Tailoring Neural Architectures for Translating from Morphologically Rich Languages

Peyman Passban, Andy Way, Qun Liu
ADAPT Centre
School of Computing
Dublin City University, Ireland
firstname.lastname@adaptcentre.ie

Abstract

A morphologically complex word (MCW) is a hierarchical constituent with meaning-preserving subunits, so word-based models which rely on surface forms might not be powerful enough to translate such structures. When translating *from* morphologically rich languages (MRLs), a source word could be mapped to several words or even a full sentence on the target side, which means an MCW should not be treated as an atomic unit. In order to provide better translations for MRLs, we boost the existing neural machine translation (NMT) architecture with a double-channel encoder and a double-attentive decoder. The main goal targeted in this research is to provide richer information on the encoder side and redesign the decoder accordingly to benefit from such information. Our experimental results demonstrate that we could achieve our goal as the proposed model outperforms existing subword- and character-based architectures and showed significant improvements on translating from German, Russian, and Turkish into English.

Title and Abstract in Farsi (Persian)

بهبود معماری مدل‌های مبتنی بر شبکه‌های عصبی برای ترجمه از زبان‌های باساختار مورفولوژیکی پیچیده

کلمات با الگوهای پیچیده مورفولوژیکی همانند ساختارهای سلسله مراتبی با واحد (زیربخش) های معنادار هستند، لذا مدل‌های مبتنی بر کلمات که هر کلمه را یک واحد تجزیه‌ناپذیر در نظر می‌گیرند گزینه‌های مناسبی برای پردازش چنین ساختارهای سلسله مراتبی نمی‌باشند. به هنگام ترجمه از یک زبان با ساختارهای پیچیده مورفولوژیکی (ترکی) به یک زبان ساده‌تر از نظر ساختار (انگلیسی)، یک کلمه پیچیده می‌تواند به چندین کلمه و یا حتی یک جمله کامل در زبان مقصد نگاشت (ترجمه) شود، و این پدیده تایید کننده این ضرورت است که کلمات پیچیده نباید همانند واحدهای تجزیه‌ناپذیر درنظر گرفته شوند. در این پژوهش، به منظور تولید ترجمه‌های با کیفیت بهتر، ما سعی کردیم تا معماری متداول (Encoder-Decoder) در مدل‌های ترجمه ماشینی مبتنی بر شبکه‌های عصبی را تغییر دهیم. در معماری پیشنهادی بجای یک کانال در قسمت انکودر دو کانال به صورت همزمان فعال هستند تا اطلاعات غنی‌تری از سمت مبدا را فراهم کنند؛ در سمت دکودر نیز واحد تمرکز (Attention mechanism) موجود با واحد دیگری با ساختار پیچیده‌تر جابجا شده است. هدف اصلی از انجام این پروژه بهبود کیفیت ترجمه و بدست آوردن بازنمایی بهتری از سمت مبدا بود که نتایج آزمایشات انجام شده مؤ ید این دستاورد هستند. نتایج مطالعات نشان می‌دهد که مدل پیشنهادی قادر است عملکرد بهتری از دیگر مدل‌های موجود داشته باشد و در ترجمه از سه زبان مورد مطالعه ما یعنی آلمانی، روسی، و ترکی به انگلیسی موفق‌تر و با دقت بیشتری عمل می‌کند.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3134–3145
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

In this paper we propose a neural architecture which is designed to deal with morphological complexities on the source side. In such scenarios the neural model and specifically the encoder should be able to handle morphologically complex inputs. It is hard (or even impossible) to benefit from all subunit information within an MCW when it is treated as an atomic unit, e.g. the Turkish word '*ev.de.ki.ler.de.ki*' meaning '*one of those (things/people) at home*' clearly demonstrates that there are meaningful subunits within the word which should be processed separately. Expecting a neural model to learn such intra-word relations by itself complicates the whole process drastically as it requires powerful neural structures which may not lead to the desired result. Accordingly, we believe that there should be an architectural design (manipulation) inspired by the nature of this particular problem, which is able to decompose MCWs or process them in decomposed forms.

