Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Morphological Segmentation with Window LSTM Neural Networks

Linlin Wang and **Zhu Cao** and **Yu Xia** and **Gerard de Melo**
Institute for Interdisciplinary Information Sciences
Tsinghua University, Beijing 100084, China
{ll-wang13, cao-z13, xiay12}@mails.tsinghua.edu.cn, gdm@demelo.org

Abstract

Morphological segmentation, which aims to break words into meaning-bearing morphemes, is an important task in natural language processing. Most previous work relies heavily on linguistic preprocessing. In this paper, we instead propose novel neural network architectures that learn the structure of input sequences directly from raw input words and are subsequently able to predict morphological boundaries. Our architectures rely on Long Short Term Memory (LSTM) units to accomplish this, but exploit windows of characters to capture more contextual information. Experiments on multiple languages confirm the effectiveness of our models on this task.

1 Introduction

Morphological segmentation is an important sub-task in many natural language processing (NLP) applications, aiming to break words into meaning-bearing sub-word units called morphemes (Creutz et al. 2007). Numerous methods in NLP, information retrieval, and text mining make use of word-level information. However, since the number of word forms in a language is often infinite, morphological preprocessing may be vital for such methods to generalize to new forms. Morphological segmentation may allow us to break them down into more familiar units that have been observed before in the data. For instance, in the English language, we may want to segment forms such as *subreddits*, *clickbait*, and *nanodiamonds*. While some of these may be found in external resources (de Melo 2014), others may be fairly rare. In languages with complex morphology, we moreover may find thousands of different forms of the same word, causing severe sparsity challenges in models that rely on token identities. Accurate morphological segmentation not only allows us to relate new forms to previously known word forms but also greatly decreases data sparsity.

Most previous work in this area, however, relies heavily on linguistic knowledge to make morphological predictions, and accurate segmentation is particularly challenging especially for languages with complex morphology that are under-resourced, such as Arabic and Hebrew. Thus, unsupervised approaches often deliver unsatisfactory results due to a lack of linguistic guidance, while supervised and semi-supervised approaches have tended to rely on extensive feature engineering.

In recent years, several studies have pointed out the merits of being able to learn linguistic structure "from scratch" (Collobert et al. 2011), i.e., from raw input features, such as the raw input characters in our case, forgoing any additional feature engineering.

In this paper, we embrace this trend and attempt to achieve this goal by drawing on neural networks equipped with Long Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber 1997). LSTM networks have recently enjoyed great success in sequence learning tasks, including state-of-the-art results on highly non-trivial tasks such as machine translation (Sutskever, Vinyals, and Le 2014).

Still, regular LSTM networks do not necessarily give satisfactory results on morphological segmentation. Most such networks operate in a forward-directional manner, receiving inputs one-by-one and later predicting outputs. In our paper, we propose three window-based LSTM neural network architectures, simple Window LSTMs, Multi-Window LSTMs, and Bidirectional Multi-Window LSTMs. All of these rely on a window-based approach that facilitates predictions based on both past and future inputs, i.e., left and right neighbors. This enables the network to better learn the input sequence structure and predict morphological boundaries. While reasonably straightforward, this approach proves fairly powerful.

In the simple Window LSTM model, we use LSTMs to capture a range of input character sequences. Then, in the Multi-Window LSTM model, an entire word is processed jointly, with the number of LSTM applications varying in accordance with the number of labels in the input word. Finally, we introduce Bidirectional Multi-Window LSTMs to capture more informative features. When looking at input character windows and label sequences, this bidirectional model accomplishes this by making both a forward and a backward pass for a given word.

The main contributions in this work are as follows:

1. Instead of relying heavily on linguistic knowledge as in previous work using CRFs, our models automatically learn the structure of input sequences and predict morphological boundaries for raw words.
2. We present several new LSTM architectures with a window design that captures necessary letter context for mor-

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

phology prediction. While these architectures aim at morphological segmentation, they do not use morphology-specific engineering and in fact they could also be applied to other sequence labeling tasks.

