speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

4 Experiments

To enable comparison to earlier work, we use a dataset that was prepared by Cotterell et al. (2016) for canonical segmentation.[3]

4.1 Languages

The dataset we work on covers 3 languages: English, German and Indonesian. English and German are West Germanic Languages, with the former being an official languages in nearly 60 different states and the latter being mainly spoken in Western Europe. Indonesian — or *Bahasa Indonesia*— is the official language of Indonesia.

Cotterell et al. (2016) report the best experimental results for Indonesian, followed by English and finally German. The high error rate for German might be caused by it being rich in orthografic changes. In contrast, Indonesian morphology is comparatively simple.

4.2 Corpora

The data for the English language was extracted from segmentations derived from the CELEX database (Baayen et al., 1993). The German data was extracted from DerivBase (Zeller et al., 2013), which provides a collection of derived forms together with the transformation rules, which were used to create the canonical segmentations. Finally, the data for Bahasa Indonesia was collected by using the output of the MORPHIND analyzer (Larasati et al., 2011), together with an open-source corpus of Indonesian. For each language we used the 10,000 forms that were selected at random by Cotterell et al. (2016) from a uniform distribution over types to form the corpus. Following them, we perform our experiments on 5 splits of the data into 8000 training forms, 1000 development forms and 1000 test forms and report averages.

[3] ryancotterell.github.io/canonical-segmentation

4.3 Training

We train an ensemble of five encoder-decoder models. The encoder and decoder RNNs each have 100 hidden units. Embedding size is 300. We use ADADELTA (Zeiler, 2012) with a minibatch size of 20. We initialize all weights (encoder, decoder, embeddings) to the identity matrix and the biases to zero (Le et al., 2015). All models are trained for 20 epochs. The hyperparameter values are taken from Kann and Schütze (2016) and kept unchanged for the application to canonical segmentation described here.

To train the reranking model, we first gather the sample set $\mathcal{S}_w$ on the training data. We take 500 individual samples, but (as we often sample the same form multiple times) $|\mathcal{S}_w| \approx 5$. We optimize the log-likelihood of the training data using ADADELTA. For generalization, we employ L_2 regularization and we perform grid search to determine the coefficient $\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. To decode the model, we again take 500 samples to populate $\mathcal{S}_w$ and select the best segmentation.

Baselines. Our first baseline is the joint transduction and segmentation model (**JOINT**) of Cotterell et al. (2016). It is the current state of the art on the datasets we use and the task of canonical segmentation in general. This model uses a jointly trained, separate transduction and segmentation component. Importantly, the joint model of Cotterell et al. (2016) *already contains segment-level features*. Thus, reranking this baseline would not provide a similar boost.

Our second baseline is a weighted finite-state transducer (**WFST**) (Mohri et al., 2002) with a log-linear parameterization (Dreyer et al., 2008), again, taken from Cotterell et al. (2016). The WFST baseline is particularly relevant because, like our encoder-decoder, it formulates the problem directly as a string-to-string transduction.

Evaluation Metrics. We follow Cotterell et al. (2016) and use the following evaluation measures: error rate, edit distance and morpheme F_1. Error rate is defined as 1 minus the proportion of guesses that are completely correct. Edit distance is the Levenshtein distance between guess and gold standard. For this, guess and gold are each represented as one string with a distinguished character denoting the segment boundaries. Morpheme F_1 compares the morphemes in guess and gold. Precision (resp. recall) is the proportion of morphemes in guess (resp. gold) that occur in gold (resp. guess).

		RR	ED	Joint	WFST	UB
error	en	**.19** (.01)	.25 (.01)	0.27 (.02)	0.63 (.01)	.06 (.01)
	de	**.20** (.01)	.26 (.02)	0.41 (.03)	0.74 (.01)	.04 (.01)
	id	**.05** (.01)	.09 (.01)	0.10 (.01)	0.71 (.01)	.02 (.01)
edit	en	**.21** (.02)	.47 (.02)	0.98 (.34)	1.35 (.01)	.10 (.02)
	de	**.29** (.02)	.51 (.03)	1.01 (.07)	4.24 (.20)	.06 (.01)
	id	**.05** (.00)	.12 (.01)	0.15 (.02)	2.13 (.01)	.02 (.01)
F_1	en	**.82** (.01)	.78 (.01)	0.76 (.02)	0.53 (.02)	.96 (.01)
	de	**.87** (.01)	.86 (.01)	0.76 (.02)	0.59 (.02)	.98 (.00)
	id	**.96** (.01)	.93 (.01)	0.80 (.01)	0.62 (.02)	.99 (.00)

Table 1: Error rate (top), edit distance (middle), F_1 (bottom) for canonical segmentation. Each double column gives the measure and its standard deviation. Best result on each line (excluding UB) in bold. RR: encoder-decoder+reranker. ED: encoder-decoder. JOINT, WFST: baselines (see text). UB: upper bound, the maximum score our reranker could obtain, i.e., considering the best sample in the predictions of ED.

5 Results

The results of the canonical segmentation experiment in Table 1 show that both of our models improve over all baselines. The encoder-decoder alone has a .02 (English), .15 (German) and .01 (Indonesion) lower error rate than the best baseline. The encoder-decoder improves most for the language for which the baselines did worst. This suggests that, for more complex languages, a neural network model might be a good choice.

The reranker achieves an additional improvement of .04 to .06. for the error rate. This is likely due to the additional information the reranker has access to: morpheme embeddings and existing words.