Subword-level NMT models are the most preferred alternatives to translate from MRLs, which can be discussed in two main categories. One group of models does not change the neural architecture but manipulates data by decomposing words into their subunits. In this approach either linguistically motivated morphological analyzers such as Morfessor (Smit et al., 2014) or purely statistical models such as the byte-pair encoding (*bpe*) model (Sennrich et al., 2016) are applied to process words. This approach, by its very nature, seems to be a promising solution as it changes the sparse surface form-based vocabulary set into a much smaller set of fundamental subunits. The size of a set including atomic subunits, especially for MRLs, is considerably smaller than that of the vocabulary set. Moreover, this solution partially solves the out-of-vocabulary word problem, as the chance of facing an unknown surface form is much higher than the chance of facing an unknown subunit.

The aforementioned approach could help generate better translations, but there is a potential problem with its single encoder shared among all different types of subunits. In the Morfessor-based segmentation (or any other linguistically motivated segmentation models) the NMT model is fed by stems and affixes instead of words, but all those subunits are processed with a single encoder. Clearly, this is not the best way of learning/representing source-side information, as stems and affixes are different constituents and should be processed separately. The same problem also applies to statistical models such as *bpe*, because although the difference between different subunits might not be obvious (as that of stems and affixes), it does not mean that they are the same types and should be treated equally. Furthermore, there could be another issue raised by the type of the neural architecture. The encoder-decoder architecture (Cho et al., 2014; Sutskever et al., 2014) employed in this approach was originally designed to work at the word level but is instead used for subwords, which might produce some problems, e.g. a sequence of subwords can be considerably longer than a sequence of words. Long sequences are usually hard to remember for neural models and this can affect NMT model's performance. Considering all positive and negative aspects of subword-based models, we still believe that such models are truly beneficial but our main concern is their architecture which we demonstrate can be improved to work with subwords.

As we previously mentioned, there are two main categories of models proposed to tackle the problem of translating from MRLs. We already explained the first group and its advantages and disadvantages. Unlike the first group, the second one focuses more on the neural architecture rather than data preprocessing (word segmentation), and boosts the network with character-based encoders. In such models words are consumed character by character. Usually, there is a convolutional module after the input layer to read characters, connect them to each other, and extract inter-character relations. This approach is helpful as it requires no preprocessing step, but there are two main problems with that. Similar to the previous group, the length of the input sequences could be an issue as by segmenting words into characters an input sequence with a limited number of words is changed to a long sequence of characters. Furthermore, the convolutional computation over characters could be quite costly, e.g. Lee et al. (2017) use 200 convolutional filters of width 1 along with 200 filters of width 2, 250 filters of width 3, and continue up to a filter size of 300 with width 8 to extract useful information on the source side. This amount of computation is carried out only in one layer (for one word), so in addition there are max-pooling

layers followed by 4 highway layers (Srivastava et al., 2015), and then recurrent layers on top. This is a complex architecture which is determined by the type of the input data (a sequence of characters).

Considering both groups of approaches, we assert that there is a need for a third one, for two main reasons: i) models from the first approach work at the subword level but the neural architecture they use is not designed for this purpose; and ii) systems from the second approach use complex architectures to be able to learn character-level information but (it seems) the complex mechanism is not completely effective as results reported from different studies demonstrate that character-based models usually fail to compete with other architectures.[1] Our experimental results also confirm this (see Table 2). The main issue we try to point out by studying these approaches is that existing models may provide state-of-the-art results but they are not designed based on the nature of the problem at hand. Rather, they are just powerful models with recurrent, convolutional, and highway layers which work with simplified (segmented) inputs. They might improve translation quality or partially address the problem, but do not resolve, in a meaningful way, the challenge of handling morphological (subword-level) complexities on the source side.

In this paper we propose a new architecture to address the shortcomings of these previous approaches which is our main contribution. We prefer to work with subwords not characters, because we might lose useful information when we segment structures into characters, or might not be able to extract desired information via the character-based architecture. Related characters are already grouped and connected to each other in subwords, so there is no need to decompose and then glue them back together. Therefore, our basic unit for translation is a morpheme (subword). As mentioned earlier, it seems that existing architectures can be improved further to work with subwords. Accordingly, we propose a novel neural architecture which relies on subwords and is supposed to outperform other character- and subword-based counterparts. In the next section, we first review similar models which tackle the same problem with different approaches, then we explain our own solution. The comparison between our model and others confirms that our hypothesis on designing the neural architecture according to the nature of the problem is true and directly affects performance.