3. Even with limited amounts of training data, our LSTM models achieve good results for languages with complex morphological forms, opening up the possibility for widespread use for under-resourced languages.

2 Related Work

Research on unsupervised morphological segmentation started with Harris (1951) half a century ago. For many years, morphological processing was either performed using extensive manual rule engineering, which is very costly, or using finite state automata representing large morphological dictionaries, which however do not generalize well to new out-of-dictionary words.

More recent machine learning approaches tend to fall into three general categories: minimum description length (MDL) based unsupervised approaches, conditional random field (CRF) based supervised approaches, and semi-supervised approaches. MDL-based unsupervised segmentation methods provided for an important breakthrough and were applied to several languages (Goldsmith 2001). However, this approach requires manual human retrofitting. Creutz and Lagus (2002) proposed an unsupervised word segmentation approach that relies on the MDL principle and maximum likelihood optimization. One of the most well-known systems is Morfessor (Creutz et al. 2007), a generative probabilistic model. Poon, Cherry, and Toutanova (2009) presented an unsupervised method that makes segmentation decisions based on the classic log-linear model framework, into which contextual and global features like lexicon priors are incorporated. However, entirely unsupervised systems tend not to be competitive as long as even just a small amount of segmented training data is available. Furthermore, unsupervised segmentation still has considerable weaknesses, including oversegmentation of roots and erroneous segmentation of affixes, among others.

To deal with limitations of this sort, approaches such as minimally supervised Adaptor Grammars have been proposed (Sirts and Goldwater 2013). Ruokolainena et al. (2013) presented a supervised approach for morphological segmentation based on conditional random fields, obtaining outstanding results in segmentation with only annotated data. These supervised approaches only make use of labeled data. To leverage the unannotated part, a semi-supervised version of Morfessor was introduced (Kohonen, Virpioja, and Lagus 2010), as well a method to incorporate unlabeled data into CRFs by exploiting it in sophisticated ways to augment the feature set with rich features (Ruokolainena et al. 2014).

Still, these supervised and semi-supervised segmentation approaches require extensive and rather involved linguistic feature engineering. Apart from this, semi-supervised morphological segmentation also requires rather complex training procedures in conjunction with significant amounts of labeled and unlabeled data.

In recent years, however, Collobert et al. (2011) and several other studies have proposed eschewing background knowledge-driven feature engineering in favor of approaches that learn linguistic structure "from scratch", i.e., from the raw input data. Neural network-based approaches are particularly well-suited for this sort of learning and have been applied to numerous different tasks (Collobert et al. 2011; Wang et al. 2015). Theoretical analysis by Wang and Manning (2013) has shed further light on the relationship between CRFs and deep neural networks, confirming the benefits of neural network-based approaches.

In this paper, we follow this promising new direction, making use of neural networks architectures to achieve such an independence from feature engineering. Our approach relies on LSTM cells' capabilities for capturing information in long term memory. LSTMs are a well-known recurrent neural network architecture, due to their ability to capture long-range associations aggregating increasing amounts of information, while avoiding the issue of vanishing gradients in training (Hochreiter and Schmidhuber 1997). LSTM-based architectures have achieved state-of-the-art results on tasks such as English-to-French machine translation (Sutskever, Vinyals, and Le 2014), syntactic constituency parsing (Tai, Socher, and Manning 2015), transition-based dependency parsing (Dyer et al. 2015), and grapheme-to-phoneme conversion (Rao et al. 2015).

Although LSTMs have proven successful in a range of different settings, they do not yield state-of-the-art results on morphological segmentation. In this paper, we propose modified network architectures that aim to better capture contextual information by going beyond the simple forward-pass architecture used in regular LSTM networks. Instead, we obtain hybrid models that add aspects of feedforward networks, similar to how architectures for automatic image captioning combine convolutional neural networks for image analysis with recurrent networks for sentence generation (Karpathy and Fei-Fei 2015; Vinyals et al. 2014). Although our new models are guided by the task of morphological segmentation, we note that these do not rely on segmentation-specific engineering. Instead, their architecture is fairly generic and likely to benefit other sequence labeling tasks.

