Neural Sequence-to-sequence Learning of Internal Word Structure

Tatyana Ruzsics **Tanja Samardžić**
CorpusLab, URPP Language and Space,
University of Zurich, Switzerland
{tatiana.ruzsics, tanja.samardzic}@uzh.ch

Abstract

Learning internal word structure has recently been recognized as an important step in various multilingual processing tasks and in theoretical language comparison. In this paper, we present a neural encoder-decoder model for learning canonical morphological segmentation. Our model combines character-level sequence-to-sequence transformation with a language model over canonical segments. We obtain up to 4% improvement over a strong character-level encoder-decoder baseline for three languages. Our model outperforms the previous state-of-the-art for two languages, while eliminating the need for external resources such as large dictionaries. Finally, by comparing the performance of encoder-decoder and classical statistical machine translation systems trained with and without corpus counts, we show that including corpus counts is beneficial to both approaches.

1 Introduction

One of the most obvious structural differences between languages is the variation in the complexity of internal word structure. In some languages, such as English, words are relatively short and morphologically less complex. In other languages, such as Chintang in Example 1[1], words tend to be long and encapsulate rather rich structure. The Chintang verb *thaptakha* in Example 1 consists of a number of morphemes expressing the imperative mode, aspect and deixis. The information expressed by a single Chintang verb requires several words in English, as it can be seen in the glosses.[2] and in the translation.

Example 1

a.			cuwa thaptakha		
b.	cuwa	thapt	-a	-khag	-a
c.	water	move across	-IMP	-see	-IMP [2sS]
d.	Bring some water over here!				

The variation in word structure is observed even in common categories such as plural, which is typically part of a word, but expressed using different structures. The items in Example 2[3] show three different structural types associated with expressing plural across languages.

Example 2

Type		Sg.	Pl.
Suffix	Turkish	*ev* 'house'	*ev-ler* 'houses'
Prefix	Swahili	*m-toto* 'child'	*wa-toto* 'children'
Reduplication	Malay	*anak* 'child'	*anak-anak* 'children'

With the spread of natural language processing to a wider range of languages, learning internal word structure becomes increasingly important for developing practical applications. Analysis of internal word structure, usually termed morphological segmentation, has been shown helpful in tasks such as machine translation (Dyer et al., 2008; Narasimhan et al., 2014), speech processing (Creutz et al., 2007) and parsing (Seeker and Çetinoglu, 2015). Additionally, there is a growing interest in automatic learning of morphological segmentation for the purpose of theoretical lan-

[1] The example is adapted from (Stoll et al., In press)

[2] Cf. Leipzig Glossing Rules at https://www.eva.mpg.de/lingua/resources/glossing-rules.php

[3] The examples are adapted from (Eifring and Theil, 2005)

Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 184–194,
Vancouver, Canada, August 3 - August 4, 2017.

guage comparison (Bentz et al., 2017). In this context, it becomes particularly important to be able to process a wide variety of languages, for which the available data sets consist of small, annotated corpora.

In the present work, we cast the task of morphological segmentation as supervised neural sequence-to-sequence learning over characters. Our goal is to define a method for automatic segmentation that can be easily ported across languages, taking advantage of the relatively small, manually analyzed corpora increasingly available in the linguistic community. Our approach therefore needs to rely only on the data available in an annotated corpus.

We follow the line of research based on the soft-attention encoder-decoder paradigm (Bahdanau et al., 2014). This paradigm was recently applied to the task of canonical segmentation by Kann et al. (2016). It was shown to achieve the state-of-the-art performance when combined with a neural re-ranker method that employs additional external dictionary information (Kann et al., 2016). Our approach improves the results achieved by Kann et al. (2016) in case of Indonesian and German languages eliminating at the same time the need for resources outside of annotated corpora.

2 Related Work

The task of the morphological segmentation can be defined in two ways, illustrated with the Chintang verb from Example 1:

a. Surface segmentation:
thapta**kh**a → thapt-a-**kh**-a

b. Canonical segmentation:
thapta**kh**a → thapt-a-**khag**-a

The term *surface* or *allomorph* segmentation is used by Creutz and Linden (2004) to refer to the analysis where the input word is segmented into substrings without any further string transformation. This definition is most widely applied in computational processing; it is, however, too simplistic for the majority of languages. It does not allow, for instance, to identify *-es* in *bus-es* and *-s* in *car-s* as two variants of the same English plural marker.