2 Related Work

The problem of morphology and dealing with MCWs is an important issue in the field of sequence modeling as it directly affects the model's architecture and its performance. There is a fair amount of research carried out to address this problem, which can be discussed in two main categories. One group of models tries to cope with the problem on the encoder side where the goal is to understand the rich morphology of the source language. Kim et al. (2016) proposed a convolutional module to process complex inputs for the problem of language modeling. Costa-jussà and Fonollosa (2016) and Lee et al. (2017) adapted the same convolutional encoder to NMT. Luong and Manning (2016) designed a hybrid character- and word-based encoder to try to solve the out-of-vocabulary problem. Vylomova et al. (2016) tackled the problem by comparing the impact of different representation schemes on the encoder. Similarly, Burlot et al. (2017) investigated the impact of different word representation models in the context of factored NMT. Our work is also an example of models which try to provide richer information when the source side is an MRL.

Models reviewed so far address the problem of morphology on the source side. In contrast, there is a group of models which study the same problem for the target side. Huck et al. (2017) compared different word-segmentation models, including linguistically motivated as well as statistical techniques, to find the most appropriate segmentation scheme when translating into MRLs. Chung et al. (2016) tried to design a suitable architecture when the target language is an MRL. They benefit from using a character-based decoder which partially resolves the problem. Passban et al. (2018) proposed a similar approach in which they equipped the character-based decoder with an additional morphology table to inform the decoder with the target language's morphological structures.

[1] Comparing translation results generated by different word-, subword-, and character-based architectures on different language pairs shows that there is no character-based model which is able to outperform its word- and/or subword-based counterparts. Please see results at `http://matrix.statmt.org/?mode=all`

Apart from these models, there are others that do not directly address the problem of morphology but their solutions could be quite useful to translate MRLs. Jean et al. (2015) proposed a model to handle very large vocabularies. Luong et al. (2015) addressed the problem of rare and unseen words with the help of a post-translation phase to exchange unknown tokens with their potential translations. Dalvi et al. (2017) did not propose a new model but studied the impact of morphological information in NMT. They evaluated the behaviour of an encoder-decoder model to see what sort of morphological information is learned via the model and how the model deals with complex structures. Passban (2018) extensively discussed the problem of morphology at the word and sequence level and proposed solutions for modeling and translating sequences in monolingual and bilingual settings.

3 Proposed Approach

We propose an NMT architecture with a double-source encoder and double-attentive decoder for translating from MRLs. Our neural architecture is inspired by models proposed in Firat et al. (2016) and Zoph and Knight (2016). It takes inputs from two different channels: one channel which is referred to as the *main* channel sends stem information (main input), and the other one (the *auxiliary* channel) sends affix information. If the input is $w_0, w_1, ..., w_n$ for the (original) encoder-decoder model, our proposed architecture takes two sequences of $\varsigma_0, \varsigma_1, ..., \varsigma_n$ and $\tau_0, \tau_1, ..., \tau_n$ through the main and auxiliary channels, respectively, where w_i shows the surface form of a word whose stem is ς_i, and affix information associated with w_i is given by τ_i.

Our new neural architecture is based on a hypothesis which assumes the *core semantic unit* is the stem. The translation generated on the target side could appear in different surface forms but it should convey the core meaning dictated by source-side stems. Therefore, to generate a translation the minimum requirement is stem information. Defining the translation process based on stems considerably simplifies the problem, because we no longer need to work with complex surface forms. The sentence representation generated based on stems provides the decoder with high-level source-side information. While it could steer the decoder toward potentially correct target tokens, the decoder needs more than this to generate precise translations. Accordingly, stem information is accompanied with auxiliary information supplied by the affix channel.

There are two main motivations underpinning this type of modelling. We can have different words on the target side (morphologically naive) which could be translations of a single stem. The reason they differ from one another and appear in different forms is because the source-side stem collocates with different affixes. By having different input channels we can model these combinations to help the decoder with the generation of the translation. The double-channel encoder can understand such combinations better, as it separately processes the stem and its affixes. This is the first motivation for our design. Moreover, the neural model is not totally able to extract all information preserved in MCWs when it works with surface forms, but we can simplify the process by explicitly showing subword units to the network, which is a novel point in our architecture. This is the second reason why we process stems and affixes separately.

