Improving Character-based Decoding Using Target-Side Morphological Information for Neural Machine Translation

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

Abstract

Recently, neural machine translation (NMT) has emerged as a powerful alternative to conventional statistical approaches. However, its performance drops considerably in the presence of morphologically rich languages (MRLs). Neural engines usually fail to tackle the large vocabulary and high out-of-vocabulary (OOV) word rate of MRLs. Therefore, it is not suitable to exploit existing word-based models to translate this set of languages. In this paper, we propose an extension to the state-of-the-art model of Chung et al. (2016), which works at the character level and boosts the decoder with target-side morphological information. In our architecture, an additional morphology table is plugged into the model. Each time the decoder samples from a target vocabulary, the table sends auxiliary signals from the most relevant affixes in order to enrich the decoder's current state and constrain it to provide better predictions. We evaluated our model to translate English into German, Russian, and Turkish as three MRLs and observed significant improvements.

1 Introduction

Morphologically complex words (MCWs) are multi-layer structures which consist of different subunits, each of which carries semantic information and has a specific syntactic role. Table 1 gives a Turkish example to show this type of complexity. This example is a clear indication that word-based models are not suitable to process such complex languages. Accordingly, when translating MRLs, it might not be a good idea to treat words as atomic units as it demands a large vocabulary that imposes extra overhead. Since MCWs can appear in various forms we require a very large vocabulary to i) cover as many morphological forms and words as we can, and ii) reduce the number of OOVs. Neural models by their nature are complex, and we do not want to make them more complicated by working with large vocabularies. Furthermore, even if we have quite a large vocabulary set, clearly some words would remain uncovered by that. This means that a large vocabulary not only complicates the entire process, but also does not necessarily mitigate the OOV problem. For these reasons we propose an NMT engine which works at the character level.

Word	**Translation**
terbiye	good manners
terbiye.siz	rude
terbiye.siz.lik	rudeness
terbiye.siz.lik.leri	their rudeness
terbiye.siz.lik.leri.nden	from their rudeness

Table 1: Illustrating subword units in MCWs. The boldfaced part indicates the stem.

In this paper, we focus on translating into MRLs and issues associated with word formation on the target side. To provide a better translation we do not necessarily need a large target lexicon, as an MCW can be gradually formed during decoding by means of its subunits, similar to the solution proposed in character-based decoding models (Chung et al., 2016). Generating a complex word character-by-character is a better approach compared to word-level sampling, but it has other disadvantages.

One character can co-occur with another with almost no constraint, but a particular word or morpheme can only collocate with a very limited number of other constituents. Unlike words, characters are not meaning-bearing units and do not preserve syntactic information, so (in the extreme case) the chance of sampling each character by the decoder is almost equal to the others, but this situation is less likely for words. The only constraint that prioritize which character should be sampled is information stored in the decoder, which we believe is insufficient to cope with all ambiguities. Furthermore, when everything is segmented into characters the target sentence with a limited number of words is changed to a very long sequence of characters, which clearly makes it harder for the decoder to remember such a long history. Accordingly, character-based information flows in the decoder may not be as informative as word- or morpheme-based information.

In the character-based NMT model everything is almost the same as its word-based counterpart except the target vocabulary whose size is considerably reduced from thousands of words to just hundreds of characters. If we consider the decoder as a classifier, it should in principle be able to perform much better over hundreds of classes (characters) rather than thousands (words), but the performance of character-based models is almost the same as or slightly better than their word-based versions. This underlines the fact that the character-based decoder is perhaps not fed with sufficient information to provide improved performance compared to word-based models.

Character-level decoding limits the search space by dramatically reducing the size of the target vocabulary, but at the same time widens the search space by working with characters whose sampling seems to be harder than words. The freedom in selection and sampling of characters can mislead the decoder, which prevents us from taking the maximum advantages of character-level decoding. If we can control the selection process with other constraints, we may obtain further benefit from restricting the vocabulary set, which is the main goal followed in this paper.

In order to address the aforementioned problems we redesign the neural decoder in three different scenarios. In the first scenario we equip the decoder with an additional morphology table including target-side affixes. We place an attention module on top of the table which is controlled by the decoder. At each step, as the decoder samples a character, it searches the table to find the most relevant information which can enrich its state. Signals sent from the table can be interpreted as additional constraints. In the second scenario we share the decoder between two output channels. The first one samples the target character and the other one predicts the morphological annotation of the character. This multi-tasking approach forces the decoder to send morphology-aware information to the final layer which results in better predictions. In the third scenario we combine these two models. Section 3 provides more details on our models.