More recently, the term *canonical segmentation* was used by Cotterell et al. (2016) to refer to the same definition that was termed *morpheme* segmentation by Creutz and Linden (2004). In this case, a more abstract internal word structure is learned by transforming the resulting substrings into their canonical forms.

While a great variety of methods has been proposed for surface segmentation, canonical segmentation, which is address in our work, has started being addressed only recently.

The task of surface morphological segmentation is traditionally approached using finite-state technology, such as OpenFst library (Allauzen and Riley, 2012) and OpenGrm Thrax Grammar Compiler library (Roark et al., 2012), with sequence modeling used to disambiguate the finite-state output (Heintz, 2008).

Another line of research has addressed surface segmentation with unsupervised algorithms (MORFESSOR (Creutz and Lagus, 2002), MORFESSOR CAT-MAP (Creutz and Lagus, 2005)), and, more recently, with semi-supervised approaches (Poon et al., 2009; Kohonen et al., 2010)). Narasimhan et al. (2015) include semantic information in the unsupervised learning, staying at the level of surface segmentation. Their approach is extended by Bergmanis and Goldwater (2017), who make a step towards canonical segmentation, proposing a method to generalize over spelling variants of functionally similar morphemes.

Supervised approaches to learning morphological structure are rather rare. Ruokolainen et al. (2013) apply conditional random fields algorithm, used for different sequence classification tasks, to the task of surface segmentation. Their CRF-MORPH system tags each character of a morphologically complex word with one of the tags 'B' for the beginning, 'M' middle, and 'E' end of a segment, and 'S' for a single character segment. The CHIPMUNK model of Cotterell et al. (2015) based on a semi-Markov model extends the CRF-MORPH approach by adding features from standalone dictionaries and affix lists.

Canonical segmentation is tackled by Cotterell et al. (2016), who develop a log-linear model on conjunction of a finite-state transduction model for modeling orthographic changes and a semi-Markov segmentation model.

Recently, neural network models achieved state-of-the-art results for both types of segmentation tasks. Wang et al. (2016) applied window LSTM model for surface segmentation. Kann et al. (2016) improved the results by Cotterell

et al. (2016) on canonical segmentation by applying the encoder-decoder RNN framework. Kann et al. (2016) achieve the current state-of-the-art for canonical segmentation by re-ranking the output of the encoder-decoder system. The re-ranking component is a multilayer perception run on the morphemes embeddings (Kann et al., 2016). The morphemes embedding used for this re-ranking model are calculated using additional information from the Aspell dictionaries. We follow Kann et al. (2016) in using the encoder-decoder RNN framework, but we do not use any external resources. Instead of that, we extract and exploit more information from the training corpus.

Our approach is in the spirit of the "shallow fusion" approach to machine translation of Gulcehre et al. (2017). Like Gulcehre et al. (2017), we integrate a language model into an encoder-decoder framework. There are, however, several important differences.

First of all, the role of the language model is different. Integrating a language model allows Gulcehre et al. (2017) to augment the parallel training data with additional monolingual corpora on the target side. In this way, they add new information about sequencing, not captured in training on parallel data alone. Both components of their system are trained on the same kind of units — characters. As opposed to this, we use a language model to extract more information from the parallel data. We add new information by training the system at two levels: the basic encoder-decoder component is trained on character sequences and the language model component is trained on the sequences of morphemes. In the case of Gulcehre et al. (2017), the use of a language model is motivated by the fact that external monolingual target-side data is almost always universally available. The situation is reversed for the task of morphological segmentation: morphologically segmented corpora are produced manually by experts in the process of linguistic analysis and they tend to be small and expensive. Our approach is motivated by the need to extract as much information as possible from relatively small target-side data sets.

Second, we add a third component to our model which controls for the difference in characters length between the input word string and the output segmentation string. This helps overcome language model preference for a short output.

Last, while Gulcehre et al. (2017) use a language model implemented with recurrent neural networks, we employ a statistical language model, which is better adapted to our settings with small data sets.

Finally, we note that corpus frequencies are not used in previous supervised approaches, but that systems are trained on word types (training data consist of a list of word types and a segmentation for each type). In the present work, we study token versus types training set up and how this difference affects the performance of statistical models.