3.1 The Double-Channel Architecture

Our neural model is an extension to the encoder-decoder model of Cho et al. (2014) with some fundamental changes. In our architecture, given an input sequence $w_1, w_2, ..., w_n$, words are segmented into their stems (ς) and affix tokens (τ). Regarding the structure of the input data, our encoder should process two tokens (stem and affix) instead of one (word) at each time step, so the neural architecture is adapted to the new data structure. To this end, we have two encoders where the stem encoder reads stems one after another and the other one (affix encoder) reads affix tokens. The process can be formulated as in (1):

$$\begin{aligned} s_t &= f(s_{t-1}, \varsigma_t^e) \\ a_t &= f(a_{t-1}, \tau_t^e) \end{aligned} \tag{1}$$

where s_t and a_t are the hidden states of the stem and affix encoders, respectively, at time step t. ς_t^e indicates the embedding of the t-th word's stem, and τ_t^e is the embedding of the affix token for the same

word.

When the final time step is reached for an input sentence of length n, we have two vectors that represent the summary of stem and affix sequences. The decoder should be informed about source-side information to start sampling/generation, for which we place a fully connected layer with a transformation matrix between the encoder and decoder to map both source vectors to the first hidden state of the decoder, as in (2):

$$h_0 = tanh\big(W_{\varsigma\tau}[s_n \bullet a_n]\big) \tag{2}$$

where h_0 is the decoder's hidden state at the beginning, *tanh* applies non-linearity, $W_{\varsigma\tau} \in \mathbb{R}^{(|s_n|+|a_n|)\times|h_0|}$ is the transformation matrix, and $\bullet$ indicates the concatenation operation.

At each time step the decoder samples a token from the target vocabulary. In the simple encoder-decoder model, the decoder conditions the prediction on its hidden state h_t, the last predicted token y_{t-1}, and a dedicated context vector $\mathbf{c}_t$ generated by the attention mechanism (Bahdanau et al., 2014) over the encoder's hidden states. In our case, we have two sets of source-side hidden states and the decoder pays attention to both (instead of one). This process is simply formulated as in (3):

$$y_t = g(h_t, y_{t-1}, \mathbf{c}_t^{\varsigma}, \mathbf{c}_t^{\tau}) \tag{3}$$

Our decoder benefits from a double-attentive mechanism instead of the simple attention model. $\mathbf{c}_t^{\varsigma}$ is the stem-based context vector and provides information about source stems, and $\mathbf{c}_t^{\tau}$ is the affix context vector which informs the decoder about morphological properties of the input sequence.

In order to construct the stem-based context vector $\mathbf{c}_t^{\varsigma}$, we use exactly the same attention model proposed by Bahdanau et al. (2014), as in (4):

$$\begin{aligned} \mathbf{c}_t^{\varsigma} &= \sum_{t'=1}^{n} \alpha_{tt'} s_{t'} \\ \alpha_{tt'} &= \frac{\exp(e_{tt'}^{\varsigma})}{\sum_{k=1}^{n} \exp(e_{tk}^{\varsigma})} \\ e_{tt'}^{\varsigma} &= \mathbf{a}_{\varsigma}(s_{t'}, h_{t-1}) \end{aligned} \tag{4}$$

where $\mathbf{a}_{\varsigma}$ is a simple feed-forward connection. The equation shows that the decoder tries to select relevant stems at each time step which can help it sample better tokens. $\mathbf{c}_t^{\varsigma}$ summarizes all source stems and informs the decoder about the impact of each stem. For the affix context vector, the attention model has a slightly different mechanism whose detail is shown in (5):

$$\begin{aligned} \mathbf{c}_t^{\tau} &= \sum_{j=t'}^{n} \beta_{tt'} a_{t'} \\ \beta_{tt'} &= \frac{\exp(e_{tt'}^{\tau})}{\sum_{k=1}^{n} \exp(e_{tk}^{\tau})} \\ e_{tt'}^{\tau} &= \mathbf{a}_{\tau}(a_{t'}, [h_{t-1} \bullet \mathbf{c}_t^{\varsigma}]) \end{aligned} \tag{5}$$

Affix tokens are weighted given the hidden state of the decoder together with the stem context vector. We know that affixes associate with their stems, so to select the best affix we need to consider both the impact of stems and the decoder's information. Experimental results show that this combination obtains better performance (see Section 4). Figures 1 and 2 visualize our double-attentive attention module. They provide a comparison between simple and double-attentive attention models. It should be noted that for simplicity our figures only show connections associated with the attention module (and exclude others).

