Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models

Minh-Thang Luong and **Christopher D. Manning**

Computer Science Department, Stanford University, Stanford, CA 94305

{lmthang,manning}@stanford.edu

Abstract

Nearly all previous work on neural machine translation (NMT) has used quite restricted vocabularies, perhaps with a subsequent method to patch in unknown words. This paper presents a novel word-character solution to achieving open vocabulary NMT. We build hybrid systems that translate mostly at the *word* level and consult the *character* components for rare words. Our character-level recurrent neural networks compute source word representations and recover unknown target words when needed. The twofold advantage of such a hybrid approach is that it is much faster and easier to train than character-based ones; at the same time, it never produces unknown words as in the case of word-based models. On the WMT'15 English to Czech translation task, this hybrid approach offers an addition boost of $+2.1-11.4$ BLEU points over models that already handle unknown words. Our best system achieves a new state-of-the-art result with 20.7 BLEU score. We demonstrate that our character models can successfully learn to not only generate well-formed words for Czech, a highly-inflected language with a very complex vocabulary, but also build correct representations for English source words.

1 Introduction

Neural Machine Translation (NMT) is a simple new architecture for getting machines to translate. At its core, NMT is a single deep neural network that is trained end-to-end with several advantages such as simplicity and generalization. Despite being relatively new, NMT has already achieved
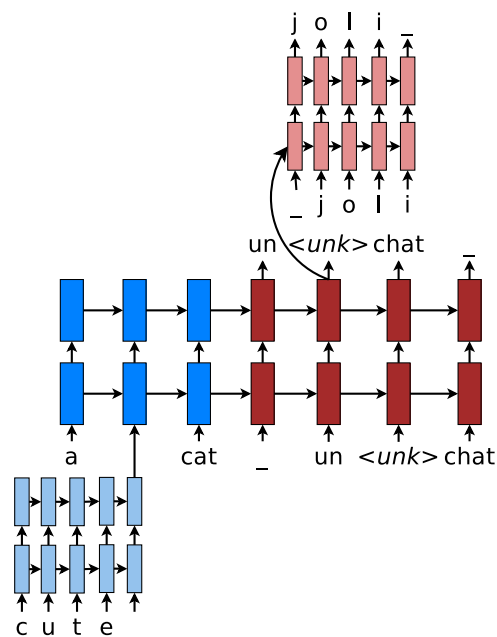

Figure 1: **Hybrid NMT** – example of a word-character model for translating "a cute cat" into "un joli chat". Hybrid NMT translates at the word level. For rare tokens, the character-level components build source representations and recover target *<unk>*. "_" marks sequence boundaries.

state-of-the-art translation results for several language pairs such as English-French (Luong et al., 2015b), English-German (Jean et al., 2015a; Luong et al., 2015a; Luong and Manning, 2015), and English-Czech (Jean et al., 2015b).

While NMT offers many advantages over traditional phrase-based approaches, such as small memory footprint and simple decoder implementation, nearly all previous work in NMT has used quite restricted vocabularies, crudely treating all other words the same with an *<unk>* symbol. Sometimes, a post-processing step that patches in unknown words is introduced to alleviate this problem. Luong et al. (2015b) propose to annotate

occurrences of target <unk> with positional information to track their alignments, after which simple word dictionary lookup or identity copy can be performed to replace <unk> in the translation. Jean et al. (2015a) approach the problem similarly but obtain the alignments for unknown words from the attention mechanism. We refer to these as the *unk replacement* technique.

Though simple, these approaches ignore several important properties of languages. First, *monolingually*, words are morphologically related; however, they are currently treated as independent entities. This is problematic as pointed out by Luong et al. (2013): neural networks can learn good representations for frequent words such as "distinct", but fail for rare-but-related words like "distinctiveness". Second, *crosslingually*, languages have different alphabets, so one cannot naïvely memorize all possible surface word translations such as name transliteration between "Christopher" (English) and "Kryštof" (Czech). See more on this problem in (Sennrich et al., 2016).