3 Model Description

Given an input sequence, such as the Chintang sentence in Example 1 (line a.), we produce a canonical segmentation (line b.), where we recognize that the sequence *kh* in the surface form is an instance of the light verb *khag*.

We follow the notations of Kann et al. (2016) in the formalization of the task of canonical segmentation. First, we define two discrete alphabets, Σ of the surface symbols and Σ_{can} of the canonical symbols. For many languages these two alphabets coincide, for example in the case of English they consist of 26 letters of the Latin alphabet. In the case of Chintang, these alphabets are different: the surface symbols express more specific pronunciation features. Our task is to learn a mapping from a surface word form $w \in \Sigma^*$ (e.g., w='thaptakha'), to its canonical segmentation $c \in \Omega^*$ (e.g., c=thapt|-a|-khag|-a). We define $\Omega = \Sigma_{can} \cup \{|\}$, where the symbol '|' marks segmentation boundaries.[4]

To learn the mappings, we combine the general sequence transformation framework — known as the encoder-decoder RNN — trained on a character level with a language model trained on morphemes. Note that a kind of a language model over characters is implicitly included as a part of the decoder in the character-based encoder-decoder RNN. The language model in our approach is trained over higher level units, providing additional information about the sequences. This, however, poses a challenge for its integration in the general framework. We tackle this problem with our "synchronization" method applied at the decoding stage: the segmentation hypotheses

[4]Furthermore, specifically to the Chintang corpus, the canonical symbols additionally use a dash element '-' to distinguish between suffixes, prefixes and roots. We do not exclude this symbol in our experiments.

are expanded and scored using a log-linear combination of (a) scores from a lower-level encoder-decoder model and (b) higher-level scores of the language model. The fusion of the scores is triggered only at the segmentation boundaries.

In this section, we first review the encoder-decoder RNN framework. Then, we present our fusion approach for integrating a language model into the encoder-decoder system.

3.1 Background: Standard Encoder-Decoder Set-up (cED)

The canonical segmentation problem fits the general sequence-to-sequence framework, that is mapping a variable-length sequence to another variable-length sequence. In machine translation, a relatively standard way to perform this task is using encoder-decoder RNN (Cho et al., 2014; Sutskever et al., 2014), extended with a bidirectional encoder and attention mechanism of Bahdanau et al. (2014). For the task of canonical segmentation, the input and the output to the encoder-decoder model is represented as a sequence of characters separated by spaces. Formally, we use the following set up.

The encoder RNN processes the input sequence of vectors, $X = (x^1, \ldots, x^{n_x})$, into a sequence of vectors representing hidden states:

$$h_t = f(h_{t-1}, x_t), \quad t = 1, \ldots, n_x \qquad (1)$$

where f stands for gated recurrent units (Cho et al., 2014). The decoder RNN is conditioned on the information produced by the encoder to generate the output sequence $Y = (y^1, \ldots, y^{n_y})$. Specifically, the decoder RNN models a conditional probability at each step as a function of previous output, current decoder hidden state and current context vector:

$$p(y_t|y_1, \ldots, y_{t-1}, X) = g(y_{t-1}, s_t, c_t), \qquad (2)$$

$$s_t = f(s_{t-1}, y_{t-1}, c_t), \quad t = 1, \ldots, n_y \qquad (3)$$

The context vector c_t is computed at each step as a weighted sum of the hidden states:

$$c_t = \sum_{k=1}^{n_x} \alpha_{tk} h_k, \qquad (4)$$

where the weights are calculated by an alignment model which scores how well the inputs around the position k and the output at the position t match. Intuitively, the decoder produces an output element one at a time, each time focusing (putting attention) on a different part of the input sequence in order to gather the details that are required to produce the next output element.

We use the bidirectional setting of the encoder model (Bahdanau et al., 2014): the hidden state h_t for each time step is obtained by concatenating a forward and backward state $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$. This means that the hidden state contains the summaries of both the preceding elements and the following elements. We refer to this standard framework as cED.

3.2 Integrating a Morpheme Language Model (LM) into cED

Before the integration, we assume that cED system and language model LM have been trained separately. The cED system is trained on character sequences in a parallel corpus where the source side consists of unsegmented words and the target side consist of the aligned canonically segmented words. During the training the cED model learns conditional probability distribution over the character sequences (2). The LM is trained over morpheme sequences on the target side of the corpus and scores how likely a given sequence is in a given language.