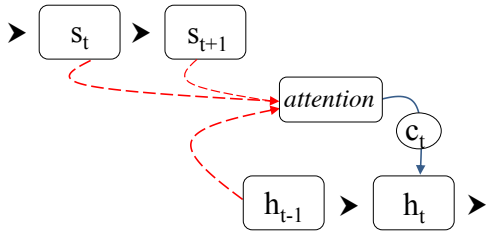

Figure 1: This figure shows the original attention mechanism where the source side-encoder consumes one word at each time step and updates its hidden state s_t. The decoder constructs the context vector $\mathbf{c}_t$ based on the decoder's previous hidden state and all source-side hidden states. The context vector is used along with the (current) hidden state h_t to generate the output.

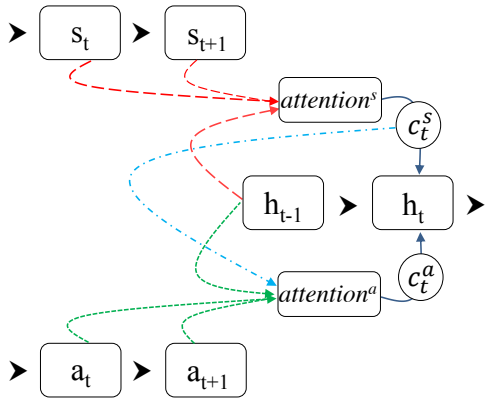

Figure 2: This figure illustrates the double-source encoder, where the stem encoder takes one stem at each time step and updates s_t, and the affix encoder updates a_t after consuming the t-th affix token. The stem-based context vector is depicted by $\boldsymbol{c}_t^s$. The affix-based context vector is constructed using information provided by all of the affix-encoder's hidden states, the decoder's previous hidden state, and the stem context vector.

4 Experimental Study

Our NMT model is an encoder-decoder model with gated recurrent units (GRUs) (Cho et al., 2014). On the encoder side we have two encoders, one for stems and the other for affix tokens. Both encoders include two GRU layers where the first layer is a bidirectional layer and the second layer is a unidirectional layer. Stems and affix tokens are represented with 512-dimensional embeddings. On the decoder side, we have the double-attentive attention module and two GRU layers. The GRU size for both the encoder and decoder is 1024. Similar to input tokens, target tokens are also represented with 512-dimensional embeddings. The network was trained using stochastic gradient descent with Adam (Kingma and Ba, 2015). The mini-batch size is 80, the beam search width is 20, and the norm of the gradient is clipped with the threshold 1.0.

4.1 Data Preparation

Our models are trained to translate from German (De), Russian (Ru), and Turkish (Tr) into English (En). For German and Russian, we used `WMT-15` datasets,[2] where the De–En corpus includes 4.5M parallel sentences and the size of the Ru–En corpus is 2.1M. The `newstest-2013` and `newstest-2015` datasets are used as the development and test sets, respectively. For Tr–En, we used the `OpenSubtitle2016` collection (Lison and Tiedemann, 2016).[3] We randomly selected 3K sentences for each of the development and test sets, and 4M for training.

[2] `http://www.statmt.org/wmt15/translation-task.html`
[3] `http://opus.nlpl.eu/OpenSubtitles2016.php`

	English–German		English–Russian		English–Turkish	
	En	**De**	**En**	**Ru**	**En**	**Tr**
Sentence	4.2M		2.1M		4.0M	
Token	103,692,553	96,235,845	43,944,989	39,694,475	24,875,286	17,915,076
Type	103,574	143,329	70,376	119,258	108,699	178,672
Stem	21,223	26,301	15,964	19,557	15,714	17,419
Affix	13,410	24,054	9,320	17,542	7,112	8,041
Prefix	3,104	5,208	2,959	3,324	1,211	753
Suffix	3,285	4,974	2,219	3,460	1,766	2,701
Char	355	302	360	323	189	231

Table 1: *Sentence* is the number of parallel sentences in the corpus. *Token* is the number of words. *Type* shows the number of unique words. *Stem*, *Prefix*, *Suffix*, and *Affix* show the number of unique stems, prefixes, suffixes, and affix tokens in the corpus. *Char* is the number of unique characters appeared in the corpus.

We segment input words with Morfessor. The longest subunit is selected as the stem. What appears before the stem is the prefix and what follows the stem is considered as the suffix. We do not postprocess Morfessor's output, e.g. Turkish is a suffix-based language with no prefixes, but Morfessor extracted some prefixes for this language. We use segmented units as they are and do not change them. Table 1 summarizes some statistics about our training corpora.

