Character-Aware Decoder for Neural Machine Translation

Adithya Renduchintala* and **Pamela Shapiro*** and **Kevin Duh** and **Philipp Koehn**
Department of Computer Science
Johns Hopkins University
{adi.r,pshapiro,phi}@jhu.edu, kevinduh@cs.jhu.edu

Abstract

Standard neural machine translation (NMT) systems operate primarily on words, ignoring lower-level patterns of morphology. We present a *character-aware* decoder for NMT that can simultaneously work with both word-level and subword-level sequences which is designed to capture such patterns. We achieve character-awareness by augmenting both the softmax and embedding layers of an attention-based encoder-decoder network with convolutional neural networks that operate on spelling of a word (or subword). While character-aware embeddings have been successfully used in the *source-side*, we find that mixing character-aware embeddings with standard embeddings is crucial in the *target-side*. Furthermore, we show that a simple approximate softmax layer can be used for large target-side vocabularies which would otherwise require prohibitively large memory. We experiment on the TED multi-target dataset, translating English into 14 typologically diverse languages. We find that in this low-resource setting, the character-aware decoder provides consistent improvements over word-level and subword-level counterparts with BLEU score gains of up to +3.37.

Introduction

Traditional attention-based encoder-decoder neural machine translation (NMT) models learn *word-level* embeddings, with a continuous representation for each unique word type (Bahdanau, Cho, and Bengio 2015). However, this results in a prohibitively large vocabulary with a long tail of rare words for which we do not learn good representations. Early work in NMT truncated the vocabulary, replacing low-frequency words with an `UNK` token, but this basically punted on translating low-frequency words. More recently, it has become standard practice to mitigate the vocabulary size problem with Byte-Pair Encoding (BPE) (Gage 1994; Sennrich, Haddow, and Birch 2016). BPE iteratively merges consecutive characters into larger chunks based on their frequency, which results in the breaking up of less common words into "subword units."

While BPE addresses the vocabulary size problem, the spellings of the subword units are still completely ignored. On the other hand, purely *character-level* NMT translates one character at a time. Such models can implicitly learn about morphological patterns within words and are able to generate unseen vocabulary. However, there is still debate about how they might suffer from much longer sequences and modeling syntax (Sennrich 2017).

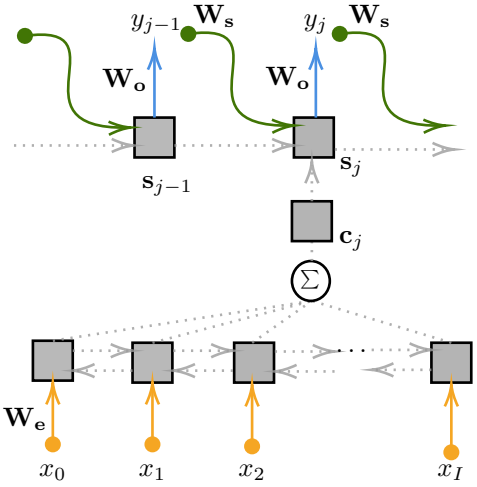

Figure 1: A decoding step generating an output word y_t from the decoder state $\mathbf{s}_t$. x is an input token and $\mathbf{W_e}$ is the source side embedding layer. We highlight the softmax layer $\mathbf{W_o}$ (blue arrows) and the target-side embedding layer $\mathbf{W_s}$ (green arrows). Our goal is to make these embeddings "character-aware."

A middle-ground alternative is character-*aware* word-level modeling. Here, the NMT system operates over words but also pays attention to their spellings and thereby has the ability to learn morphological patterns in the language. Such character-aware approaches have been applied successfully in NMT to the *source-side* word embedding layer (Costa-jussà and Fonollosa 2016), but similar gains have not been achieved on the target side (Belinkov et al. 2017). While this approach was introduced only for word-level modeling, it can be applied on top of BPE as well (Shapiro and Duh 2018). Our goal is to advance character-aware word-level and subword-level modeling on the target side, i.e. in the decoder (Table 2). Our approach is orthogonal to purely character-level methods in that it does not attempt generate novel words, though it can function on top of subword meth-

*equal contribution

No BPE	everybody applauded politely	"word-level"
60k	everybody applauded politely	"subword-level"
30k	everybody appla_ uded politely	
15k	everybody appla_ u_ ded pol_ itely	
3.2k	everybody appla_ u_ ded pol_ ite_ ly	
0	e_ v_ e_ r_ y_ b_ o_ d_ y a_ p_ p_ l_ a_ u_ d_ e_ d p_ o_ l_ i_ t_ e_ l_ y	"char-level"

Table 1: An example sentence under various BPE merge operation thresholds. Even with a relatively small number of merge operations we can see a frequent word like "everybody" completely merged. At 30k-60k merge operations the corpus is largely at "word-level".