Important is also the upper bound we report. It shows the maximum performance the reranker could achieve, i.e., evaluates the best solution that appears in the set of candidate answers for the reranker. The right answer is contained in $\geq 94\%$ of samples. Note that, even though the upper bound goes up with the number of samples we take, there is no guarantee for any finite number of samples that they will contain the true answer. Thus, we would need to take an infinite number of samples to get a perfect upper bound. However, as the current upper bound is quite high, the encoder-decoder proves to be an appropri-

ate model for the task. Due to the large gap between the performance of the encoder-decoder and the upper bound, a better reranker could further increase performance. We will investigate ways to improve the reranker in future work.

Error analysis. We give for representative samples the error (E for the segmentation produced by our method) and the correct analysis (G for gold).

We first analyze cases in which the right answer does not appear at all in the samples drawn from the encoder-decoder. Those include problems with umlauts in German (G: *verflüchtigen*↦ *ver+flüchten+ig*, E: *verflucht+ig*) and orthographic changes at morpheme boundaries (G:*cutter*↦*cut+er*, E: *cutter* or *cutt+er*, sampled with similar frequency). There are also errors that are due to problems with the annotation, e.g., the following two gold segmentations are arguably incorrect: *tec*↦*detective* and *syrerin*↦*syr+er+in* (*syr* is neither a word nor an affix in German).

In other cases, the encoder-decoder does find the right solution (G), but gives a higher probability to an incorrect analysis (E). Examples are a wrong split into adjectives or nouns instead of verbs (G: *fügsamkeit*↦*fügen+sam+keit*, E: *fügsam+keit*), the other way around (G: *zähler*↦*zahl+er*, E: *zählen+er*), cases where the wrong morphemes are chosen (G: *precognition*↦*pre+cognition*, E: *precognit+ion*), difficult cases where letters have to be inserted (G: *redolence*↦*redolent+ence*, E: *re+dolence*) or words the model does not split up, even though they should be (G: *additive*↦ *addition+ive*, E: *additive*).

Based on its access to lexical information and morpheme embeddings, the reranker is able to correct some of the errors made by the encoder-decoder. Samples are G: *geschwisterpärchen*↦ *geschwisterpaar+chen*, E: *geschwisterpar+chen* (*geschwisterpaar* is a word in German but *geschwisterpar* is not) or G: *zickig*↦ *zicken+ig*, E: *zick+ig* (with *zicken*, but not *zick*, being a German word).

Finally, we want to know if segments that appear in the test set without being present in the training set are a source of errors. In order to investigate that, we split the test samples into two groups: The first group contains the samples for which our system finds the right answer. The second one contains all other samples. We compare the percentage of samples that do not appear in the training data for both groups. We exemplarily use the German data and the results results are shown in Table 2. First, it can be seen that very roughly about a third of all segments does not appear in the training data. This is mainly due to unseen lemmas as their stems are naturally unknown to the system. However, the correctly solved samples contain nearly 10% more unseen segments. As the average number of segments per word for wrong and right solutions — 2.44 and 2.11, respectively — does not differ by much, it seems unlikely that many errors are caused by unknown segments.

wrong samples	right samples
27.33 (.02)	36.60 (.01)

Table 2: Percentage of segments in the solutions for the test data that do not appear in the training set - split by samples that our system does or does not get right. We use the German data and average over the 5 splits. Standard deviation in parenthesis.

6 Conclusion and Future Work

We developed a model consisting of an encoder-decoder and a neural reranker for the task of canonical morphological segmentation. Our model combines character-level information with features on the morpheme level and external information about words. It defines a new state of the art, improving over baseline models by up to .21 accuracy, 16 points F_1 and .77 Levenshtein distance.

We found that $\geq 94\%$ of correct segmentations are in the sample set drawn from the encoder-decoder model, demonstrating the upper bound on the performance of our reranker is quite high; in future work, we hope to develop models to exploit this.

Acknowledgments

We gratefully acknowledge the financial support of Siemens for this research.

References

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. 2006. On the use of morphological analysis for dialectal Arabic speech recognition. In *Proc. of INTERSPEECH*.

R. H. Baayen, R. Piepenbrock, and H. Van Rijn. 1993. The CELEX lexical data base on CD-ROM.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. of EMNLP*.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proc. of ACL*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *Proc. of CoNLL*.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *Proc. of NAACL*.

Thomas M Cover and Joy A Thomas. 2012. *Elements of Information Theory*. John Wiley & Sons.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proc. of the ACL-02 Workshop on Morphological and Phonological Learning*.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1):3:1–3:29.

Markus Dreyer, Jason R. Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proc. of EMNLP*.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc of. ICASSP*.

Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proc. of ACL*.

Martin Kay. 1977. Morphological and syntactic analysis. *Linguistic Structures Processing*, 5:131–234.

Scott Kirkpatrick. 1984. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proc. of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*.

Septina Dian Larasati, Vladislav Kuboň, and Daniel Zeman. 2011. Indonesian morphology tool (morphind): Towards an indonesian corpus. In *Proc. of SFCM*. Springer.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Jackson L. Lee and John A. Goldsmith. 2016. Linguistica 5: Unsupervised learning of linguistic structure. In *Proc. of NAACL*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proc. of IJCAI*.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *Proc. of EMNLP*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of NAACL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proc. of CoNLL*.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and mikko kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proc. of EACL*.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. Comparative study of minimally supervised morphological segmentation. *Computational Linguistics*, 42(1):91–120.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL*, 3:359–373.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proc. of AAAI*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Britta Zeller, Jan Šnajder, and Sebastian Padó. 2013. Derivbase: Inducing and evaluating a derivational morphology resource for german. In *Proc. of ACL*.