Together with different findings that will be discussed in the next sections, there are two main contributions in this paper. We redesigned and tuned the NMT framework for translating into MRLs. It is quite challenging to show the impact of external knowledge such as morphological information in neural models especially in the presence of large parallel corpora. However, our models are able to incorporate morphological information into decoding and boost its quality. We inject the decoder with morphological properties of the target language. Furthermore, the novel architecture proposed here is not limited to morphological information alone and is flexible enough to provide other types of information for the decoder.

2 NMT for MRLs

There are several models for NMT of MRLs which are designed to deal with morphological complexities. García-Martínez et al. (2016) and Sennrich and Haddow (2016) adapted the factored machine translation approach to neural models. Morphological annotations can be treated as extra factors in such models. Jean et al. (2015) proposed a model to handle very large vocabularies. Luong et al. (2015) addressed the problem of rare words and OOVs with the help of a post-translation phase to exchange unknown tokens with their potential translations. Sennrich et al. (2016) used subword units for NMT. The model relies on frequent subword units instead of words. Costa-jussà and Fonollosa (2016) designed a model for translating from MRLs. The model encodes source words with a convolutional module proposed by Kim et al. (2016). Each word is represented by a convolutional combination of its characters.

Luong and Manning (2016) used a hybrid model for representing words. In their model, unseen and complex words are encoded with a character-based representation, with other words encoded via the usual surface-form embeddings. Vylomova et al. (2016) compared different representation models (word-, morpheme, and character-level models) which try to capture complexities on the source side, for the task of translating from MRLs.

Chung et al. (2016) proposed an architecture which benefits from different segmentation schemes. On the encoder side, words are segmented into subunits with the byte-pair segmentation model (*bpe*) (Sennrich et al., 2016), and on the decoder side, one target character is produced at each time step. Accordingly, the target sequence is treated as a long chain of characters without explicit segmentation. Grönroos et al. (2017) focused on translating from English into Finnish and implicitly incorporated morphological information into NMT through multi-task learning. Passban (2018) comprehensively studied the problem of translating MRLs and addressed potential challenges in the field.

Among all the models reviewed in this section, the network proposed by Chung et al. (2016) could be seen as the best alternative for translating into MRLs as it works at the character level on the decoder side and it was evaluated in different settings on different languages. Consequently, we consider it as a baseline model in our experiments.

3 Proposed Architecture

We propose a compatible neural architecture for translating into MRLs. The model benefits from subword- and character-level information and improves upon the state-of-the-art model of Chung et al. (2016). We manipulated the model to incorporate morphological information and developed three new extensions, which are discussed in Sections 3.1, 3.2, and 3.3.

3.1 The Embedded Morphology Table

In the first extension an additional table containing the morphological information of the target language is plugged into the decoder to assist with word formation. Each time the decoder samples from the target vocabulary, it searches the morphology table to find the most relevant affixes given its current state. Items selected from the table act as guiding signals to help the decoder sample a better character.

Our base model is an encoder-decoder model with attention (Bahdanau et al., 2014), implemented using gated recurrent units (GRUs) (Cho et al., 2014). We use a four-layer model in our experiments. Similar to Chung et al. (2016) and Wu et al. (2016), we use bidirectional units to encode the source sequence. Bidirectional GRUs are placed only at the input layer. The forward GRU reads the input sequence in its original order and the backward GRU reads the input in the reverse order. Each hidden state of the encoder in one time step is a concatenation of the forward and backward states at the same time step. This type of bidirectional processing provides a richer representation of the input sequence.

On the decoder side, one target character is sampled from a target vocabulary at each time step. In the original encoder-decoder model, the probability of predicting the next token y_i is estimated based on i) the current hidden state of the decoder, ii) the last predicted token, and iii) the context vector. This process can be formulated as $p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(h_i, y_{i-1}, \mathbf{c}_i)$, where $g(.)$ is a softmax function, y_i is the target token (to be predicted), $\mathbf{x}$ is the representation of the input sequence, h_i is the decoder's hidden state at the i-th time step, and $\mathbf{c}_i$ indicates the context vector which is a weighted summary of the input sequence generated by the attention module. $\mathbf{c}_i$ is generated via the procedure shown in (1):

$$\begin{aligned} \mathbf{c}_i &= \sum_{j=1}^{n} \alpha_{ij} s_j \\ \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}; \quad e_{ij} = a(s_j, h_{i-1}) \end{aligned} \tag{1}$$

where α_{ij} denotes the weight of the j-th hidden state of the encoder (s_j) when the decoder predicts the i-th target token, and $a()$ shows a combinatorial function which can be modeled through a simple feed-forward connection. n is the length of the input sequence.