We can find the most probable segmentation using a beam search algorithm guided by "synchronized" character-level cED and morpheme-level LM scores. At each time step t, the cED system computes a score for each possible next character y_t in the vocabulary Ω as a continuation of the segmentation hypothesis from the previous step $\{(y_1 y_2 \ldots y_{t-1})^i\}$, $i = 1, \ldots, K$ where K is the beam size (how many best scored segmentation hypothesis we keep from each step). This score is a logarithm of the probability (2). Then, each possible continuation $\{(y_1 y_2 \ldots y_{t-1})^i y_t\}$, $y_t \in \Omega$, $i = 1, \ldots, K$ gets a score which is a sum of cED scores for each character, that is, a sum of the scores for a hypothesis from a previous time step $(y_1 y_2 \ldots y_{t-1})^i$ and a score for the next character y_t. Thus we get a set of $|\Omega| \times K$ new hypothesis of length $|t|$ together with their respective scores. All these new hypotheses at the step t are then sorted according to their respective scores, and the top K ones are selected as candidates for the expansion at the next time step.

In order to guide the described beam decoding with the LM scores we perform a "synchronization". Specifically, we continue the beam search based on the character cED scores till the step $s1$ where all the segmentation hypothesis $\{(y_1 y_2 \dots y_{s1})^i\}$ $i = 1, \dots, K$ end with a boundary symbol. The boundary symbol can be either end of word symbol '$< /w >$' or a segmentation boundary symbol '|'. At this step, we re-score the segmentation hypotheses with a weighted sum of the cED score and the LM score:

$$\begin{aligned}\log p(y_{s1}|y_1, \dots, y_{s1-1}, X) \\ = \log p_{cED}(y_{s1}|y_1, \dots, y_{s1-1}, X) \\ + \alpha_{LM} \log p_{LM}(y_1, \dots, y_{s1-1}) \quad (5)\end{aligned}$$

At this step, $y_1, \dots, y_{s1}$ is considered a sequence of $s1$ characters by the cED system and $y_1, \dots, y_{s1-1}$ (without the last boundary symbol) is considered one morpheme by the LM.

From this synchronization point $s1$ we continue to expand the re-scored hypothesis $\{(y_1 y_2 \dots y_{s1})^i\}$ $i = 1, \dots, K$ at the next time step using again only cED scores. We continue this process until we get to the next synchronization point $s2$ where all the hypothesis $\{(y_1 y_2 \dots y_{s2})^i\}$ $i = 1, \dots, K$ end with a boundary symbol. After rescoring them with a weighted sum of cED and LM we continue this process again till the next synchronization point. The decoding process ends at a synchronization point where the last symbol of the best scored hypothesis (using the combined cED and LM score) is an end-of-word symbol.

The described decoding process therefore scores the segmentation hypotheses at two levels: normally working at the character level with cED scores and adding the LM scores only when it hits a boundary symbol. In this way, the LM score helps to evaluate how probable the last generated morpheme is based on the morpheme history, that is the sequence of morphemes generated at the previous synchronization time steps.

3.3 The Length Constraint

It is well known that language models give higher preference to shorter sequences. This becomes an issue in the proposed fused model described above: at the synchronization points high LM scores tend to stop further hypothesis expansion. For example, only the first segment can be generated as a model output if it happens to be a frequent standalone word. This leads to favoring segmentation predictions where the output is shorter than the input, which is rarely plausible in segmentation. Our early experiments confirmed this intuition, therefore we consider the length constraint component to be an integral part of the language model inclusion and we do not report experiments without this component.

To deal with the length issue, we add a "length constraint" component LC. The LC score is based on the difference in character length between the input word and its segmentation hypothesis. To synchronize the LC score with LM scoring process described before we assign it only at the synchronization time steps and attach it to the boundary symbol. Therefore, the LC score, combined with the LM score, helps to evaluate how probable is the last generated morpheme given the sequence of morphemes generated at the previous steps.