	Encoder	**Decoder**
(Bahdanau, Cho, and Bengio 2015)	Word	Word
(Sennrich, Haddow, and Birch 2016)	Subword	Subword
(Lee, Cho, and Hofmann 2017)	Chars	Chars
(Chung, Cho, and Bengio 2016)	Subword	Chars
(Costa-jussà and Fonollosa 2016)	Char-Aware, Word	Word
This Work	**(Sub)Word**	**Char-Aware, (Sub)Word**

Table 2: Landscape of prior work in incorporating character information in NMT. The list of papers in each cell is meant to be a representative sample, not necessarily comprehensive.

ods which do so.

Figure 1 shows the three word embedding layers in a typical encoder-decoder model. They are: (i) *source-side embedding layer*, shown with orange arrows, which has been the focus of prior work by (Costa-jussà and Fonollosa 2016). (ii) *target-side embedding layer*, essentially the target-side counterpart for the source-side embedding layer (green arrows). Here, character-awareness has been attempted and reported ineffective on its own (Belinkov et al. 2017). (iii) *softmax layer* (blue arrows) used with the decoder state to generate a distribution over target vocabulary. While it is not typically considered an embedding, we note that each row in the matrix corresponds to a target vocabulary type and this vector representation affects the output distribution generated. We show that we can make these embeddings character-aware as well. We focus on (ii) and (iii).

Our approach uses a mixture of both compositional embeddings and standard word embeddings via gating functions. Our contributions can be summarized as follows:

1. We propose a method for utilizing character-aware embeddings in an NMT decoder that can be used over word or subword sequences. We evaluate our model on 14 target languages and observe consistent improvements over baselines with an increase of up to $+3.37$ BLEU.
2. We present a simple training-time approximation that allows character-awareness even for large target side vocabularies since naïve application of source-side character-awareness methods to the target side results in prohibitively large memory requirements.
3. We explore how our method interacts with BPE, varying the hyperparameter of merge operations.
4. We analyze to what extent the success of our method corresponds to improved handling of morphology.

Related Work

There has been a long line of work into character-aware word representations for Language Modeling (LM). A summary of word, subword, morphologically informed subword, and character-aware models for LM tasks can be found in (Vania and Lopez 2017). Our work is most aligned with the character-aware models proposed in (Kim et al. 2016). We additionally employ a gating mechanism between character-aware representations and standard word representations similar to language modeling work by (Miyamoto and Cho 2016). However, our gating is a learned type-specific vector rather than a fixed hyperparameter.

NMT has benefited from character-aware word representations on the source side (Costa-jussà and Fonollosa 2016). They follow language modeling work by (Kim et al. 2016) and generate source-side input embeddings using a CNN over the character sequence of each word. Further analysis revealed that hidden states of such character-aware models have increased knowledge of morphology (Belinkov et al. 2017). They additionally try using character-aware representations in the target side embedding layer, leaving the softmax matrix with standard representations, and found no improvements.

There is additionally a line of work on purely character-level NMT, which generates words one character at a time (Ling et al. 2015; Chung, Cho, and Bengio 2016; Passban, Liu, and Way 2018). (Luong and Manning 2016) generate words at the character level only for rare words, which they call a "hybrid" model. Their model is similar in spirit to our work in that they balance both character-based and standard representations. However, their decision to use characters or words is rigidly defined by vocabulary frequency. Meanwhile our system learns gating vectors jointly with the rest of the model. We position our work in terms of the related work in Table 2.

Finally, Byte-Pair Encoding (BPE) has become a standard preprocessing step in NMT pipelines and provides an easy way to generate sequences with a mixture of full words and word fragments. The degree of fragmentation can be controlled using a merge hyperparameter. Furthermore, BPE tends to keep common words together while breaking up rare words. Table 1 illustrates the effect of the merge hyperparameter. As we can see, BPE affords an entire spectrum of sequences from word-level to fully character-level. Our approach can be applied to word-level sequences and sequences at any BPE merge hyperparameter greater than 0 (which is effectively pure character-level). Increasing the hyperparameter results in more words and longer subwords that can exhibit morphological patterns. Our goal is to exploit these morphological patterns and enrich the word (or subword) representations with character-awareness.

Encoder-Decoder Neural Machine Translation

An attention-based encoder-decoder network (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015) models the probability of a target sentence $\mathbf{y}$ of length J given a source sentence $\mathbf{x}$ as:

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{j=1}^{J} (y_j \mid \mathbf{y}_{0:j-1}, \mathbf{x}; \boldsymbol{\theta}) \quad (1)$$

where $\boldsymbol{\theta}$ represents all the parameters of the network. Under this standard model, each unique word is represented by a continuous vector. Then a bidirectional RNN over word vectors is used to encode the source sentence, an attention mechanism weights the hidden states of this encoder, and a unidirectional RNN over the target sentence takes this weighted vector as additional input to predict target words one at a time. Usually GRUs or LSTMs are used for RNNs.