It is clear what the stem encoder takes as its input, but for a word we may have none (null), one or many prefixes and suffixes, whose combination constructs the affix token in our setting. For the affix encoder we have a look-up table which includes prefix and suffix embeddings. The encoder retrieves associated prefix and suffix embeddings at each time step, combines them (via vector summation) and sends the combined vector to the encoder. This architecture is referred to as *pre+suf* in our experiments. If a word has more than one prefix/suffix the encoder retrieves all of them, and if it has none the encoder uses a *null* embedding.

In the *pre+suf* setting, what the affix encoder receives as its input is a vector which has some information about morphological properties of the associated stem (word), but it is possible to provide the encoder with richer information, so we propose the *affix* extensions (*affix*-$\mathbf{c}^\tau$ and *affix*-$\mathbf{c}^\varsigma\mathbf{c}^\tau$).[4] In this new setting we combine the surface form of all prefixes and suffixes to generate a new affix token, e.g if $word_i = prefix_i^1.stem_i.suffix_i^1.suffix_i^2$, the affix token τ_i would be $prefix_i^1.suffix_i^1.suffix_i^2$. We assign a unique embedding for this new combination. Clearly, the embedding of the new affix token is different from the summation of the embeddings of the prefix(es) and suffix(es) and introduces a new token.

A real example can show the difference between the *pre+suf* and *affix* extensions. If the given word is *pre.process.ing*, in the first extension *pre+suf*, the stem encoder takes the embedding of *process* and the affix encoder takes the summation of the embeddings of *pre* and *ing*. In the second extension (either *affix*-$\mathbf{c}^\tau$ or *affix*-$\mathbf{c}^\varsigma\mathbf{c}^\tau$), the stem token is the same but the affix encoder takes an embedding which is exclusively defined for *pre.ing*.

The embedding of the affix token is a new unit which preserves information about: i) morphological properties of its associated word, ii) morphological properties of other words appearing in the sentence, and iii) the collocation and order of different prefixes and suffixes, at the word and sentence level. It seems that the simple combination of prefix and suffix embeddings as in the *pre+suf* extension cannot preserve this amount of information, because the role of each suffix/prefix is clear and their summation will provide us with specific information, while in the second extension we can expect the neural network to learn our desired information. Basically, by defining the affix token we try to introduce an additional cell of memory, in which the NMT model and the affix encoder store the useful information it has learned

[4]In these two extensions everything is the same but the attention mechanism. Equations (5) and (6) explain the attention mechanisms for *affix*-$\mathbf{c}^\varsigma\mathbf{c}^\tau$ and *affix*-$\mathbf{c}^\tau$, respectively.

(in addition to stem).

4.2 Experimental Results

The results obtained from our experiments are reported in Table 2. The numbers in the table are BLEU scores (Papineni et al., 2002) of different neural models. We compare our models to all existing models which translate from MRLs or reported experimental results on our datasets. The first row of the table shows an encoder-decoder model where the decoder works at the character level and uses the architecture proposed in Chung et al. (2016). The first row can be considered as a baseline for all other models reported in the table, as it does not use any complicated neural architecture on the source side. For each word it simply sums stem, prefix, and suffix embeddings together and sends the summed vector as the word-level representation to the GRU-based encoder. Although the model is quite simple, it is able to generate comparable results to other more complex architectures, which reinforces our claim that existing neural architectures are not suitable to work at the subword level. If we find a better way to provide the neural model with subword information, we will be able to improve translation quality still further. For all other models we use the same neural architecture where we keep the character-based decoder unchanged and only change the encoder to compare the capacity of different encoders in modeling morphological information.

Model	Source	Target	De→En	Ru→En	Tr→En
Baseline	ς+*pre*+*suf*	*char*	22.11	22.79	22.98
Costa-jussà and Fonollosa (2016)	*word*	*word*	18.83	-	-
	char	*word*	21.40	-	-
Firat et al. (2016)	*bpe*	*bpe*	24.00	22.44	-
Lee et al. (2017)	*bpe*	*char*	25.27	22.83	-
	char	*char*	25.83*	22.73	-
Sennrich and Haddow (2016)$\star$	$\varsigma \bullet \tau$	*char*	22.41	23.01*	23.14*
Our model					
pre+*suf*	$[\varsigma]^1$ $[$*pre*+*suf*$]^2$	*char*	26.11	22.95	23.62
affix-$\mathbf{c}^{\varsigma}\mathbf{c}^{\tau}$	$[\varsigma]^1$ $[\tau]^2$	*char*	**26.74**	**23.44**	**23.81**
affix-$\mathbf{c}^{\tau}$	$[\varsigma]^1$ $[\tau]^2$	*char*	26.29	23.40	23.74