3 Segmentation Models

3.1 Morphological Segmentation

Morphological segmentation aims to split words into morphemes. This task can be viewed as a structured classification problem, in which each character is to be assigned one of several predetermined classes. We denote these classes as follows: (B) represents the beginning of a multi-character morpheme, (M) the middle of a multi-character morpheme, (E) stands for the end of a multi-character morpheme, and (S) denotes a single character morpheme. Other schemes are possible as well. For instance, Green and DeNero (2012) rely on just two class labels (B) and (M), while Ruokolainena et al. (2013) suggests adopting the more fine-grained set $\{$B, M, E, S$\}$.

Taking the word "actors" as an example, the corresponding morphological segmentation should be:

act+or+s

By adding the two extra notational symbols <w> and </w> to indicate the start and end of a word, respectively, we represent the above segmentation form as:

<w>	a	c	t	o	r	s	</w>
START	B	M	E	B	E	S	STOP

To summarize, morphological segmentation can be cast as a sequence labeling problem in which the objective is to classify each character of a word into one of the aforementioned classes. This typically boils down to maximizing a log-likelihood function.

Several possibilities for this exist. For instance, one could formulate the objective as follows with respect to model parameters θ:

$$\theta^* = \arg\max_{\theta} \sum_{(v,l)} \log p(l|v;\theta) \quad (1)$$

Here, v corresponds to a character from a training word, l is the corresponding label for this character, (v, l) is a training pair and θ represents model parameters. Thus, given an input character from a word, with Eq. (1), we seek the best corresponding labels in terms of maximizing the conditional probability. The overall morphological segmentation task aims at maximizing the log-likelihood over all words in the training data.

Alternatively, we may also adopt the following form of log-likelihood function:

$$\theta^* = \arg\max_{\theta} \sum_{(V,L)} \log p(L|V;\theta) \quad (2)$$

Here, V is the input word, L stands for the corresponding label sequences for this word, (V, L) is a training pair and θ represents model parameters. Given an input word, composed of several characters, the objective is to maximize the probability of predicting appropriate label sequences for this word with current model parameters.

Suppose we have a label sequence $L = (l_0, \ldots, l_T)$ for a word of length T, where each l_t represents one particular label for the t-th time step, i.e., character, of the word. With the chain rule, we obtain:

$$\log p(L|V;\theta) = \sum_{t=0}^{T} \log p(l_t|V;\theta;l_0,\ldots,l_{t-1}) \quad (3)$$

In the following, we will present LSTM-based neural network architectures capable of modeling either $p(L|V;\theta)$ or $p(L_t|V, l_0, \ldots, l_{t-1})$.

4 Model Architectures

We propose a series of window-based LSTM architectures for morphological segmentation, with different objective functions. We begin with a very basic window model, followed by the Multi-Window LSTM model that models entire words. Finally, we present the Bidirectional Multi-Window LSTM model that applies both forward and backward passes to learn more fine-grained sequence features.

4.1 LSTM Memory Blocks

Like regular recurrent neural networks, LSTMs compute a new hidden state given a previous hidden state and a new input. However, since ordinary recurrent neural networks, in training, suffer from the phenomena of exploding and vanishing gradients, LSTM networks adopt a special form of memory block to avoid these challenges (Hochreiter and Schmidhuber 1997).