Formally, given the source sequence $\mathbf{x} = \{x_0, x_1, ..., x_i, ...x_I\}$ and a partially generated output sequence $\mathbf{y}_{0:j-1} = \{y_0, y_1, ..., y_{j-1}\}$, the next output token y_j is generated by:

$$p(y_j \mid \mathbf{y}_{0:j-1}, \mathbf{x}) = \text{softmax}(\mathbf{W_o} \mathbf{s}_j) \quad (2)$$

where $\mathbf{s}_j \in \mathbb{R}^{D \times 1}$ is the decoder hidden state at time j and $\mathbf{W_o} \in \mathbb{R}^{|\mathcal{V}| \times D}$ is the weight matrix of the softmax layer, which provides a continuous representation for target words when selecting one to generate. In order to compute the decoder hidden state, the following recurrence is computed at each time-step:

$$\tilde{\mathbf{s}}_j = f(\mathbf{s}_{j-1}, \mathbf{w_s}^{y_{j-1}}, \tilde{\mathbf{s}}_{j-1}) \quad (3)$$

$$\mathbf{s}_j = \tanh(\mathbf{W_c} [\mathbf{c}_j; \tilde{\mathbf{s}}_j]) \quad (4)$$

where f is an LSTM cell.[1] $\mathbf{W_s} \in \mathbb{R}^{|\mathcal{V}| \times E}$ is the embedding matrix, which provides continuous representations for the previous target word when used as input to the RNN. Here, $\mathbf{w_s}^{y_{j-1}} \in \mathbb{R}^{1 \times E}$ is a row vector from the embedding matrix $\mathbf{W_s}$ corresponding to the value of y_{j-1}. $\mathcal{V}$ is the target vocabulary set, D is the is the RNN size and E is embedding size. Often these matrices $\mathbf{W_o}$ and $\mathbf{W_s}$ are tied.

The context vector $\mathbf{c}_j$ is obtained by taking a weighted average over the concatenation of a bidirectional RNN encoder's hidden states.

$$\mathbf{c}_j = \sum_{i=1}^{I} \alpha_i, \mathbf{h}_i \quad (5)$$

$$\alpha_i = \frac{\exp(\mathbf{s}_j^T \mathbf{W_a} \mathbf{h}_i)}{\sum_l \exp(\mathbf{s}_j^T \mathbf{W_a} \mathbf{h}_l)} \quad (6)$$

The attention weight matrix $\mathbf{W_a} \in \mathbb{R}^{D \times H}$ is learned jointly with the model, multiplying with the previous decoder state and bidirectional encoder state $\mathbf{h}_i \in \mathbb{R}^{H \times 1}$, normalized over encoder hidden states via the softmax operation. The encoder hidden states $\mathbf{h}$, are generated using bidirectional recurrent layers over the input sequence:

$$\mathbf{h} = [\overrightarrow{\mathbf{h}} : \overleftarrow{\mathbf{h}}] \quad (7)$$

$$\overrightarrow{\mathbf{h}} = \text{LSTM}_{\text{fwd}}(\text{emb}(\mathbf{x}; \mathbf{W_e})) \quad (8)$$

$$\overleftarrow{\mathbf{h}} = \text{LSTM}_{\text{bwd}}(\text{emb}(\mathbf{x}; \mathbf{W_e})) \quad (9)$$

[1]Note that our notation diverges from (Luong, Pham, and Manning 2015) so that $\mathbf{s}_j$ refers to the state used to make the final predictions.

Character-Aware Extension to the Decoder

In this section we detail the incorporation of character-awareness into the two decoder embedding matrices $\mathbf{W_o}$ and $\mathbf{W_s}$. To begin, we consider an example target side word (or subword in the case of preprocessing with BPE), `cat`. In both $\mathbf{W_o}$ and $\mathbf{W_s}$, there exist row vectors, $\mathbf{w_o}^{\texttt{cat}}$ and $\mathbf{w_s}^{\texttt{cat}}$ that contain the continuous vector representation for the word `cat`. In a traditional NMT system, these vectors are learned as the entire network tries to maximize the objective in Equation 1. The objective does not require the vectors $\mathbf{w_o}^{\texttt{cat}}$ and $\mathbf{w_s}^{\texttt{cat}}$ to model any aspect of the spelling of the word. Figure 2a illustrates a simple non-compositional word embedding.

At a high level, we can view our notion of character-awareness as a "composition function" comp(), parameterized by $\boldsymbol{\omega}$, that takes the character sequence that makes up a word (i.e. its spelling) as input and then produces a continuous vector representation:

$$\mathbf{w}_{\text{comp}}^{\texttt{cat}} = \text{comp}(\langle s \rangle, \texttt{c}, \texttt{a}, \texttt{t}, \langle /s \rangle; \boldsymbol{\omega}) \quad (10)$$

Where $\boldsymbol{\omega}$ is learned jointly with the overall objective. Special characters $\langle s \rangle$ and $\langle /s \rangle$ denote the beginning and end of sequence respectively. We use a composition function that has been successful in language modeling (Kim et al. 2016) and in source-side NMT (Costa-jussà and Fonollosa 2016).

Figure 2b illustrates our compositional approach to generating embeddings. First, a character-embedding layer converts the spelling of a word into a sequence of character embeddings. Next, a series of convolutions is applied over the character sequence and the resulting output matrix is max-pooled along the character sequence dimension. We use 4 convolution filters and set the output channel size for each convolution to $\frac{1}{4}$ the final desired word embedding size. The max-pooled vector from each convolution is concatenated to create the composed word representation. Finally, we add highway layers to obtain the final embeddings.