Table 2: Source and Target indicate the data type for the encoder and decoder, respectively. ς is the stem, pre is the prefix and suf is the suffix. τ is the affix token. The bold-faced score is the best score for the direction and the score with * shows the best performance reported by other existing models. According to paired bootstrap re-sampling (Koehn, 2004) with $p = 0.05$, the bold-faced number is significantly better than the score with *. Brackets show different channels and the + mark indicates the summation, e.g. $[\varsigma]^1$ $[$*pre*+*suf*$]^2$ means the first channel takes a stem at each step and the second channel takes the summation of the prefix and suffix of the associated stem. $\bullet$ is the concatenation operation and $\star$ indicates that results were produced in our experimental setting using our re-implementation of the original model.

The second row shows the model proposed by Costa-jussà and Fonollosa (2016), in which a complicated convolutional module is used to model the relation between characters and build the word-level representation. It shows that the character-level modeling works better than the word-level model but the simple baseline engine still outperforms both settings. The third row belongs to Firat et al. (2016) which is a multi-way multilingual NMT model. The fourth row shows a fully character-level NMT model (Lee et al., 2017) where both the encoder and decoder are designed based on characters. As previously discussed, this model has quite a complicated architecture.

The fifth row reports results from the model of Sennrich and Haddow (2016) that is a suitable model to cope with the problem of complex structures on the source side. The idea behind the model is to enrich input words via additional annotations such as part-of-speech and/or morphological tags. The embedding of the additional annotation(s) is concatenated to the embedding of the word/stem. The idea of attaching

auxiliary information to words/stems was successful and our experimental results also confirm this. In their original model the training data is tagged with external tools which show the need for annotated training sets. We borrowed the same idea and tried to evaluate that architecture in our setting. In the original model, proper morphological tags are defined for input words/stems whereas in our case we consider the affix token as the morphological tag of each stem and we do not need a morphological tagger. The embedding of the affix token is concatenated to the stem embedding and sent to the encoder. Accordingly, the NMT engine is an extension to the baseline model in which the encoder is fed by concatenated structures and the decoder still has the same character-based architecture. This simple concatenation provides better results than the summation applied in the baseline model, which approves affix and subword level information is useful but we only need to find an optimal way to explore it.

The last block shows 3 variations of our model. In the *pre+suf* model the encoder has two encoding channels, one for stems and the other for prefixes and suffixes. At each time step, the stem encoder takes one stem embedding. On the other channel, the affix encoder takes an embedding which is the summation of the prefix and suffix embeddings of the word whose stem is processed by the stem encoder at the same time step. *pre+suf* employs our novel double-attentive attention mechanism. This new architecture enables the NMT engine to perform better than the complicated fully character-based model, which is an indication that the character-level representation does not necessarily provide a better representation. We think that our model outperforms the character-based model because its architecture is more compatible with the nature of MRLs. The impact of the compatible neural architecture becomes more important when we compare the baseline and *pre+suf* models. The input format is exactly the same for both, but the baseline model simply sums stem, prefix, and suffix embeddings, whereas *pre+suf* has two separate channels to process stem and affix information.

The *affix*-$\mathbf{c}^{\varsigma}\mathbf{c}^{\tau}$ variation is the best model among our double-channel models. As previously discussed, assigning a unique embedding to the combination of prefixes and suffixes instead of summing their embeddings generates better results and provides richer information.