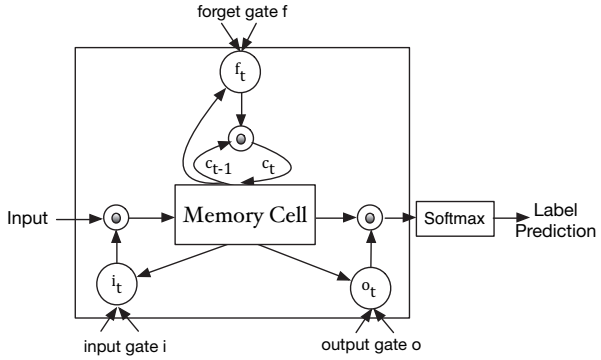

Figure 1: LSTM Memory Cell Structure

Inside the block depicted in Figure 1, there is a memory cell with a self-recurrent connection and three gates to control its information flow. The input gate i governs the input flow into the memory cell, and the output gate o determines to what degree to output new activations. Additionally, the crucial forget gate f regulates a self-recurrent connection, enabling this cell to memorize or forget the previous state. The previous activation h from time $t-1$ is fed to the memory at time t through these three gates. The state of the memory cell is fed back together with the forget gate. Formally, the working principle of the memory block is as follows:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1}) \quad (4)$$
$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1}) \quad (5)$$
$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1}) \quad (6)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot H(W_{cx}x_t + W_{ch}h_{t-1}) \quad (7)$$
$$h_t = o_t \odot c_t \quad (8)$$

Here, i_t, f_t, o_t and c_t are activation vectors at time t of the input gate, forget gate, output gate and memory cell, respectively, the size of which should be the same as for the hidden vector h. W stands for the corresponding weight parameters, and its subscripts indicate different connections. $\odot$ denotes the element-wise product operation with a gate value, $\sigma(\cdot)$ is the logistic sigmoid function, and $H(\cdot)$ represents the hyperbolic tangent.

4.2 Window LSTM Model

We begin with a simpler model that makes individual predictions for characters. In Eq. (1), we proposed a morphological segmentation objective that involves maximizing the conditional probability of the predicted label given an input char-

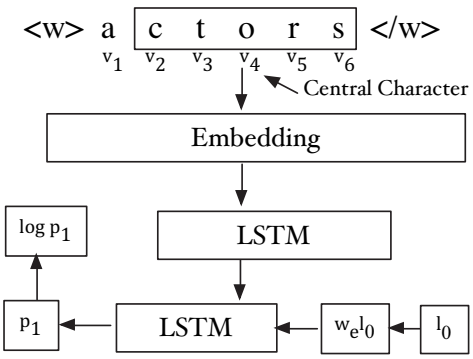

Figure 2: Window LSTM Model

acter and its associated features with current model parameters. This requires our model to possess the capability of capturing effective features for the input character and properly describing the conditional probability for the possible labels. Accordingly, we rely on the architecture illustrated in Figure 2. An LSTM unit is responsible for automatically learning the structure of the input window. Given a starting label (motivated by our desire to later extend this model to sequences of labels), it then yields a probability distribution for the candidate labels of the target character.

Unlike regular LSTMs, we adopt a fixed-width window approach to extract features for the target input character. This window approach rests on the assumption that the label for a character depends quite substantially on its neighboring characters. For a given character in a word, we consider the size of the window around this character as a hyperparameter. Every character in this window is encoded with a one-hot representation, and then we feed the entire window to the LSTM in the hope that it will scrutinize the embedding so as to capture meaningful signals and structures.

Subsequently, a label is predicted given the previous hidden state and an input l_0 (=START) as input. Since the activation of h at time $t-1$ is fed to the LSTM unit's memory at time t via three gates and the state of the memory cell is fed back with the forget gate shown in Eq. (4), we pass the resulting h_t into the Softmax for label prediction, which yields a probability distribution p over all candidate labels:

$$p_{t+1} = S(h_t), \tag{9}$$

where $S(\cdot)$ stands for the Softmax function.

With the above circuitry, our Window LSTM model tackles the morphological segmentation task in accordance with Eq. (10) as follows:

$$T_{t-1} = LT_W(v) \tag{10}$$

$$T_t = W_e l_0 \tag{11}$$

$$p = LSTM(T_t) \tag{12}$$

where $LT_W(\cdot)$ is the embedding form for a character v in the input window after a lookup table operation, l_0 is the input label, and W_e is the label embedding matrix. T_{t-1} is obtained based on v by the first LSTM, which aims to inform the LSTM about the input character contents. Then, T_t is obtained based on an input, in this case just a starting symbol START. Finally, p is the probability distribution over labels for the central target character in this input window.