In our first extension, the prediction probability is conditioned on one more constraint in addition to those three existing ones, as in $p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(h_i, y_{i-1}, \mathbf{c}_i, \mathbf{c}_i^m)$, where $\mathbf{c}_i^m$ is the morphological context vector and carries information from those useful affixes which can enrich the decoder's information. $\mathbf{c}_i^m$ is generated via an attention module over the morphology table which works in a similar manner to word-based attention model. The attention procedure for

$i=$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
y_i	*b*	*u*		*t*	*e*	*r*	*b*	*i*	*y*	*e*	*s*	*i*	*z*	*l*	*i*	*k*		*i*	*ç*	*i*	*n*
l_i	*stem-c*	*stem-c*	*w-space*	*stem-c*	*stem-c*	*stem-c*	*stem-c*	*stem-c*	*stem-c*	*stem-c*	*siz-c*	*siz-c*	*siz-c*	*lik-c*	*lik-c*	*lik-c*	*w-space*	*stem-c*	*stem-c*	*stem-c*	*stem-c*

Figure 1: The target label that each output channel is supposed to predict when generating the Turkish sequence '*bu*$_1$ *terbiyesizlik*$_2$ *için*$_3$' meaning '*because*$_3$ *of*$_3$ *this*$_1$ *rudeness*$_2$'.

generating $\mathbf{c}_i^m$ is formulated as in (2):

$$\mathbf{c}_i^m = \sum_{u=1}^{|\mathcal{A}|} \beta_{iu} f_u$$
$$\beta_{iu} = \frac{\exp{(e_{iu}^m)}}{\sum_{v=1}^{|\mathcal{A}|} \exp{(e_{iv})}}; \quad e_{iu}^m = a^m(f_u, h_{i-1}) \tag{2}$$

where f_u represents the embedding of the u-th affix (u-th column) in the morphology/affix table $\mathcal{A}$, β_{iu} is the weight assigned to f_u when predicting the i-th target token, and a^m is a feed-forward connection between the morphology table and the decoder.

The attention module in general can be considered as a search mechanism, e.g. in the original encoder-decoder architecture the basic attention module finds the most relevant input words to make the prediction. In multi-modal NMT (Huang et al., 2016; Calixto et al., 2017) an extra attention module is added to the basic one in order to search the image input to find the most relevant image segments. In our case we have a similar additional attention module which searches the morphology table.

In this scenario, the morphology table including the target language's affixes can be considered as an external knowledge repository that sends auxiliary signals which accompany the main input sequence at all time steps. Such a table certainly includes useful information for the decoder. As we are not sure which affix preserves those pieces of useful information, we use an attention module to search for the best match. The attention module over the table works as a filter which excludes irrelevant affixes and amplifies the impact of relevant ones by assigning different weights (β values).

3.2 The Auxiliary Output Channel

In the first scenario, we embedded a morphology table into the decoder in the hope that it can enrich sampling information. Mathematically speaking, such an architecture establishes an extra constraint for sampling and can control the decoder's predictions. However, this is not the only way of constraining the decoder. In the second scenario, we define extra supervision to the network via another predictor (output channel). The first channel is responsible for generating translations and predicts one character at each time step, and the other one tries to understand the morphological status of the decoder by predicting the morphological annotation (l_i) of the target character.

The approach in the second scenario proposes a multi-task learning architecture, by which in one task we learn translations and in the other one morphological annotations. Therefore, all network modules –especially the last hidden layer just before the predictors– should provide information which is useful enough to make correct predictions in both channels, i.e. the decoder should preserve translation as well as morphological knowledge. Since we are translating into MRLs this type of mixed information (morphology+translation) can be quite useful.