Assume that the input word is $X = x_1 \dots x_n$ and the produced segmentation hypothesis at the first synchronization step $s1$ is $y_1 \dots y_{s1}$ where y_{s1} is a boundary symbol. Then the LC score assigned to the morpheme $y_1 \dots y_{s1-1}$ and attached to the boundary symbol y_{s1} is calculated as the negative value of the absolute difference between the morpheme length and input word length divided by the the input length: $LC(y_1 \dots y_{s1-1}) = -(|y_1 \dots y_{s1-1}| - |X|)/|X| = -|s1 - 1 - n|/n$. At the next synchronization point $s2$ the LC score is calculated using the length of the next produced segment: $LC(y_{s1+1} \dots y_{s2-1}) = -(|y_{s1+1} \dots y_{s2-1}| - |X|)/|X|$. In a general case, the LC score for the last generated segment σ_i can be expressed as

$$LC(\sigma_i) = -(|\sigma_i| - |X|)/|X| \quad (6)$$

Note that boundary symbols are excluded for the segments length calculation.

The intuition behind the LC score is that it gives a contribution to the total score of a segmentation hypothesis showing how different the length of the so far produced hypothesis is compared to the length of the input word. The characters in the canonical segments tend to be either inserted or deleted compared to their surface form equivalents, therefore we measure LC score using an absolute difference in the length. The higher the absolute value of the difference between the input and the hypothesis, the higher the penalty.

With the inclusion of the LC score for the length

control the total score of our fusion model becomes:

$$\begin{aligned}\log p(y_{s1}|y_1,\ldots,y_{s1-1},X) \\ = \log p_{cED}(y_{s1}|y_1,\ldots,y_{s1-1},X) \\ + \alpha_{LM}\log p_{LM}(y_1,\ldots,y_{s1-1}) \\ + \alpha_{LC}LC(y_1,\ldots,y_{s1-1}) \quad (7)\end{aligned}$$

where the weights α_{LM} and α_{LC} are optimized on a development set.

4 Data and Experiments

In this section, we first give a description of the corpora that we employ for the experiments. Then, we discuss the experimental setup for our model. Finally, we discuss the different configurations of the corpus we employ to explore encoder-decoder model behavior with and without corpus frequencies.

4.1 Corpora

We run our experiments on the datasets for English, German and Indonesian released by Cotterell et al. (2016).[5] The corpus for each language is constructed based on the 10,000 forms selected at random from a uniform distribution over types. This data is further used to sample 5 splits into 8000 training forms, 1000 development forms and 1000 test forms. Following Cotterell et al. (2016) and Kann et al. (2016), we report the results on each of the 5 splits for all three languages.

In addition to these sets, we use a manually segmented and glossed corpus of Chintang (Bickel et al., 2004-2015), a language that features a high degree of synthesis and free prefix ordering (Bickel et al., 2007). The total corpus size of 955,025 word tokens makes the Chintang corpus an exceptional resource for the task of morphological segmentation. As discussed in Section 2, the target segmented data is not easily available and corpora of this size are not likely to be developed for many langauges. We are interested in experimenting with a realistic setup, therefore we use around 150,000 tokens out of the total corpus size. This set is divided into the training set (around 100,000 word tokens) and development and test set (around 25,000 word tokens each).

The data set taken from Cotterell et al. (2016) allows us a comparison of our system with the previous state-of-the-art. The Chintang set allows us to run the segmentation models in different training regimes, with and without corpus frequencies, and therefore to assess the influence of the corpus counts on the performance.

4.2 Tools

We combine the three different components, cED, LM and LC, into a single fused model using the SGNMT (syntactically guided neural machine translation) framework of Stahlberg et al. (2016).[6] This framework provides an elegant solution to decomposing an encoder-decoder system into three components: training, decoding and scoring.

The training module implements the encoder-decoder model with attention mechanism using the Blocks framework built on top of Theano.[7]. We employ this implementation for the cED model.

The scoring component of SGNMT consists of predictor modules, which define scores over the target vocabulary given the current internal predictor state, the history, the source sentence, and external side information. Predictors can be combined with other predictors to form complex decoding tasks. In the case of our model, we use three predictors: cED, LM and LC.

The decoding component is represented by the decoder modules which are search strategies that traverse the space spanned by the predictors. We use a beam search module.

We train the language model LM over morpheme sequences using SRILM toolkit.[8]. The model is trained on the target side of the parallel corpus, i.e. the canonical segmentations.