4.3 Multi-Window LSTMs

Similar to probabilistic sequence models, Collobert et al. (2011) point out that it can be worth modeling strong dependencies in label sequences. The transition from label i to label j can be modeled explicitly in a transition score matrix, as a set of parameters, to describe such dependencies. Indeed, it appears that such transitions can be of great significance in morphological segmentation. To better capture the label transitions and allow our network to acquire more detailed knowledge of the input structure, we devise a second architecture that applies LSTMs to an entire label sequence.

This architecture is designed for the objective function given by Eq. (3). We call it the Multi-Window LSTM model. While the previous simple Window LSTM model considers a new character window and label independently at each step, the Multi-Window LSTM model presented here processes an entire word jointly. In a first time step, windows for all characters are fed to a first LSTM, and then in subsequent time steps, LSTM units predict labels given previous character labels, denoted as $L_0, \ldots, L_{t-1}$, and depicted in Figure 3.

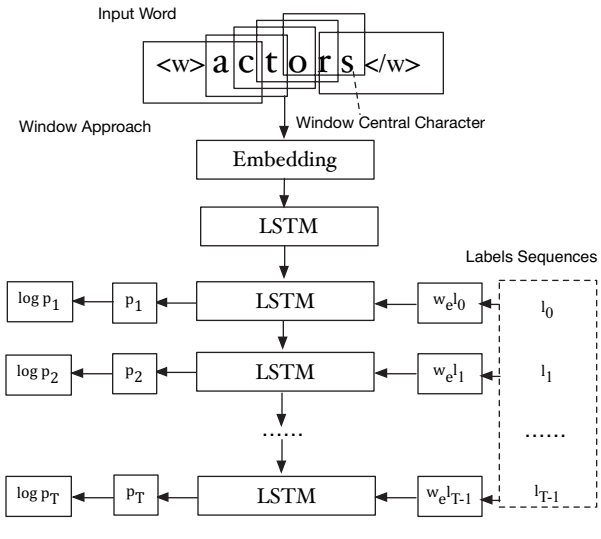

Figure 3: Multi-Window LSTM model

First, an LSTM encodes the entire input word V, at what we define to be time step $t = -1$. We again adopt the window approach proposed earlier to link characters to information about their respective neighboring characters. However, this time, we connect all the individual windows for the respective characters in a word. We conjecture that this enables the network architecture to better capture structural features within the input word automatically.

The LSTMs are then deployed to predict all the labels for the word. Unrolled, these can be thought of as a series of

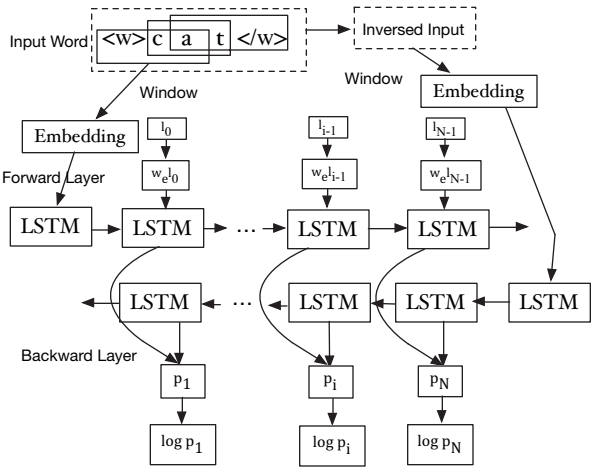

Figure 4: Bidirectional Multi-Window LSTM model

LSTMs that share model parameters and each focus on one label l_t for a time step $t \in \{0, \ldots, T-1\}$. LSTMs have a feed-forward information flow, and h_{t-1} of the LSTM at time $t-1$ is sent to the LSTM for time t. In terms of Eq. (4), the predicted label at time $t-1$ is fed back in addition to the memory output at time t into a Softmax operation for label prediction. Thus, at each time step (after step -1), the probabilities for one label are determined given previous labels.