In our setting, the morphological annotation l_i predicted via the second channel shows to which part of the word or morpheme the target character belongs, i.e. the label for the character is the morpheme that includes it. We clarify the prediction procedure via an example from our training set (see Section 4). When the Turkish word '*terbiyesizlik*' is generated, the first channel is supposed to predict *t*, *e*, *r*, up to *k*, one after another. For the same word, the second channel is supposed to predict *stem-C* for the fist 7 steps as the first 7 characters '*terbiye*' belong to the stem of the word. The *C* sign indicates that *stem-C* is a class label. The second channel should also predict *siz-C* when the first channel predicts *s* (eighth character), *i* (ninth character), and *z* (tenth character), and *lik-C* when the first channel samples the last three characters. Clearly, the second channel is a classifier which works over the {*stem-C*, *siz-C*, *lik-C*, ...} classes. Figure 1 illustrates a segment of a sentence including this Turkish word and explains which class tags should be predicted by each channel.

To implement the second scenario we require a single-source double-target training corpus: [source sentence] → [sequence of target characters & sequence of morphological annotations] (see Section 4). The objective function should also be manipulated accordingly. Given a training set $\{\mathbf{x}_t, \mathbf{y}_t, \mathbf{m}_t\}_{t=1}^{T}$ the goal is to maximize the joint loss function shown in (3):

$$\lambda \sum_{t=1}^{T} \log P(\mathbf{y}_t|\mathbf{x}_t;\theta) + (1-\lambda) \sum_{t=1}^{T} \log P(\mathbf{m}_t|\mathbf{x}_t;\theta) \tag{3}$$

where $\mathbf{x}_t$ is the t-th input sentence whose translation is a sequence of target characters shown by $\mathbf{y}_t$. $\mathbf{m}_t$ is the sequence of morphological annotations and T is the size of the training set. θ is the set of network parameters and λ is a scalar to balance the contribution of each cost function. λ is adjusted on the development set during training.

3.3 Combining the Extended Output Layer and the Embedded Morphology Table

In the first scenario, we aim to provide the decoder with useful information about morphological properties of the target language, but we are not sure whether signals sent from the table are what we really need. They might be helpful or even harmful, so there should be a mechanism to control their quality. In the second scenario we also have a similar problem as the last layer requires some information to predict the correct morphological class through the second channel, but there is no guarantee to ensure that information in the decoder is sufficient for this sort of prediction. In order to address these problems, in the third extension we combine both scenarios as they are complementary and can potentially help each other.

The morphology table acts as an additional useful source of knowledge as it already consists of affixes, but its content should be adapted according to the decoder and its actual needs. Accordingly, we need a trainer to update the table properly. The extra prediction channel plays this role for us as it forces the network to predict the target language's affixes at the output layer. The error computed in the second channel is back-propagated to the network including the morphology table and updates its affix information into what the decoder actually needs for its prediction. Therefore, the second output channel helps us train better affix embeddings.

The morphology table also helps the second predictor. Without considering the table, the last layer only includes information about the input sequence and previously predicted outputs, which is not directly related to morphological information. The second attention module retrieves useful affixes from the morphology table and concatenates to the last layer, which means the decoder is explicitly fed with morphological information. Therefore, these two modules mutually help each other. The external channel helps update the morphology table with high-quality affixes (backward pass) and the table sends its high-quality signals to the prediction layer (forward pass). The relation between these modules and the NMT architecture is illustrated in Figure 2.

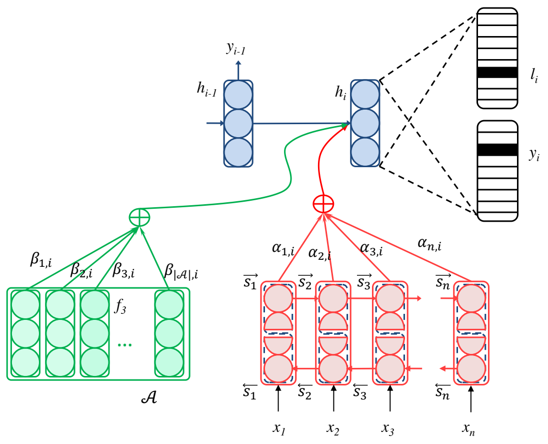

Figure 2: The architecture of the NMT model with an auxiliary prediction channel and an extra morphology table. This network includes only one decoder layer and one encoder layer. ⊕ shows the attention modules.