The weights of the predictors are optimized using MERT (Och, 2003). This is a standard optimization routine in statistical machine translation which searches for the weights of the model components by directly maximizing the performance of the system on a development set. We use the Z-MERT tool[9] in our implementation. The code is available on our GitHub account.[10]

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

[6] http://ucam-smt.github.io/sgnmt

[7] https://github.com/mila-udem/blocks

[8] http://www.speech.sri.com/projects/srilm/

[9] http://www.cs.jhu.edu/~ozaidan/zmert/

[10] https://github.com/tatyana-ruzsics/uzh-corpuslab-morphological-segmentation

4.3 Experimental Setup

Baseline and comparison As a baseline, we use the basic component of our model (cED), an ensemble of 5 character-level attention encoder-decoder models with the hyperparameters described below.

We also compare the encoder-decoder model to the character-level statistical machine translation (cSMT). This approach is a natural choice in machine translation with small training sets, but no results have been reported so far for the task of canonical segmentation. We used the Moses toolkit with the following settings: distortion is disallowed and build-in MERT optimization is used to optimize the translation model and language model.

As a reference, we compare our results to the state-of-the-art neural re-ranker model of Kann et al. (2016) Note, however, that the results cannot be directly compared since Kann et al. (2016) use extra training material in the form of external dictionaries.

Evaluation Since our method is intended to be used for processing corpora, the evaluation is performed at the level of word tokens using accuracy of the full segmentation. In addition, we evaluate the performance on subsets of test words. Besides the seen words, we distinguish between two kinds of test words that are not seen in the training corpus: a) new combinations of morphemes already seen during training and b) words that contain unseen morphemes.

Hyperparameters The bidirectional encoder consists of a forward and a backward recurrent RNN each having 100 hidden units. The decoder also has 100 hidden units. The dimensionality of the character embedding is 300. We use a minibatch stochastic gradient descent (SGD) algorithm together with Adadelta to train each model. Each update direction is computed using a minibatch of 20 training examples. At decoding, we use a beam search with a beam size of 12 to find the segmentation that approximately maximizes the conditional probability. All the described hyperparameters are the same as in the work of Kann et al. (2016).

Initialization of all weights (encoder, decoder, embeddings) to the identity matrix and the biases to zero (Le et al., 2015) results in a very fast convergence rate compared to other initializations. We train a single model for 20 epochs with an early stopping based on the development set performance. We also shuffle the training data between the epochs.

Finally, we use an ensemble of five encoder-decoder models with different random initializations. We shuffle the training data for each of the model using different seed value. The ensemble model is based on a combined score from all 5 models and is used to guide the decoding process.

Following earlier experiments, we use morpheme 3-gram language model and apply Kneser-Ney smoothing.

As the objective for the MERT weight optimization we use accuracy on the development set.

Tokens vs Types The Chintang corpus allows us to assess the influence of the corpus counts on the performance of the models used for canonical segmentation. In these experiments, we run only the two baseline models, cSMT and cED (without our language model component), in order to evaluate directly the relevance of such corpus signal to these two training paradigms.[11]

We run each model, cSMT and cED, in two regimes. In the first, type regime, we train the models using a parallel corpus which consists of word types, i.e. unique pairs of surface form and its canonical segmentation. The size of such type corpus is around 21,000 word forms. In the second regime, we train the models using a parallel corpus where each pair of surface form and its canonical segmentation appears as many time as the corresponding word appears in the corpus. The size of the token-based corpus is then 100,000 tokens.

In the type regime, the amount of training examples is substantially smaller than in the token regime. In order to make the comparison between the token regime and the type regime more fair in terms of the amount of training and testing data, we add one more experiment. Specifically, we train the cED model in the type regime using the same number of iterations as in the token regime: 100,000 iterations. In this way, each word type is seen multiple times both in the type and the token regime. The difference is that all the types are equally frequent in the type regime, while we observe their natural text frequency in the token

[11] One way to include language model into SMT would be the n-best list reranking which is not exactly the same as our synchronization approach that guides the decoding process. Integrating our fusion approach for a higher-level language model into MOSES is not trivial, and the systems are not comparable without that.

regime.

5 Results and Discussion

The performance of our fused model cED+LM together with the two baseline models is reported for English, German, Indonesian in Table 1.[12] We show the average results over five splits for each language along with the standard deviation (in brackets). Additionally, we present the results on the words not seen in the training which make up for 99% of the test sets in this corpora.