Composed & Standard Gating

The composition is applied to every type in the vocabulary and thus generates a complete embedding matrix (and softmax matrix). In doing so, we assume that *every* word in the vocabulary has a vector representation that can be composed from its spelling sequence. This is a strong assumption as many words, in particular high frequency words, are not normally compositional, e.g. the substring `ing` in `thing` is not compositional in the way that it is in `running`. Thus, we mix the compositional and standard embedding vectors. We expect standard embeddings to better represent the meaning

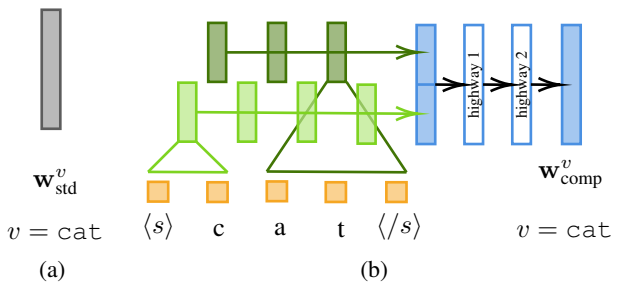

Figure 2: Different approaches to generating embeddings. (a) standard word embedding that treats words as a single symbol. (b) CNN-based composition function. We use multiple CNNs with different kernel sizes over the character embeddings. The resulting hidden states are combined into a single word embedding via max pooling. Note that (b) shows only 2 convolution filters for clarity, in practice we use 4.

of certain words, such has function words and other high-frequency words. For each word v in the vocabulary we also learn a gating vector $\mathbf{g}^v \in [0, 1]^{1 \times D}$.

$$\mathbf{g}^v = \sigma(\mathbf{w}^v_{\text{gate}}) \tag{11}$$

Where, σ is a sigmoid operation and type-specific parameters $\mathbf{w}^v_{\text{gate}}$ are jointly learned along with all the other parameters of the composition function. These parameters are regularized to remain close to $\mathbf{0}$ using dropout. [2] Our final mixed word representation for each word $v \in \mathcal{V}$ by:

$$\mathbf{w}^v_{\text{mix}} = \mathbf{g}^v \odot \mathbf{w}^v_{\text{std}} + (\mathbf{1}. - \mathbf{g}^v) \odot \mathbf{w}^v_{\text{comp}} \tag{12}$$

Where $\mathbf{w}^v_{\text{mix}}$ is the final word embedding, $\mathbf{w}^v_{\text{std}}$ is the standard word embedding, $\mathbf{w}^v_{\text{comp}}$ is the embedding by the composition function and $\mathbf{g}^v$ is the type-specific gating vector for the v'th word. The weight matrix is obtained by stacking the word vectors for each word $v \in \mathcal{V}$. The same representation is used for the target embedding layer and the softmax layer i.e. we set $\mathbf{w_o}^{\texttt{cat}} = \mathbf{w_s}^{\texttt{cat}} = \mathbf{w}^{\texttt{cat}}_{\text{mix}}$, when $v = \texttt{cat}$. Thus, tying the composition function parameters for the softmax weight matrix and the target-side embedding matrix.

Preliminary experiments comparing the standard embedding model and the compositional embedding model with and without gating are summarized in Table 3. Unlike composition in source-side NMT, we find that a gating mechanism is crucial when using compositional embeddings. A purely, composition based softmax layer and target embedding layer result in a catastrophic ≈ 14 BLEU point drop. The gated composition model, however, outperforms the baseline by 0.86 BLEU points suggesting that the CNN composition function is able to extract information that is otherwise lost in the standard embedding scheme. We use the abbreviation CGS (Character-Aware, Gated, Standard) to refer to this model.

Large Vocabulary Softmax Approximation

In Equation 2 of the general NMT framework, the softmax operation generates a distribution over the output vocabulary. Our character-aware model generates a word (or subword) embedding for each entry in the vocabulary by performing convolution operations over its corresponding character embeddings. This entire computational graph has to be in memory in order to perform the backwards pass (i.e. parameter update) during training. Thus, as the vocabulary increases, the character-aware model runs out of memory. To make our character-aware model flexible over a wide range of inputs, including word-level, we incorporate an approximation for the softmax inspired by existing approximations for the standard word embedding based models (Jean et al. 2015). Instead of computing the softmax over the entire vocabulary, a subset of word types $\mathcal{T}$ that occur in a particular training batch and a set $\mathcal{S}$ of word types selected uniformly at random from the entire vocabulary. For a batch size of 80, $\mid \mathcal{T} \mid$ is between 1000 to 2000 and we chose the size of $\mathcal{S}$ as $20K$. For each batch we treat $\tilde{\mathcal{V}} = \mathcal{T} \cup \mathcal{T}$ as the vocabulary.

During decoding, we compute the forward pass $\mathbf{W_o s}_j$ in Equation 2 in several splits of the target vocabulary. As no backward pass is required we clear the memory (i.e. delete the computation graph) after each split is computed.

[2] However, in practice we found that this regularization did not affect performance noticeably in this setting.