4 Experimental Study

As previously reviewed, different models try to capture complexities on the encoder side, but to the best of our knowledge the only model which proposes a technique to deal with complex constituents on the decoder side is that of Chung et al. (2016), which should be an appropriate baseline for our comparisons. Moreover, it outperforms other existing NMT models, so we prefer to compare our network to the best existing model. This model is referred to as CDNMT in our experiments. In the next sections first we explain our experimental setting, corpora, and how we build the morphology table (Section 4.1), and then report experimental results (Section 4.2).

4.1 Experimental Setting

In order to make our work comparable we try to follow the same experimental setting used in CDNMT, where the GRU size is 1024, the affix and word embedding size is 512, and the beam width is 20. Our models are trained using stochastic gradient descent with Adam (Kingma and Ba, 2015). Chung et al. (2016) and Sennrich et al. (2016) demonstrated that *bpe* boosts NMT, so similar to CDNMT we also preprocess the source side of our corpora using *bpe*. We use `WMT-15` corpora[1] to train the models, `newstest-2013` for tuning and `newstest-2015` as the test sets. For English–Turkish (En–Tr) we use the `OpenSubtitle2016` collection (Lison and Tiedemann, 2016). The training side of the English–German (En–De), English–Russian (En–Ru), and En–Tr corpora include 4.5, 2.1, and 4 million parallel sentences, respectively. We randomly select 3K sentences for each of the development and test sets for En–Tr. For all language pairs we keep the 400 most frequent characters as the target-side character set and replace the remainder (infrequent characters) with a specific character.

One of the key modules in our architecture is the morphology table. In order to implement it we use a look-up table whose columns include embeddings for the target language's affixes (each column represents one affix) which are updated during training. As previously mentioned, the table is intended to provide useful, morphological information so it should be initialized properly, for which we use a morphology-aware embedding-learning model. To this end, we use the neural language model of Botha and Blunsom (2014) in which each word is represented via a linear combination of the embeddings of its surface form and subunits, e.g. $\overrightarrow{terbiyesizlik} = \overrightarrow{terbiyesizlik} + \overrightarrow{terbiye} + \overrightarrow{siz} + \overrightarrow{lik}$. Given a sequence of words, the neural language model tries to predict the next word, so it learns sentence-level dependencies as well as intra-word relations. The model trains surface form and subword-level embeddings which provides us with high-quality affix embeddings.

Our neural language model is a recurrent network with a single 1000-dimensional GRU layer, which is trained on the target sides of our parallel corpora. The embedding size is 512 and we use a batch size of 100 to train the model. Before training the neural language model, we need to manipulate the training corpus to decompose words into morphemes for which we use Morfessor (Smit et al., 2014), an unsupervised morphological analyzer. Using Morfessor each word is segmented into different subunits where we consider the longest part as the stem of each word; what appears before the stem is taken as a member of the set of prefixes (there might be one or more prefixes) and what follows the stem is considered as a member of the set of suffixes.

Since Morfessor is an unsupervised analyzer, in order to minimize segmentation errors and avoid noisy results we filter its output and exclude subunits which occur fewer than 500 times.[2] After decomposing, filtering, and separating stems from affixes, we extracted several affixes which are reported in Table 2. We emphasize that there might be wrong segmentations in Morfessor's output, e.g. Turkish is a suffix-based language, so there are no prefixes in this language, but based on what Morfessor generated we extracted 11 different types of prefixes. We do not post-process Morfessor's outputs.

Language	Prefix	Suffix
German	75	160
Russian	110	260
Turkish	11	293

Table 2: The number of affixes extracted for each language.

Using the neural language model we train word, stem, and affix embeddings, and initialize the look-up table (but not other parts) of the decoder using those affixes. The look-up table includes high-quality affixes trained on the target side of the parallel corpus by which we train the translation model. Clearly, such an affix table is an additional knowledge source for the decoder. It preserves information which is very close to what the decoder actually needs. However, there might be some missing pieces of information or some incompatibility between the decoder and the table, so we do not freeze the morphology table during training, but let the decoder update it with respect to its needs in the forward and backward passes.

[1] `http://www.statmt.org/wmt15/`

[2] The number may seem a little high, but for a corpus with more than 115M words this is not a strict threshold in practice.