To overcome these shortcomings, we propose a novel *hybrid* architecture for NMT that translates mostly at the word level and consults the character components for rare words when necessary. As illustrated in Figure 1, our hybrid model consists of a word-based NMT that performs most of the translation job, except for the two (hypothetically) rare words, "cute" and "joli", that are handled separately. On the *source* side, representations for rare words, "cute", are computed on-the-fly using a deep recurrent neural network that operates at the character level. On the *target* side, we have a separate model that recovers the surface forms, "joli", of <unk> tokens character-by-character. These components are learned jointly end-to-end, removing the need for a separate unk replacement step as in current NMT practice.

Our hybrid NMT offers a twofold advantage: it is much faster and easier to train than character-based models; at the same time, it never produces unknown words as in the case of word-based ones. We demonstrate at scale that on the WMT'15 English to Czech translation task, such a hybrid approach provides an additional boost of $+2.1-11.4$ BLEU points over models that already handle unknown words. We achieve a new state-of-the-art result with 20.7 BLEU score. Our analysis demonstrates that our character models can successfully learn to not only generate well-formed words for Czech, a highly-inflected language with a very complex vocabulary, but also build correct representations for English source words.

We provide code, data, and models at `http://nlp.stanford.edu/projects/nmt`.

2 Related Work

There has been a recent line of work on end-to-end character-based neural models which achieve good results for part-of-speech tagging (dos Santos and Zadrozny, 2014; Ling et al., 2015a), dependency parsing (Ballesteros et al., 2015), text classification (Zhang et al., 2015), speech recognition (Chan et al., 2016; Bahdanau et al., 2016), and language modeling (Kim et al., 2016; Jozefowicz et al., 2016). However, success has not been shown for cross-lingual tasks such as machine translation.[1] Sennrich et al. (2016) propose to segment words into smaller units and translate just like at the word level, which does not learn to understand relationships among words.

Our work takes inspiration from (Luong et al., 2013) and (Li et al., 2015). Similar to the former, we build representations for rare words on-the-fly from subword units. However, we utilize recurrent neural networks with characters as the basic units; whereas Luong et al. (2013) use recursive neural networks with morphemes as units, which requires existence of a morphological analyzer. In comparison with (Li et al., 2015), our hybrid architecture is also a hierarchical sequence-to-sequence model, but operates at a different granularity level, word-character. In contrast, Li et al. (2015) build hierarchical models at the sentence-word level for paragraphs and documents.

3 Background & Our Models

Neural machine translation aims to directly model the conditional probability $p(y|x)$ of translating a source sentence, $x_1, \ldots, x_n$, to a target sentence, $y_1, \ldots, y_m$. It accomplishes this goal through an *encoder-decoder* framework (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). The *encoder* computes a representation s for each source sentence. Based on that source

---

[1]Recently, Ling et al. (2015b) attempt character-level NMT; however, the experimental evidence is weak. The authors demonstrate only small improvements over word-level baselines and acknowledge that there are no differences of significance. Furthermore, only small datasets were used without comparable results from past NMT work.

representation, the *decoder* generates a translation, one target word at a time, and hence, decomposes the log conditional probability as:

$$\log p(y|x) = \sum_{t=1}^{m} \log p\left(y_t | y_{<t}, \boldsymbol{s}\right) \quad (1)$$

A natural model for sequential data is the recurrent neural network (RNN), used by most of the recent NMT work. Papers, however, differ in terms of: (a) architecture – from unidirectional, to bidirectional, and deep multi-layer RNNs; and (b) RNN type – which are long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (Cho et al., 2014). All our models utilize the *deep multi-layer* architecture with *LSTM* as the recurrent unit; detailed formulations are in (Zaremba et al., 2014).

Considering the top recurrent layer in a deep LSTM, with $\boldsymbol{h}_t$ being the current target hidden state as in Figure 2, one can compute the probability of decoding each target word y_t as:

$$p\left(y_t | y_{<t}, \boldsymbol{s}\right) = \text{softmax}\left(\boldsymbol{h}_t\right) \quad (2)$$

For a parallel corpus $\mathbb{D}$, we train our model by minimizing the below cross-entropy loss:

$$J = \sum_{(x,y) \in \mathbb{D}} - \log p(y|x) \quad (3)$$

Attention Mechanism – The early NMT approaches (Sutskever et al., 2014; Cho et al., 2014), which we have described above, use only the last encoder state to initialize the decoder, i.e., setting the input representation $\boldsymbol{s}$ in Eq. (1) to $[\bar{\boldsymbol{h}}_n]$. Recently, Bahdanau et al. (2015) propose an *attention mechanism*, a form of random access memory for NMT to cope with long input sequences. Luong et al. (2015a) further extend the attention mechanism to different scoring functions, used to compare source and target hidden states, as well as different strategies to place the attention. In all our models, we utilize the *global* attention mechanism and the *bilinear form* for the attention scoring function similar to (Luong et al., 2015a).