Composition Method	BLEU
Std. (no composition)	26.76
C	12.22
CGS (target embedding only)	26.61
CGS (softmax embedding only)	27.16
CGS	**27.62**

Table 3: Preliminary experiments to determine the effectiveness of composition based embeddings and gated embeddings. We used en-de language pair from the TED multi-target dataset. Std. is our baseline with standard word embeddings, model C is the composition only model and CGS is the mixed model which combines the character-aware (composed) embedding and standard embedding via a gating function.

Experiments & Results

We evaluate our character aware model in comparison to a standard baseline on 14 different languages in a lower-resource setting. Additionally, we sweep over several BPE merge hyperparameter settings from character-level to fully word-level for both our model and the baseline and find consistent gains in the character-aware model over the baseline. These gains are stable across all BPE merge hyperparameters all the way up to word-level where they are the highest.

Datasets

We use a collection of TED talk transcripts (Duh 2018; Cettolo, Girardi, and Federico 2012). This dataset has languages with a variety of morphological typologies, which allows us to observe how the success of our character-aware decoder relates to morphological complexity. We keep the source language fixed as English and translate into 14 different languages, since our focus is on the decoder. The training sets for each vary from 73k sentences pairs for Ukrainian to around 170k sentences pairs for Russian, but the validation

L	M	Char-Level	BPE (Subwords) 1.6k	3.2k	7.5k	15k	30k	60k	Word-Level
cs	Std.	11.16	20.28	20.51	20.57	19.60	18.73	17.60	18.44
	CGS	-	20.71	21.04	21.41	21.14	21.28	20.97	**21.49**
uk	Std.	4.77	13.35	15.51	15.79	15.36	14.27	12.50	12.94
	CGS	-	13.80	16.16	15.48	16.28	**16.60**	15.54	15.30
hu	Std.	2.51	15.77	16.33	15.62	16.61	15.45	14.81	14.18
	CGS	-	16.58	16.61	16.88	**17.23**	17.21	17.05	16.52
pl	Std.	10.65	16.14	16.40	16.34	16.76	15.98	15.47	15.49
	CGS	-	16.88	17.12	16.84	17.63	**18.0**	17.32	17.20
he	Std.	22.28	23.07	23.36	23.32	22.76	22.47	21.84	21.26
	CGS	-	23.52	23.38	23.65	23.33	**23.86**	22.78	23.01
tr	Std.	5.29	14.92	14.58	15.11	14.75	13.82	13.69	12.58
	CGS	-	14.42	15.25	15.51	15.54	**15.83**	15.05	14.75
ar	Std.	3.58	15.66	15.67	16.22	15.70	15.05	14.86	14.45
	CGS	-	15.96	15.55	16.17	15.99	**16.28**	15.53	16.05
pt	Std.	35.00	37.47	37.53	37.61	37.85	37.05	37.11	37.13
	CGS	-	37.94	37.98	37.77	38.28	**38.35**	38.11	38.36
ro	Std.	21.58	23.48	24.02	23.72	23.78	22.88	22.73	22.39
	CGS	-	23.55	23.42	23.61	**24.20**	24.00	23.38	23.27
bg	Std.	26.40	31.17	31.41	31.63	31.09	30.92	30.44	30.18
	CGS	-	31.43	31.71	31.81	**32.20**	31.90	31.58	31.43
ru	Std.	14.63	18.17	18.71	19.05	18.80	19.28	18.28	17.60
	CGS	-	18.68	19.26	19.40	19.30	**19.61**	19.23	19.04
de	Std.	23.89	26.98	27.34	27.37	27.23	26.94	27.21	26.84
	CGS	-	26.94	27.55	27.46	27.89	**28.12**	27.37	27.75
fa	Std.	7.44	12.87	12.71	12.86	12.94	12.94	13.20	12.85
	CGS	-	12.35	12.98	13.38	13.36	**13.52**	13.31	12.79
fr	Std.	32.71	35.97	35.75	35.82	35.90	35.31	35.33	35.28
	CGS	-	35.89	35.68	**36.17**	36.10	35.92	36.08	36.01

Table 4: BLEU scores (case insensitive) for a standard embedding encoder-decoder baseline (Std), and character-aware model, composed embedding combined with standard embedding (CGS) for 14 languages and various BPE merge hyperparameters. For purely character-level we only train the standard model as CGS would not have a sequence of characters to compose. For BPE of 60k and word-level we use the softmax approximation described. We see that CGS obtains the best result in all languages.

and test sets are "multi-way parallel", meaning the English sentences (the source side in our experiments) are the same across all 14 languages, and are about 2k sentences each.

NMT Setup

We work with OpenNMT-py (Klein et al. 2017), making modifications to allow for compositional functions over characters in both the target-side embeddings and softmax. We set the RNN hidden state size and embedding size to 1000. We set the character embedding size to 50 and use four CNNs with kernel widths $3, 4, 5$ and 6. The four CNN outputs are concatenated into a compositional embeddings and gated with a standard word embedding. The same composition function (with shared parameters) was used for the target embedding layer and the softmax layer.