4.2 Experimental Results

Table 3 summarizes our experimental results. We report results for the *bpe→char* setting, which means the source token is a *bpe* unit and the decoder samples a character at each time step. CDNMT is the baseline model. Table 3 includes scores reported from the original CDNMT model (Chung et al., 2016) as well as the scores from our reimplementation. To make our work comparable and show the impact of the new architecture, we tried to replicate CDNMT's results in our experimental setting, we kept everything (parameters, iterations, epochs etc.) unchanged and evaluated the extended model in the same setting. Table 3 reports BLEU scores (Papineni et al., 2002) of our NMT models.

Model	En→De	En→Ru	En→Tr
CDNMT	21.33	26.00	-
CDNMT*	21.01	26.23	18.01
CDNMT$^*_m$	**21.27**	**26.78**	**18.44**
CDNMT$^*_o$	**21.39**	26.39	**18.59**
CDNMT$^*_{mo}$	**21.48**	**26.84**	**18.70**

Table 3: CDNMT* is our implementation of CDNMT. m and o indicates that the base model is extended with the morphology table and the additional output channel, respectively. mo is the combination of both the extensions. The improvement provided by the boldfaced number compared to CDNMT* is statistically significant according to paired bootstrap re-sampling (Koehn, 2004) with $p = 0.05$.

Table 3 can be interpreted from different perspectives but the main findings are summarized as follows:

- The morphology table yields significant improvements for all languages and settings.
- The morphology table boosts the En–Tr engine more than others and we think this is because of the nature of the language. Turkish is an agglutinative language in which morphemes are clearly separable from each other, but in German and Russian morphological transformations rely more on fusional operations rather than agglutination.
- It seems that there is a direct relation between the size of the morphology table and the gain provided for the decoder, because Russian and Turkish have bigger tables and benefit from the table more than German which has fewer affixes.
- The auxiliary output channel is even more useful than the morphology table for all settings but En–Ru, and we think this is because of the morpheme-per-word ratio in Russian. The number of morphemes attached to a Russian word is usually more than those of German and Turkish words in our corpora, and it makes the prediction harder for the classifier (the more the number of suffixes attached to a word, the harder the classification task).
- The combination of the morphology table and the extra output channel provides the best result for all languages.

Figure 3 depicts the impact of the morphology table and the extra output channel for each language.

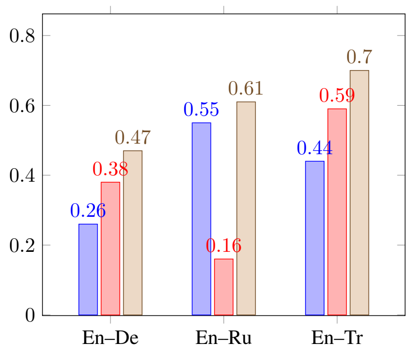

Figure 3: The y axis shows the difference between the BLEU score of CDNMT* and the extended model. The first, second, and third bars show the m, o, and mo extensions, respectively.

To further study our models' behaviour and ensure that our extensions do not generate random improvements we visualized some attention weights when generating '*terbiyesizlik*'. In Figure 4, the upper figure shows attention weights for all Turkish affixes, where the *y* axis shows different time steps and the *x* axis includes attention weights of all affixes (304 columns) for those time steps, e.g. the first row and the first column represents the attention weight assigned to the first Turkish affix when sampling ***t*** in '*terbiyesizlik*'. While at the first glance the figure may appear to be somewhat confusing, but it provides some interesting insights which we elaborate next.

In addition to the whole attention matrix we also visualized a subset of weights to show how the morphology table provides useful information. In the second figure we study the behaviour of the morphology table for the first ($\boldsymbol{t}_1$), fifth ($\boldsymbol{i}_5$), ninth