Specifically, we set $\boldsymbol{s}$ in Eq. (1) to the set of source hidden states at the top layer, $[\bar{\boldsymbol{h}}_1, \ldots, \bar{\boldsymbol{h}}_n]$. As illustrated in Figure 2, the attention mechanism consists of two stages: (a) *context vector* – the current hidden state $\boldsymbol{h}_t$ is compared with individual source hidden states in $\boldsymbol{s}$ to learn an alignment vector, which is then used to compute the context vector $\boldsymbol{c}_t$ as a weighted average of $\boldsymbol{s}$; and (b) *attentional hidden state* – the context vector $\boldsymbol{c}_t$ is then
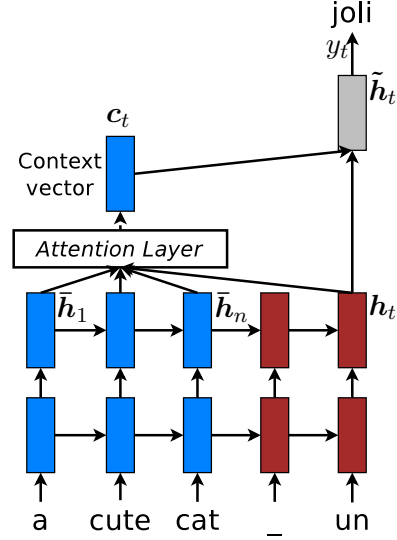
Figure 2: **Attention mechanism**.

used to derive a new attentional hidden state:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad (4)$$

The attentional vector $\tilde{\boldsymbol{h}}_t$ then replaces $\boldsymbol{h}_t$ in Eq. (2) in predicting the next word.

4 Hybrid Neural Machine Translation

Our hybrid architecture, illustrated in Figure 1, leverages the power of both words and characters to achieve the goal of open vocabulary NMT. The core of the design is a *word*-level NMT with the advantage of being fast and easy to train. The *character* components empower the word-level system with the abilities to compute any source word representation on the fly from characters and to recover character-by-character unknown target words originally produced as *<unk>*.

4.1 Word-based Translation as a Backbone

The core of our hybrid NMT is a deep LSTM encoder-decoder that translates at the *word* level as described in Section 3. We maintain a vocabulary of $|V|$ frequent words for each language. Other words not inside these lists are represented by a universal symbol *<unk>*, one per language. We translate just like a word-based NMT system with respect to these source and target vocabularies, except for cases that involve *<unk>* in the source input or the target output. These correspond to the character-level components illustrated in Figure 1.

A nice property of our hybrid approach is that by varying the vocabulary size, one can control

how much to blend the word- and character-based models; hence, taking the best of both worlds.

4.2 Source Character-based Representation

In regular word-based NMT, for all rare words outside the source vocabulary, one feeds the universal embedding representing $<unk>$ as input to the encoder. This is problematic because it discards valuable information about the source word. To fix that, we learn a deep LSTM model over characters of source words. For example, in Figure 1, we run our deep character-based LSTM over 'c', 'u', 't', 'e', and '_' (the boundary symbol). The final hidden state at the top layer will be used as the on-the-fly representation for the current rare word. The layers of the deep character-based LSTM are always initialized with *zero* states. One might propose to connect hidden states of the word-based LSTM to the character-based model; however, we chose this design for various reasons. First, it simplifies the architecture. Second, it allows for efficiency through *precomputation*: before each mini-batch, we can compute representations for rare source words all at once. All instances of the same word share the same embedding, so the computation is per *type*.[2]

4.3 Target Character-level Generation

General word-based NMT allows generation of $<unk>$ in the target output. Afterwards, there is usually a post-processing step that handles these unknown tokens by utilizing the alignment information derived from the attention mechanism and then performing simple word dictionary lookup or identity copy (Luong et al., 2015a; Jean et al., 2015a). While this approach works, it suffers from various problems such as alphabet mismatches between the source and target vocabularies and multi-word alignments. Our goal is to address all these issues and create a coherent framework that handles an unlimited output vocabulary.