Our fused approach gives an improvement from 1% to 4% over the stronger cED baseline. The bigger improvement for Indonesian could be attributed to the regular patterns of orthographic changes which appear on the segmentation boundaries. In the category of unseen data, the LM component helps to correct errors for the words consisting of new combinations of seen morphemes. In case of Indonesian, around 80% of new words (an average over five splits) belongs to this category, while its share is only around 25% for German and English. The overall lower performance for English and German thus might be due to the less regular patterns and more unseen roots in the training data.

We observe that out of the two baseline models, cED and cSMT, cED performs on average better although their behavior is very similar with a difference of only 1% in accuracy in the case of German and Indonesian.

For reference, we also show the results of the joint model of Cotterell et al. (2016) and the state-of-the-art neural reranker model of Kann et al. (2016) which are available for these data sets. We can see that our approach (cED+LM) gives an improvement of 1% for German and 2% for Indonesian over the state-of-the-art performance while we do not employ extra information from external dictionaries. Note that the languages for which we improve the state of the art are morphologically richer than English.

Table 2 shows the cED and cSMT model on Chintang in two training regimes, word types and word tokens. We also report the results for comparative setting of the type-based regime.

[12]We obtained significantly better results for cED model than those reported in Kann et al. (2016). We speculate that the difference might be due to the shuffling of the data between the training epochs and early stopping based on the validation set performance.

We observe that the corpus-wide training based on word tokens increases the overall performance for both, the cED and cSMT models. The observed improvement can be partially explained by the fact that our evaluation is token based: we count the same result as many times as it appears in the test set. Nevertheless, the score is informative because it shows the coverage over the whole corpus. Additionally, our comparable setting shows that seeing a segmentation for a word type multiple times is not what helps learning. What is beneficial is knowing the actual distribution in the corpus.

It can be seen in Table 2 that cSMT outperforms cED in the token regime. One possible explanation for this outcome is that the inclusion of the word counts helps to learn the character alignments better. This explanation would be in line with the results of Aharoni and Goldberg (2017), who showed that using pretrained character-level statistical alignments to guide the encoder-decoder network in training time can help to improve over the end-to-end soft attention approach for morphological inflection generation task, which also falls into the category of a more general sequence transduction task.

Regarding the different subsets of the test data, the highest improvement on unseen words with new morphemes is achieved by the cED model, while the cSMT model generalizes better in the category of unseen words that consist of new combination of seen morphemes. Both models behave similarly on seen words with the cSMT being slightly better. This leads to an overall best performance of the cSMT model, since the category of seen words has the largest weight among all the word tokens in the test data.

6 Conclusion and Future Work

We presented a neural model based on character level encoder-decoder framework for morphological canonical segmentation in a low-resource setting. The model is fused with a language model over morpheme segments and length control model. Our approach gives higher results than the state-of-the-art approach to canonical segmentation for languages with more morphology, Indonesian and German, while using only the information contained in the training corpus.

Future work may include a development of a single canonical segmentation model where the

		Error Rate (%)				
		Types Regime				
		cED+LM	cED Baseline	cSMT Baseline	Joint*	cED+RR*
English	Total	**0.21** (.01)	0.22 (.01)	0.27 (.02)	0.27 (.02)	0.19 (.01)
	New comb.	0.15 (.03)	0.24 (.01)	-	-	-
	New morph.	0.23 (.01)	0.20 (.01)	-	-	-
German	Total	**0.19** (.00)	0.23 (.02)	0.24 (.02)	0.41 (.03)	0.20 (.01)
	New comb.	0.11 (.02)	0.33 (.03)	-	-	-
	New morph.	0.21 (.01)	0.20 (.01)	-	-	-
Indonesian	Total	**0.03** (.02)	0.07 (.01)	0.06 (.01)	0.10 (.01)	0.05 (.01)
	New comb.	0.02 (.03)	0.06 (.01)	-	-	-
	New morph.	0.09 (.03)	0.09 (.03)	-	-	-

Table 1: Performance on the task of canonical segmentation for English, German and Indonesian. Type-based regime. cED+LM - character based encoder-decoder model fused with morpheme based language model. Baseline models: cED - character based encoder-decoder model, cSMT - character based statistical machine translation model. For reference only: Joint* - model of Cotterell et al. (2016), cED+RR* - model of Kann et al. (2016), not directly comparable since based on external dictionary information)