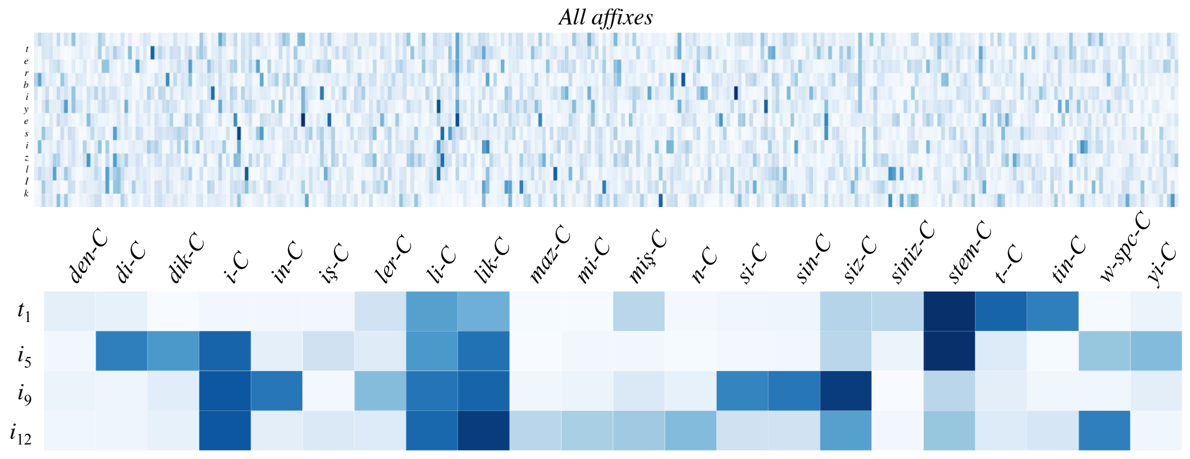

Figure 4: Visualizing the attention weights between the morphology table and the decoder when generating '*terbiyesizlik*.

($\boldsymbol{i}_9$), and twelfth ($\boldsymbol{i}_{12}$) time steps when generating the same Turkish word '$\boldsymbol{t}_1$*er***b***i*$_5$*yes***i**$_9$*zl***i**$_{12}$*k*'. $\boldsymbol{t}_1$ is the first character of the word. We also have three $\boldsymbol{i}$ characters from different morphemes, where the first one is part of the stem, the second one belongs to the suffix '*siz*', and the third one to '*lik*'. It is interesting to see how the table reacts to the same character from different parts. For each time step we selected the top-10 affixes which have the highest attention weights. The set of top-10 affixes can be different for each step, so we made a union of those sets which gives us 22 affixes. The bottom part of Figure 4 shows the attention weights for those 22 affixes at each time step.

After analyzing the weights we observed interesting properties about the morphology table and the auxiliary attention module.[3] The main findings about the behaviour of the table are as follows:

- The model assigns high attention weights to *stem-C* for almost all time steps. However, the weights assigned to this class for $\boldsymbol{t}_1$ and $\boldsymbol{i}_5$ are much higher than those of affix characters (as they are part of the stem). The vertical lines in both figures approve this feature (bad behaviour).

- For some unknown reasons there are some affixes which have no direct relation to that particulate time step but they receive a high attention, such as *maz* in $\boldsymbol{t}_{12}$ (bad behaviour).

- For almost all time steps the highest attention weight belongs to the class which is expected to be selected, e.g. weights for ($\boldsymbol{i}_5$,*stem-C*) or ($\boldsymbol{i}_9$,*siz-C*) (good behaviour).

- The morphology table may send bad or good signals but it is consistent for similar or co-occurring characters, e.g. for the last three time steps $\boldsymbol{l}_{11}$, $\boldsymbol{i}_{12}$, and $\boldsymbol{k}_{13}$, almost the same set of affixes receives the highest attention weights. This consistency is exactly what we are looking for, as it can define a reliable external constraint for the decoder to guide it. Vertical lines on the figure also confirm this fact. They show that for a set of consecutive characters which belong to the same morpheme the attention module sends a signal from a particular affix (good behaviour).

- There are some affixes which might not be directly related to that time step but receive high attention weights. This is because those affixes either include the same character which the decoder tries to predict (e.g. *i-C* for $\boldsymbol{i}_4$ or *t-C* and *tin-C* for $\boldsymbol{t}_1$), or frequently appear with that part of the word which includes the target character (e.g. *mi-C* has a high weight when predicting $\boldsymbol{t}_1$ because $\boldsymbol{t}_1$ belongs to *terbiye* which frequently collocates with *mi-C*: *terbiye+mi*) (good behaviour).

Finally, in order to complete our evaluation study we feed the English-to-German NMT model with the sentence '*Terms and conditions for sending contributions to the BBC*', to show how the model behaves differently and generates a better target sentence. Translations generated by our models are illustrated in Table 4.

[3]Our observations are not based on this example alone as we studied other random examples, and the table shows consistent behaviour for all examples.