Our solution is to have a separate deep LSTM that "translates" at the character level given the current word-level state. We train our system such that whenever the word-level NMT produces an $<unk>$, we can consult this character-level decoder to recover the correct surface form of the unknown target word. This is illustrated in Figure 1.

---

[2]While Ling et al. (2015b) found that it is slow and difficult to train source character-level models and had to resort to pretraining, we demonstrate later that we can train our deep character-level LSTM perfectly fine in an end-to-end fashion.

The training objective in Eq. (3) now becomes:

$$J = J_w + \alpha J_c \qquad (5)$$

Here, J_w refers to the usual loss of the word-level NMT; in our example, it is the sum of the negative log likelihood of generating {"un", "$<unk>$", "chat", "_"}. The remaining component J_c corresponds to the loss incurred by the character-level decoder when predicting characters, e.g., {'j', 'o', 'l', 'i', '_'}, of those rare words not in the target vocabulary.

Hidden-state Initialization Unlike the source character-based representations, which are context-independent, the target character-level generation requires the current word-level context to produce meaningful translation. This brings up an important question about what can best represent the current context so as to initialize the character-level decoder. We answer this question in the context of the attention mechanism (§3).

The final vector $\tilde{h}_t$, just before the softmax as shown in Figure 2, seems to be a good candidate to initialize the character-level decoder. The reason is that $\tilde{h}_t$ combines information from both the context vector c_t and the top-level recurrent state h_t. We refer to it later in our experiments as the *same-path* target generation approach.

On the other hand, the same-path approach worries us because all vectors $\tilde{h}_t$ used to seed the character-level decoder might have similar values, leading to the same character sequence being produced. The reason is because $\tilde{h}_t$ is directly used in the softmax, Eq. (2), to predict the same $<unk>$. That might pose some challenges for the model to learn useful representations that can be used to accomplish two tasks at the same time, that is to predict $<unk>$ and to generate character sequences. To address that concern, we propose another approach called the *separate-path* target generation.

Our separate-path target generation approach works as follows. We mimic the process described in Eq. (4) to create a counterpart vector $\breve{h}_t$ that will be used to seed the character-level decoder:

$$\breve{h}_t = \tanh(\breve{W}[c_t; h_t]) \qquad (6)$$

Here, $\breve{W}$ is a new learnable parameter matrix, with which we hope to release W from the pressure of having to extract information relevant to both the word- and character-generation processes. Only the hidden state of the first layer

is initialized as discussed above. The other components in the character-level decoder such as the LSTM cells of all layers and the hidden states of higher layers, all start with zero values.

Implementation-wise, the computation in the character-level decoder is done per word *token* instead of per *type* as in the source character component (§4.2). This is because of the context-dependent nature of the decoder.

Word-Character Generation Strategy With the character-level decoder, we can view the final hidden states as representations for the surface forms of unknown tokens and could have fed these to the next time step. However, we chose not to do so for the efficiency reason explained next; instead, *<unk>* is fed to the word-level decoder "as is" using its corresponding word embedding.

During *training*, this design choice decouples all executions over *<unk>* instances of the character-level decoder as soon the word-level NMT completes. As such, the forward and backward passes of the character-level decoder over rare words can be invoked in batch mode. At *test* time, our strategy is to first run a beam search decoder at the word level to find the best translations given by the word-level NMT. Such translations contains *<unk>* tokens, so we utilize our character-level decoder with beam search to generate actual words for these *<unk>*.

5 Experiments

We evaluate the effectiveness of our models on the publicly available WMT'15 translation task from English into Czech with *newstest2013* (3000 sentences) as a development set and *newstest2015* (2656 sentences) as a test set. Two metrics are used: case-sensitive NIST BLEU (Papineni et al., 2002) and chrF$_3$ (Popović, 2015).[3] The latter measures the amounts of overlapping character n-grams and has been argued to be a better metric for translation tasks out of English.