Overall, we compute:

$$T_{-1} = LT_W(V) \quad (13)$$

$$T_t = W_e l_t \text{ for } t \in \{0, \ldots, T-1\} \quad (14)$$

$$p_{t+1} = LSTM(T_t) \text{ for } t \in \{0, \ldots, T-1\} \quad (15)$$

where $LT_W(V)$ is the encoding form for input word V after a lookup table operation, and the entire input word here is composed of windows for every character. l_t represents labels from the label sequences for this input word, and W_e is the embedding matrix for the label information. l_0 and l_n are added to indicate label start and label end, respectively. T_{-1} is obtained based on the windows for the input word V at time -1 and T_t, $t \in \{0, \ldots, T-1\}$, is obtained based on every item in the label sequence at time t. Finally, p_{t+1} is the resulting probability distribution over labels at time step $t+1$.

4.4 Bidirectional Multi-Window LSTMs

To improve the accuracy of the morphological segmentation and exploit LSTM's advantage in capturing long term memory, we introduce a final architecture, extending the previous one even further. The Bidirectional Multi-Window LSTMs we propose in this section are illustrated in Figure 4.

In this architecture, the model first makes a forward pass to process the sequence in the normal order, and then adopts an additional backward pass to process it in reverse order. With these bidirectional passes, the network is able to learn even more fine-grained features from the input words and corresponding label sequences.

4.5 Training

For training, we turn to the log-likelihood functions introduced in Section 3.1. Consider the Multi-Window LSTM model as an example. Since our training goal in this architecture is to optimize the sum of the log probabilities in Eq. (2), we define a loss function corresponding to the sum of negative log-likelihoods of the correct label at each step. We apply standard stochastic gradient descent with mini-batches of training instances, and utilize dropout regularization, while clipping gradients element-wise. We rely on Beam Search (Cho et al. 2014) to perform inference on the sequence, i.e., we explore the search space by maintaining a list of the k most promising options when optimizing for Eq. (2). In particular, Beam Search repeatedly operates on a set of top-k most promising sequence predictions, iteratively considering the k-best list at time t as candidates to generate potential candidate sequences at time $t+1$.

5 Experimental Evaluation

5.1 Data and Evaluation

S&B data. This well-known data set by Snyder and Barzilay (2008) was derived from biblical text and consists of Hebrew, Arabic, Aramaic, and English terms together with their frequencies. Following Snyder and Barzilay (2008) and other studies, we focus on the morphologically segmented Hebrew and Arabic words, of which there are 6,192 each.

We follow the commonly used method to partition the data into training, development (tuning), and test splits (Ruokolainena et al. 2013). This involves sorting the inputs according to their frequency and assigning every fifth term starting from the first one into the test set and every fifth term from the second into the development set, while leaving the remaining data as training data. Finally, we randomly select the first 25%, 50%, 75% and 100% amount of data from the training set to carry out experiments with different training sizes.

Metric We evaluate morphological segmentation in terms of Precision, Recall and micro-averaged F-metric. The precision gives us the percentage of correctly assigned morpheme boundaries among all assigned boundaries, while the recall is defined as the percentage of correctly assigned morpheme boundaries among the reference boundaries. The F-metric is the geometric mean of precision and recall. For the F-metric, we employ token-based micro-averages (Ruokolainena et al. 2013).

5.2 Results and Analysis

We set the experimental parameters such as window size, learning rate, dropout rate, evaluation batch size, validation perplexity, threshold, etc. for the models using the development data. For instance, for 100% Hebrew training, we set 0.25 as the dropout to apply right after the encoder to an LSTM in the Window LSTM architecture. We set the encoder dropout as 0.3 and learning rate to be 0.0005 for Multi-Window LSTMs (MW-LSTM) architecture. And we set the encoder dropout as 0.2, learning rate as 0.00065, gradient clipping threshold at 10, decay rate as 0.5, and mo-