For all experiments we optimize the NMT objective (Equation 1) using SGD as it outperformed optimization methods such as Adam and Adadelta[3]. An initial learning rate of 1.0 was used for the first 8 epochs and then decayed with a decay rate of 0.5 until the learning rate reached a minimum threshold of 0.001. We use a batch size of 80 for our main experiments. At the end of each epoch we validate our model on held out validation data and used training accuracy as our model selection criteria for test time. During decoding a beam size of 5 was chosen for all the experiments.

Results

We provide case insensitive BLEU scores for our main experiments, comparing our character-aware model (CGS) which combines a composed embedding and a standard embedding against a baseline model that uses only standard word (and subword) embeddings in Table 4. We also run experiments on purely character-*level* models (i.e. predicting a single character at a time) using the standard embedding scheme. In this low-resource setting we find that character-level does not surpass the subword-level (with standard embeddings) for any of the languages, however for Hebrew (he) and Portuguese (pt) the character-level baseline is competitive. Note that there are several extensions to character-level models with varying degrees of success, in our experiments we use just use the standard encoder-decoder setup with

[3]Others have found similar trends, see (Bahar et al. 2017) and (Maruf and Haffari 2017)

Features	Corpus-dependent			Corpus-independent	
	TT	A	H	UT	UTC
Correlation	0.04	0.59	0.67	0.80	0.49

Table 5: The Pearsons correlation between the features and the relative gain in BLEU score obtained by the character-aware model.

character sequences in the source and target.

Standard settings of the merge hyperparameter are in the range of 15k to 100k. Surprisingly, our experiments show that even smaller BPE merge hyperparameters provide strong baseline BLEU scores when compared against pure character-level and pure word-level. Again, we see a divergence in performance while sweeping over BPE merge hyperparameters across languages – for Czech (cs), Turkish (tr), and Ukrainian (uk) for example, are sensitive to the BPE merge hyperparameter achieving best scores for BPE of 7.5k or 15k. On the other hand, languages like French (fr) and Farsi (fa) the performance is mostly consistent across different BPE merge hyperparameters.

We find that our character-aware model consistently outperforms the baseline for almost all BPE merge hyperparameters across all the languages. Furthermore we see that our model achieves the most gain for languages that exhibit sensitivity to BPE merge hyperparameters. Figure 3 shows the trends for a few selected languages. We display languages representing the range of our model's improvements, from Czech (largest improvement) to German (marginal improvement). We still see a clear and consistent gap between our model (solid lines) and the baseline model (dashed lines). Furthermore, the gap in performance over the baseline increases with larger BPE merge hyperparameters. We hypothesize that this is because at a small number of merge operations, our model does not get to "see" the patterns in the spelling of the words at low BPE and is unable to generalize in the same way that it can with access to larger subword units. While there are a few points at these smaller merges where the character-aware model drops slightly below the baseline (e.g. 3.2k for Arabic (ar)), the overall best score for every language was achieved by the character-aware model, and often at the $30K$ BPE hyperparameter. As the baseline performance drops, our model's performance remains relatively "flat," even improving some for word-level (e.g. in Czech (cs)), despite our need for a softmax approximation for 60k and word-level.

Analysis

We are interested in understanding whether our character-aware model is exploiting morphological patterns in the target language. We investigate this by inspecting the relationship between a set of hand-picked features and improvements obtained by our model over the baseline. These features fall into two categories, *corpus-dependent* and *corpus-independent*. We extracted corpus-dependent features which are known to correlate with human judgments of morphological complexity (Bentz et al. 2016), from each language's training data. We used the following features:

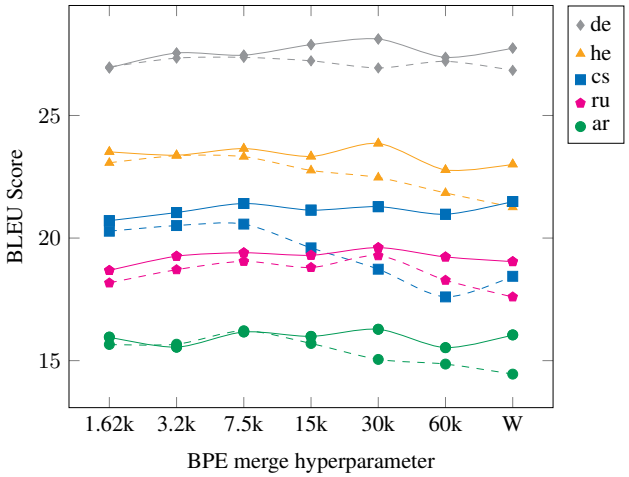

Figure 3: Plot of BLEU scores against BPE merge hyperparameters for selected languages. The solid line plots the performance of the character-aware (CGS) model, and the dashed line plots the baseline. Here we clearly observe that the CGS model is almost always above the baseline. The CGS model is also relatively "flat" when compared with the baseline which shows a consistent deterioration as the BPE merge hyperparameter is increased.

(i) Type-Token Ratio (TT): the ratio of the number of word types to the total number of word tokens in the target side.