5.1 Data

Among the available language pairs in WMT'15, all involving English, we choose *Czech* as a target language for several reasons. First and foremost, Czech is a Slavic language with not only rich and

---

[3] For NIST BLEU, we first run `detokenizer.pl` and then use `mteval-v13a` to compute the scores as per WMT guideline. For chrF$_3$, we utilize the implementation here `https://github.com/rsennrich/subword-nmt`.

	English		Czech	
	word	char	word	char
# Sents	15.8M			
# Tokens	254M	1,269M	224M	1,347M
# Types	1,172K	2003	1,760K	2053
200-char	98.1%		98.8%	

Table 1: **WMT'15 English-Czech data** – shown are various statistics of our training data such as *sentence*, *token* (word and character counts), as well as *type* (sizes of the word and character vocabularies). We show in addition the amount of words in a vocabulary expressed by a list of 200 characters found in frequent words.

complex inflection, but also fusional morphology in which a single morpheme can encode multiple grammatical, syntactic, or semantic meanings. As a result, Czech possesses an enormously large vocabulary (about 1.5 to 2 times bigger than that of English according to statistics in Table 1) and is a challenging language to translate into. Furthermore, this language pair has a large amount of training data, so we can evaluate at scale. Lastly, though our techniques are language independent, it is easier for us to work with Czech since Czech uses the Latin alphabet with some diacritics.

In terms of preprocessing, we apply only the standard tokenization practice.[4] We choose for each language a list of 200 characters found in frequent words, which, as shown in Table 1, can represent more than 98% of the vocabulary.

5.2 Training Details

We train three types of systems, purely *word-based*, purely *character-based*, and *hybrid*. Common to these architectures is a word-based NMT since the character-based systems are essentially word-based ones with longer sequences and the core of hybrid models is also a word-based NMT.

In training word-based NMT, we follow Luong et al. (2015a) to use the global attention mechanism together with similar hyperparameters: (a) deep LSTM models, 4 layers, 1024 cells, and 1024-dimensional embeddings, (b) uniform initialization of parameters in $[-0.1, 0.1]$, (c) 6-epoch training with plain SGD and a simple learning rate schedule – start with a learning rate of 1.0; after 4 epochs, halve the learning rate every 0.5 epoch, (d) mini-batches are of size 128 and shuf-

---

[4] Use `tokenizer.perl` in Moses with default settings.

	System	Vocab	Perplexity		BLEU	chrF$_3$
			w	c		
(a)	Best WMT'15, big data (Bojar and Tamchyna, 2015)	-	-	-	*18.8*	-
	Existing NMT					
(b)	RNNsearch + unk replace (Jean et al., 2015b)	200K	-	-	15.7	-
(c)	*Ensemble* 4 models + unk replace (Jean et al., 2015b)	200K	-	-	18.3	-
	Our *word-based* NMT					
(d)	Base + attention + unk replace	50K	5.9	-	17.5	42.4
(e)	*Ensemble* 4 models + unk replace	50K	-	-	18.4	43.9
	Our *character-based* NMT					
(f)	Base-512 (600-step backprop)	200	-	2.4	3.8	25.9
(g)	Base-512 + attention (600-step backprop)	200	-	1.6	17.5	*46.6*
(h)	Base-1024 + attention (300-step backprop)	200	-	1.9	15.7	41.1
	Our *hybrid* NMT					
(i)	Base + attention + same-path	10K	4.9	1.7	14.1	37.2
(j)	Base + attention + separate-path	10K	4.9	1.7	15.6	39.6
(k)	Base + attention + separate-path + 2-layer char	10K	4.7	1.6	*17.7*	44.1
(l)	Base + attention + separate-path + 2-layer char	50K	5.7	1.6	19.6	46.5
(m)	*Ensemble* 4 models	50K	-	-	**20.7**	**47.5**

Table 2: **WMT'15 English-Czech results** – shown are the vocabulary sizes, perplexities, BLEU, and chrF$_3$ scores of various systems on *newstest2015*. Perplexities are listed under two categories, word (w) and character (c). **Best** and *important* results per metric are highlighted.

fled, (e) the gradient is rescaled whenever its norm exceeds 5, and (f) dropout is used with probability 0.2 according to (Pham et al., 2014). We now detail differences across the three architectures.