The third and last variation, *affix*-$\mathbf{c}^{\tau}$, shows the impact of our architecture on the decoder side. As we modeled in Equation (5), attention weights assigned to the affix-encoder's hidden states are computed based on the decoder's hidden state and the stem context vector. As affix tokens provide complementary information to stem information, the affix context vector should be aware of the content of the stem context vector, so we proposed the model explained in Equation (5). If we only consider the decoder's information to compute affix weights, the equation will be revised as in (6):

$$\begin{aligned} \mathbf{c}_t^{\tau} &= \sum_{j=t'}^{n} \beta_{tt'} a_{t'} \\ \beta_{tt'} &= \frac{\exp(e_{tt'}^{\tau})}{\sum_{k=1}^{n} \exp(e_{tk}^{\tau})} \\ e_{tt'}^{\tau} &= \mathbf{a}_{\tau}(a_{t'}, h_{t-1}) \end{aligned} \tag{6}$$

In the *affix*-$\mathbf{c}^{\varsigma}\mathbf{c}^{\tau}$ version, the energy between the decoder's (t-1)-th state and the affix-encoder's t'-th state was computed by $e_{tt'}^{\tau} = \mathbf{a}_{\tau}(a_{t'}, [h_{t-1} \bullet \mathbf{c}_t^{\varsigma}])$, whereas this version simplifies the computation by estimating $e_{tt'}^{\tau}$ with $\mathbf{a}_{\tau}(a_{t'}, h_{t-1})$. The model described in Equation (6) is implemented in the *affix*-$\mathbf{c}^{\tau}$ extension. Results obtained from this extension show that, although the architecture is also successful compared to other exiting models, its performance is worse than the model which (additionally) involves the stem context vector to compute affix weights.

5 Conclusion and Future Work

In this paper we focused on morphological complexities on the source side in the NMT task and equipped the encoder to handle complex inputs. The proposed model includes two separate channels to consume both stem and affix tokens. This novel two-channel solution can improve performance in two ways, which can be discussed from linguistic and computational perspectives. In terms of linguistic issues,

when the first channel deals with stems instead of words, it is asked to process considerably fewer constituents as the number of stems are much fewer than the number of words. The network is trained to translate simple stem forms instead of complex words. This simplification is usually helpful when translating (from/into) MRLs (Sennrich and Haddow, 2016). Furthermore, there is another channel which preserves morphological information. Every time that the first channel is not able to provide precise translations or needs auxiliary information, the second channel completes it with morphological information.

The proposed architecture can also be studied from mathematical and computational points of view. Defining a secondary channel means adding extra neural modules and parameters (weights). Increasing the number of neural parameters increases the learning capacity of the network and gives a higher possibility to achieve better results. Experimental results confirm that this happens in practice and the extra channel is helpful.

Existing subword-based models segment MCWs using Morfessor, *bpe*, and other models, and feed the translation engine with simplified units. Character-based models also try to facilitate the process by working with characters. Both approaches are useful and improve performance but the improvement is somewhat random, as we do not exactly know what/where the origin of this improvement is. By proposing our double-channel architecture we address this problem and feed the network in a more structured way, in the expectation that we can represent the impact of morphological information. While we are still far from the goal of modeling morphological information for NMT, this research is an important step toward our goal.

Along with these useful properties of our approach, it also has some deficiencies which we plan to address in future work. Our affix information is extracted via Morfessor which could be noisy. Affix tokens are supposed to provide morphological information but we are not sure that our interpretation matches what actually happens in reality. According to experimental results, affix tokens are useful but we are not sure if they preserve real morphological information. In our future work we plan to linguistically enrich our pipeline with more precise and relevant information. As another plan, we also tend to focus on the decoder.

Acknowledgment

We thank our anonymous reviewers for their valuable feedback, as well as the Irish centre for high end computing (www.ichec.ie) for providing computational infrastructures. This work has been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, Banff, Canada.

Franck Burlot, Mercedes García-Martínez, Loïc Barrault, Fethi Bougares, and François Yvon. 2017. Word representations in factored neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 20–31, Copenhagen, Denmark.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany, August.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 142–151.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, June.

Matthias Huck, Simon Riess, and Alexander Fraser. 2017. Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 56–67, Copenhagen, Denmark.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2741–2749, Phoenix, Arizona, USA.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, USA.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, vol. 5:365–378.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 923–929, Portorož, Slovenia.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, August.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318, Pennsylvania, PA., USA.

Peyman Passban, Qun Liu, and Andy Way. 2018. Improving character-based decoding using target-side morphological information for neural machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*, New Orleans, Louisiana, USA, Jun.

Peyman Passban. 2018. *Machine Translation of Morphologically Rich Languages Using Deep Neural Networks*. Ph.D. thesis, School of Computing, Dublin City University, Ireland.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the ICML Deep Learning Workshop*, Lille, France.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Proceedings of the Confrence on Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112. Montreal, Canada.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2016. Word representation models for morphologically rich languages in neural machine translation. *CoRR*, abs/1606.04217.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June.