Reference:	*Geschäftsbedingungen für das Senden von Beiträgen an die BBC*
CDNMT*	*allgemeinen geschaftsbedingungen fur die versendung von Beiträgen an die BBC*
CDNMT$^*_{mo}$	*Geschäft s bedingungen für die versendung von Beiträgen zum BBC*

Table 4: Comparing translation results for the CDNMT* (baseline) and CDNMT$^*_{mo}$ (improved) models when the input sentence is '*Terms and conditions for sending contributions to the BBC*'.

The table demonstrates that our architecture is able to control the decoder and limit its selections, e.g. the word *'allgemeinen'* generated by the baseline model is redundant. There is no constraint to inform the baseline model that this word should not be generated, whereas our proposed architecture controls the decoder in such situations. After analyzing our model, we realized that there are strong attention weights assigned to the *w-space* (indicating white space characters) and *BOS* (beginning of the sequence) columns of the affix table while sampling the first character of the word *'Geschäft'*, which shows that the decoder is informed about the start point of the sequence. Similar to the baseline model's decoder, our decoder can sample any character including *'a'* of *'allgemeinen'* or *'G'* of *'Geschäft'*. Translation information stored in the baseline decoder is not sufficient for selecting the right character *'G'*, so the decoder wrongly starts with *'i'* and continues along a wrong path up to generating the whole word. However, our decoder's information is accompanied with signals from the affix table which force it to start with a better initial character, whose sampling leads to generating the correct target word.

Another interesting feature about the table is the new structure '*Geschäft s bedingungen'* generated by the improved model. As the reference translation shows, in the correct form these two structures should be glued together via *'s'*, which can be considered as an infix. As our model is supposed to detect this sort of intra-word relation, it treats the whole structure as two compounds which are connected to one another via an infix. Although this is not a correct translation and it would be trivial to post-edit into the correct output form, it is interesting to see how our mechanism forces the decoder to pay attention to intra-word relations.

Apart from these two interesting findings, the number of wrong character selections in the baseline model is considerably reduced in the improved model because of our enhanced architecture.

5 Conclusion and Future Work

In this paper we proposed a new architecture to incorporate morphological information into the NMT pipeline. We extended the state-of-the-art NMT model (Chung et al., 2016) with a morphology table. The table could be considered as an external knowledge source which is helpful as it increases the capacity of the model by increasing the number of network parameters. We tried to benefit from this advantage. Moreover, we managed to fill the table with morphological information to further boost the NMT model when translating into MRLs. Apart from the table we also designed an additional output channel which forces the decoder to predict morphological annotations. The error signals coming from the second channel during training inform the decoder with morphological properties of the target language. Experimental results show that our techniques were useful for NMT of MRLs.

As our future work we follow three main ideas. i) We try to find more efficient ways to supply morphological information for both the encoder and decoder. ii) We plan to benefit from other types of information such as syntactic and semantic annotations to boost the decoder, as the table is not limited to morphological information alone and can preserve other sorts of information. iii) Finally, we target sequence generation for fusional languages. Although our model showed significant improvements for both German and Russian, the proposed model is more suitable for generating sequences in agglutinative languages.

Acknowledgments

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.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *The 3st International Conference on Machine Learning (ICML)*. Beijing, China, pages 1899–1907.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada, pages 1913–1924.

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)*. Doha, Qatar, pages 1724–1734.

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)*. Berlin, Germany, pages 1693–1703.

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)*. Berlin, Germany, pages 357–361.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation. *arXiv preprint arXiv:1609.04621* .

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2017. Extending hybrid word-character neural machine translation with multi-task learning of morphological analysis. In *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark, pages 296–302.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the first Conference on Machine Translation*. pages 639–645.

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)*. Beijing, China, pages 1–10. http://www.aclweb.org/anthology/P15-1001.

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)*. Phoenix, Arizona, USA, pages 2741–2749.

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 *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Barcelona, Spain, pages 388–395.

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*. Portoroz, Slovenia, pages 923–929.

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)*. Berlin, Germany, pages 1054–1063.

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)*. Beijing, China, pages 11–19.

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 of Association for Computational Linguistics*. Pennsylvania, PA., USA, pages 311–318.

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*. Berlin, Germany, pages 83–91.

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)*. Berlin, Germany, pages 1715–1725.

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*. Gothenburg, Sweden, pages 21–24.

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. http://arxiv.org/abs/1606.04217.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.