(ii) Word-Alignment Score (A): computed as $A = \frac{|\text{many-to-one}| - |\text{one-to-many}|}{|\text{all-alignments}|}$. One-to-one, one-to-many and many-to-one alignment types are illustrated in Figure 4.[4] The logic behind this feature is that a morphologically poor source language (like English) paired with a richer target language should exhibit more many-to-one alignments—a single word in the target will contain more information (via morphological phenomena) that can only be translated using multiple words in the source.

(iii) Word-Level Entropy (H): computed as $H = \sum_{v \in \mathcal{V}} p(v) \log p(v)$ where v is a word type. This metric reflects the average information content of the words in a corpus. Languages with more dependence on having a large number of word types rather than word order or phrase structure will score higher.

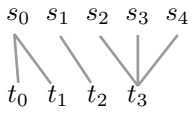

Figure 4: Example of one-to-many (s_0 to t_0, t_1), one-to-one (s_1 to t_2) and many-to-one (s_2, s_3, s_4 to t_3) alignments. For this example $A = (3-2)/6$.

For the corpus-independent features we used a morphological annotation corpus called UniMorph (Sylak-Glassman et al. 2015). The UniMorph corpus contains

[4]We use FastAlign (Dyer, Chahuneau, and Smith 2013) for word alignments with the grow-diag-final-and heuristic from (Och and Ney 2003) for symmetrization.

a large list of inflected words (in several languages) along with the word's lemma and a set of morphological tags. For example, the French UniMorph corpus contains the word `marchai` (walked), which is associated with its lemma, `marcher` and a set of morphological tags {**V,IND,PST,1,SG,PFV**}. There are 19 such tags in the French UniMorph corpus. A morphologically richer language like Hungarian, for example, has 36 distinct tags. We used the number of distinct tags (UT) and the number of different tag combinations (UTC) that appear in the UniMorph corpus for each language. Note that we do not filter out words (and its associated tags) from the UniMorph corpus that are absent in our parallel data. This ensures that the UT and UTC features are completely corpus independent.

The Pearson's correlation between these hand-picked features and relative gain observed by our model is shown in Table 5. For this analysis we used the relative gain obtained from the word-level experiments. Concretely, the relative gain for Czech was computed as $\frac{21.49-18.44}{18.44}$ We see a strong correlation between the corpus-independent feature (UT) and our model's gain. Alignment score and Word Entropy are also moderately correlated. Surprisingly, we see no correlation to type-token ratio.

We further analyzed how these features jointly relate to the relative gains, and investigate to what extent the features explain the gains on a *language-specific* bases. We trained a linear regression model setting the relative gains as the predicted variable y and the feature values as the input variables $\mathbf{x}$ (see Equation 14), with the goal of studying the linear regression weights $\boldsymbol{\phi}$ and gain insights into the relative contribution of each feature.[5] We used feature-augmented domain adaptation where we consider each language as a domain (Daumé III 2007). This method allows us to learn a set of "general" weights as well language-specific weights. The general feature weights can be interpreted as being indicative of the overall trends in the dataset across all the languages, while the language-specific weights indicate how much a particular language deviates from the overall trend.

$$\mathcal{L}(\boldsymbol{\phi}) = \sum_{i \in \mathcal{I}} \mid y_i - \tilde{y}_i \mid^2 - \lambda \mid \boldsymbol{\phi} \mid^2 \tag{13}$$

$$\tilde{y}_i = \boldsymbol{\phi}_{\text{ALL}}^T \mathbf{x}_i + \boldsymbol{\phi}_i^T \mathbf{x}_i \tag{14}$$

Where, y is the true relative gain in BLEU, $\tilde{y}$ is the predicted gain, $\mathbf{x}$ is a vector of input feature values, $\boldsymbol{\phi}_{\text{ALL}}$ and $\boldsymbol{\phi}_i$ are the general and language-specific weights, and i indexes into the set of languages in our analysis. We set λ to 0.05.

The matrix of learned weights $\boldsymbol{\phi}$ is visualized in Figure 5. The first row of weights correspond to the "general" weights that are used for all the languages, followed by language-specific weights sorted by relative gain. We are encouraged to see that the UT feature is again highly weighted overall, followed by the Alignment score and Word-entropy.

While the general weights align with the correlation results this analysis also shows that the UTC weight for Czech and Turkish are much larger than any of the other languages' and indeed we can verify that these languages have 194 and 300 different tag combinations while the average tag combinations is ≈ 110.

[5]The input features were min-max normalized for the regression analysis.

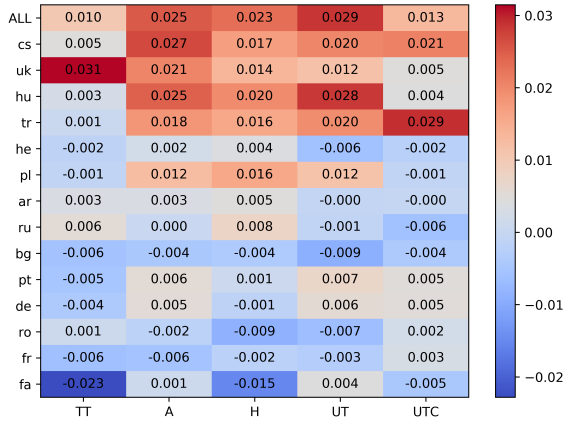

Figure 5: Feature weights of the feature-augmented language adapted linear regression model. The first row represents the "general" set of weights used for all of the languages. Each row below are the language-adapted weights that only "fire" for that specific language.