	No. of	Correct predictions (%)				
		Types Regime			Tokens Regime	
		cED	cSMT Baseline	cED Compar.	cED	cSMT Baseline
Total	24,606	0.19	0.18	0.23	0.16	**0.14**
Seen words	19,920	0.13	0.12	0.18	0.08	**0.07**
New comb.	3,959	0.44	**0.41**	0.42	0.47	**0.41**
New morph.	727	0.53	0.57	0.60	**0.48**	0.56

Table 2: Performance on the task of canonical segmentation for Chintang. Type-based vs token-based training regime. cED - character based encoder-decoder model, cSMT - character based statistical machine translation model. Comparative setting for cED: training in types regime for the same number of iterations as in the individual setting of token regime.

optimization of model components is performed using neural approaches.

Another idea relevant to explore in future work is to consider the networks that are designed to be strong at character copying which is the most common operation in string transduction tasks such as morphological segmentation, morphological reinflection and normalization (Gu et al., 2016; See et al., 2017; Makarov et al., 2017).

We also analyzed the effect of incorporating corpus counts for the purpose of training statistical and neural models for canonical segmentation. The results show that incorporating counts as they are seen in the corpora is beneficial for such task for both types of models. Our findings suggest that training sets including real word counts should be further developed for this and similar tasks and would be beneficial for development of future models.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Cyril Allauzen and Michael Riley. 2012. A pushdown transducer extension for the openfst library. In *Implementation and Application of Automata - 17th International Conference, CIAA 2012, Porto, Portugal, July 17-20, 2012. Proceedings*. pages 66–77.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Christian Bentz, Dimitrios Alikaniotis, Tanja Samardi,

and Paula Buttery. 2017. Variation in word frequency distributions: Definitions, measures and implications for a corpus-based language typology. *Journal of Quantitative Linguistics* pages 128–162. https://doi.org/10.1080/09296174.2016.1265792.

Toms Bergmanis and Sharon Goldwater. 2017. From segmentation to analyses: a probabilistic model for unsupervised morphology induction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain.

Balthasar Bickel, Goma Banjade, Martin Gaenszle, Elena Lieven, Netra Prasad Paudyal, Ichchha Purna Rai, Manoj Rai, Novel Kishore Rai, and Sabine Stoll. 2007. Free prefix ordering in chintang. *Language* 83(1):43–73. https://muse.jhu.edu/article/214599.

Balthasar Bickel, Sabine Stoll, Martin Gaenszle, Novel Kishor Rai, Elena Lieven, Goma Banjade, Toya Nath Bhatta, Netra Paudyal, Judith Pettigrew, Ichchha P. Rai, and Manoj Rai. 2004-2015. Audiovisual corpus of the chintang language, including a longitudinal corpus of language acquisition by six children. http://www.mpi. nl/DOBES.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*. pages 103–111.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 164–174.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 664–669.

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 Trans. Speech Lang. Process.* 5(1):3:1–3:29.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*. Association for Computational Linguistics, pages 21–30.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proc. International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR-05)*. Espoo, Finland, pages 106–113.

Mathias Creutz and Krister Linden. 2004. Morpheme segmentation gold standards for finnish and english.

Christopher Dyer, A Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *In ACL-HLT*.

H. Eifring and R. Theil. 2005. *Linguistics for Students of Asian and African Languages*. [available at `http://www.uio.no/studier/emner/hf/ikos/EXFAC03-AAS/h05/larestoff/linguistics/`], Universitetet i Oslo.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR* http://arxiv.org/abs/1603.06393.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech and Language* https://doi.org/http://doi.org/10.1016/j.csl.2017.01.014.

Ilana Heintz. 2008. Arabic language modeling with finite state transducers. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA, Student Research Workshop*. pages 37–42.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 961–967.

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

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

Peter Makarov, Tatyana Ruzsics, and Simon Clematide. 2017. Align and copy: Uzh at sigmorphon 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Vancouver, Canada.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics* 3:157–167.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*. pages 160–167.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 209–217.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The opengrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, ACL '12, pages 61–66.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 29–37.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

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

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Sabine Stoll, Jekaterina Mazara, and Balthasar Bickel. In press. The acquisition of polysynthetic verb forms in chintang. In Michael Fortescue, Marianne Mithun, and Nicholas Evans, editors, *Handbook of Polysynthesis*, Oxford University Press, Oxford.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. pages 2842–2848.