Hebrew	Method	Precision	Recall	F1
25%	S-Morf.	71.5	85.3	77.8
	Poon	78.7	73.5	75.9
	CRF	90.5	90.6	90.6
	LSTM	58.2	69.9	63.5
	MW-LSTM	91.7	85.0	88.2
	BMW-LSTM	**95.8**	**90.7**	**93.2**
50%	S-Morf.	82.1	81.8	81.9
	Poon	82.8	74.6	78.4
	CRF	**94.0**	**91.5**	**92.7**
	LSTM	52.5	76.2	62.2
	MW-LSTM	92.9	88.8	90.8
	BMW-LSTM	92.9	88.8	90.8
75%	S-Morf.	84.0	88.1	86.0
	Poon	83.1	77.3	80.1
	CRF	**94.0**	92.7	**93.4**
	LSTM	53.9	73.3	62.2
	MW-LSTM	91.1	92.4	91.8
	BMW-LSTM	92.4	90.4	91.4
	E. BMW-LSTM	93.1	**93.2**	93.1
100%	S-Morf.	85.3	91.1	88.1
	Poon	83.0	78.9	80.9
	CRF	94.9	94.0	94.5
	LSTM	60.3	65.7	62.9
	W-LSTM	91.0	92.8	91.9
	MW-LSTM	93.2	90.1	92.0
	BMW-LSTM	92.7	91.8	92.2
	E. BMW-LSTM	**95.2**	**95.2**	**95.2**

Table 1: Results for Hebrew

Arabic	Method	Precision	Recall	F1
25%	S-Morf.	78.7	79.7	79.2
	Poon	84.9	85.5	85.2
	CRF	**95.5**	**93.1**	**94.3**
	LSTM	74.6	68.7	71.5
	MW-LSTM	93.3	90.7	92.0
	BMW-LSTM	93.8	90.1	91.9
50%	S-Morf.	87.5	91.5	89.4
	Poon	88.2	86.2	87.5
	CRF	**96.5**	94.6	95.5
	LSTM	72.8	69.9	71.3
	MW-LSTM	95.0	92.9	94.0
	BMW-LSTM	95.0	92.6	93.8
	E. MW-LSTM	95.9	**96.1**	**96.0**
75%	S-Morf.	92.8	83.0	87.6
	Poon	89.6	86.4	87.9
	CRF	**97.2**	96.1	**96.6**
	LSTM	75.4	68.3	71.7
	MW-LSTM	95.6	94.1	94.9
	BMW-LSTM	96.9	93.2	95.0
	E. MW-LSTM	96.5	**96.4**	96.5
100%	S-Morf.	91.4	91.8	91.6
	Poon	91.7	88.5	90.0
	CRF	**98.1**	**97.5**	**97.8**
	LSTM	83.5	65.2	73.3
	W-LSTM	95.6	91.6	93.6
	MW-LSTM	97.1	96.1	96.6
	BMW-LSTM	96.0	95.0	95.5

Table 2: Results for Arabic

mentum as 0.01 for Bidirectional Multi-Window LSTMs (BMW-LSTM). Window sizes were chosen from {3,5,7}.

We compare our models with regular LSTMs, which consider characters instead of our windows as inputs. We also compare our models with supervised approaches that rely on human-designed features, including that of Ruokolainena et al. (2013), denoted CRFs for short, and of Poon, Cherry, and Toutanova (2009), denoted as Poon, as well as with the semi-supervised Morfessor variant (Kohonen, Virpioja, and Lagus 2010), denoted as S-Morf. Apart from this, "E." denotes that we use ensembles of 10 models, following Sutskever, Vinyals, and Le (2014).

From Tables 1 and 2, we see that character-based LSTMs do not yield satisfactory results, while our Window LSTM models perform significantly better at predicting morphological boundaries on the S&B data. This shows that the larger contexts that are models capture indeed are helpful. Thus, new hybrid models that operate on larger windows are a promising direction worth exploring further.