From the corpus-dependent features, word alignment score strongly predicts the gain in BLEU scores. For Czech, Ukrainian, Turkish, Hungarian, and Polish we see additional weight placed on this feature. A similar trend can be seen for the word-entropy feature. While type-token ratio does not exhibit a strong overall trend, we see that Ukrainian and Farsi are "outliers" in this feature.

Our correlation and regression analysis strongly suggest that our character-aware modeling helps the most when the target language has inherent morphological complexity and that our method does indeed have the ability to handle morphological patterns present in the target languages.

Conclusion

We extend character-aware word-level modeling to work for the decoder in NMT. To achieve improvements in the decoder, we augment the softmax layer in addition to the target embeddings with character-awareness. We also find it necessary to add a gating function to balance compositional embeddings with standard embeddings. We evaluate our method on a low-resource dataset translating from English into 14 languages, and on top of a spectrum of BPE merge operations. Furthermore, for word-level and higher merge hyperparameter settings, we introduced an approximation to the softmax layer. We achieve consistent performance gains across languages and subword granularities, and perform an analysis indicating that the gains for each language correspond to morphological complexity.

For future work, we would like to explore how our methods might be of use in higher-resource settings. This would include making our model more efficient, and exploring additional approximations for high-resource settings.

References

[Bahar et al. 2017] Bahar, P.; Alkhouli, T.; Peter, J.-T.; Brix, C. J.-S.; and Ney, H. 2017. Empirical investigation of optimization algorithms in neural machine translation. *The Prague Bulletin of Mathematical Linguistics* 108(1):13–25.

[Bahdanau, Cho, and Bengio 2015] Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

[Belinkov et al. 2017] Belinkov, Y.; Durrani, N.; Dalvi, F.; Sajjad, H.; and Glass, J. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 861–872. Association for Computational Linguistics.

[Bentz et al. 2016] Bentz, C.; Ruzsics, T.; Koplenig, A.; and Samardzic, T. 2016. A comparison between morphological complexity measures: typological data vs. language corpora. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, 142–153.

[Cettolo, Girardi, and Federico 2012] Cettolo, M.; Girardi, C.; and Federico, M. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, 261–268.

[Chung, Cho, and Bengio 2016] Chung, J.; Cho, K.; and Bengio, Y. 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)*, 1693–1703.

[Costa-jussà and Fonollosa 2016] Costa-jussà, M. R., and Fonollosa, J. A. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 357–361.

[Daumé III 2007] Daumé III, H. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 256–263.

[Duh 2018] Duh, K. 2018. The multitarget ted talks task. `http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/`.

[Dyer, Chahuneau, and Smith 2013] Dyer, C.; Chahuneau, V.; and Smith, N. A. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 644–648.

[Gage 1994] Gage, P. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38.

[Jean et al. 2015] Jean, S.; Cho, K.; Memisevic, R.; and Bengio, Y. 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)*, 1–10.

[Kim et al. 2016] Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*.

[Klein et al. 2017] Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, System Demonstrations*, 67–72.

[Lee, Cho, and Hofmann 2017] Lee, J.; Cho, K.; and Hofmann, T. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association of Computational Linguistics* 5(1):365–378.

[Ling et al. 2015] Ling, W.; Trancoso, I.; Dyer, C.; and Black, A. W. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

[Luong and Manning 2016] Luong, M.-T., and Manning, C. D. 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 (Long Papers)*, volume 1, 1054–1063.

[Luong, Pham, and Manning 2015] Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 1412–1421.

[Maruf and Haffari 2017] Maruf, S., and Haffari, G. 2017. Document context neural machine translation with memory networks. *arXiv preprint arXiv:1711.03688*.

[Miyamoto and Cho 2016] Miyamoto, Y., and Cho, K. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1992–1997.

[Och and Ney 2003] Och, F. J., and Ney, H. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1):19–51.

[Passban, Liu, and Way 2018] Passban, P.; Liu, Q.; and Way, A. 2018. Improving character-based decoding using target-side morphological information for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, 58–68.

[Sennrich, Haddow, and Birch 2016] Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, Volume 1: Long Papers*.

[Sennrich 2017] Sennrich, R. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, 376–382.

[Shapiro and Duh 2018] Shapiro, P., and Duh, K. 2018. Bpe

and charcnns for translation of morphology: A cross-lingual comparison and analysis. *arXiv preprint arXiv:1809.01301*.

[Sylak-Glassman et al. 2015] Sylak-Glassman, J.; Kirov, C.; Post, M.; Que, R.; and Yarowsky, D. 2015. A universal feature schema for rich morphological annotation and fine-grained cross-lingual part-of-speech tagging. In *International Workshop on Systems and Frameworks for Computational Morphology*, 72–93. Springer.

[Vania and Lopez 2017] Vania, C., and Lopez, A. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 2016–2027.