Our models are also able to outperform feature-rich models in some settings, but not universally. For instance, for Hebrew with 25% training size, the Bidirectional Multi-Window LSTM model outperforms the CRF method with relative improvements of 5.86%, 0.11%, and 2.87% for precision, recall, and F1, respectively. For Arabic, we outperform Poon, Cherry, and Toutanova (2009) and S-Morfessor, but our approach of learning from raw input characters mostly does not reach the level of scores obtained with the advanced feature engineering in the CRF models. Our window-based LSTM approach aims to automatically learn morphological structure from raw input characters instead of relying heavily on linguistic knowledge and manual feature engineering. Depending on the complexity of the morphology of a language, adding additional features as ingredients may still be necessary to obtain state-of-the-art results. Another promising direction is to perform inference in our models using more sophisticated decoding strategies than a simple beam search.

In terms of the training data sizes, we see that our model can perform well even on smaller percentages of the training data. This is quite remarkable given that they do not benefit from human-engineered features and thus need to induce good representations from the data. Still, there are some lower bounds on the number of training examples required for this to succeed. While the test sets from the Morpho Challenges [1] are not publicly available, we ran experiments by partitioning available data into smaller training, development, and test sets for English, Finnish, and Turkish. With just 500-1000 training examples, our models sometimes succeeded in outperforming CRFs on all three languages, but with so little training data, this varied significantly between training runs. As expected, learning good features reliably

[1] http://research.ics.aalto.fi/events/morphochallenge/

thus still requires more training data than supervised learning from human-engineered features. However, our experiments show that not very much is needed. Even just using 25% of the available training data from the modestly-sized S&B data sets for Hebrew and Arabic suffices to achieve F-scores above 90%.

6 Conclusion

In this paper, we have proposed three types of novel window-based LSTM neural network architectures to automatically learn sequence structures and predict morphological boundaries in raw words, without additional feature engineering. Our models achieve good results on a number of languages without relying on linguistic knowledge. Indeed, our architectures are fairly general and thus we hope to investigate their performance on further kinds of sequence labeling problems.

7 Acknowledgments

This work was supported in part by the National Basic Research Program of China Grants 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grants 61033001, 61361136003, 61450110088.

References

Cho, K.; van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Creutz, M., and Lagus, K. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, 21–30. Association for Computational Linguistics.

Creutz, M.; Hirsimäki, T.; Kurimo, M.; Puurula, A.; Pylkkönen, J.; Siivola, V.; Varjokallio, M.; Arisoy, E.; Saraçlar, M.; and Stolcke, A. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing (TSLP)* 5(1):3.

de Melo, G. 2014. Etymological Wordnet: Tracing the history of words. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC 2014)*. Paris, France: ELRA.

Dyer, C.; Ballesteros, M.; Ling, W.; Matthews, A.; and Smith, N. A. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

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

Green, S., and DeNero, J. 2012. A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 146–155. Association for Computational Linguistics.

Harris, Z. 1951. *Methods in structural linguistics*. University of Chicago Press.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kohonen, O.; Virpioja, S.; and Lagus, K. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, 78–86. Association for Computational Linguistics.

Poon, H.; Cherry, C.; and Toutanova, K. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of NAACL-HLT 2009*, 209–217. Association for Computational Linguistics.

Rao, K.; Peng, F.; Sak, H.; and Beaufays, F. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Proceedings of ICASSP*.

Ruokolainena, T.; Kohonena, O.; Virpiojaa, S.; and Kurimob, M. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. *CoNLL-2013* 29.

Ruokolainena, T.; Kohonenb, O.; Virpiojab, S.; and Kurimoa, M. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proceedings of EACL 2014*, 84.

Sirts, K., and Goldwater, S. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics* 1:255–266.

Snyder, B., and Barzilay, R. 2008. Cross-lingual propagation for morphological analysis. In *AAAI*, 848–854.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2014. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*.

Wang, M., and Manning, C. D. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*.

Wang, L.; Liu, K.; Cao, Z.; Zhao, J.; and de Melo, G. 2015. Sentiment-aspect extraction based on restricted boltzmann machines. In *Proceedings of ACL 2015*, 616–625.