Word-based NMT – We constrain our source and target sequences to have a maximum length of 50 each; words that go past the boundary are ignored. The vocabularies are limited to the top $|V|$ most frequent words in both languages. Words not in these vocabularies are converted into *<unk>*. After translating, we will perform dictionary[5] lookup or identity copy for *<unk>* using the alignment information from the attention models. Such procedure is referred as the *unk replace* technique (Luong et al., 2015b; Jean et al., 2015a).

Character-based NMT – The source and target sequences at the character level are often about 5 times longer than their counterparts in the word-based models as we can infer from the statistics in Table 1. Due to memory constraint in GPUs, we limit our source and target sequences to a maximum length of 150 each, i.e., we backpropagate through at most 300 timesteps from the decoder to the encoder. With smaller 512-dimensional models, we can afford to have longer sequences with

up to 600-step backpropagation.

Hybrid NMT – The *word*-level component uses the same settings as the purely word-based NMT. For the *character*-level source and target components, we experiment with both shallow and deep 1024-dimensional models of 1 and 2 LSTM layers. We set the weight α in Eq. (5) for our character-level loss to 1.0.

Training Time – It takes about 3 weeks to train a word-based model with $|V| = 50K$ and about 3 months to train a character-based model. Training and testing for the hybrid models are about 10-20% slower than those of the word-based models with the same vocabulary size.

5.3 Results

We compare our models with several strong systems. These include the winning entry in WMT'15, which was trained on a much larger amount of data, 52.6M parallel and 393.0M monolingual sentences (Bojar and Tamchyna, 2015).[6] In contrast, we merely use the provided parallel corpus of 15.8M sentences. For NMT, to the best

---

[5]Obtained from the alignment links produced by the Berkeley aligner (Liang et al., 2006) over the training corpus.

[6]This entry combines two independent systems, a phrase-based Moses model and a deep-syntactic transfer-based model. Additionally, there is an automatic post-editing system with hand-crafted rules to correct errors in morphological agreement and semantic meanings, e.g., loss of negation.

of our knowledge, (Jean et al., 2015b) has the best published performance on English-Czech.

As shown in Table 2, for a purely *word-based* approach, our single NMT model outperforms the best single model in (Jean et al., 2015b) by +1.8 points despite using a smaller vocabulary of only 50K words versus 200K words. Our ensemble system *(e)* slightly outperforms the best previous NMT system with 18.4 BLEU.

To our surprise, purely *character-based* models, though extremely slow to train and test, perform quite well. The 512-dimensional attention-based model *(g)* is best, surpassing the single word-based model in (Jean et al., 2015b) despite having much fewer parameters. It even outperforms most NMT systems on chrF$_3$ with 46.6 points. This indicates that this model translate words that closely but not exactly match the reference ones as evidenced in Section 6.3. We notice two interesting observations. First, attention is critical for character-based models to work as is obvious from the poor performance of the non-attentional model; this has also been shown in speech recognition (Chan et al., 2016). Second, long time-step backpropagation is more important as reflected by the fact that the larger 1024-dimensional model *(h)* with shorter backprogration is inferior to *(g)*.

Our *hybrid* models achieve the best results. At 10K words, we demonstrate that our *separate-path* strategy for the character-level target generation (§4.3) is effective, yielding an improvement of +1.5 BLEU points when comparing systems *(j)* vs. *(i)*. A *deeper* character-level architecture of 2 LSTM layers provides another significant boost of +2.1 BLEU. With 17.7 BLEU points, our hybrid system *(k)* has surpassed word-level NMT models.

When extending to 50K words, we further improve the translation quality. Our best single model, system *(l)* with 19.6 BLEU, is already better than all existing systems. Our ensemble model *(m)* further advances the SOTA result to **20.7** BLEU, outperforming the winning entry in the WMT'15 English-Czech translation task by a large margin of +1.9 points. Our ensemble model is also best in terms of chrF$_3$ with **47.5** points.

6 Analysis

This section first studies the effects of vocabulary sizes towards translation quality. We then analyze more carefully our character-level components by visualizing and evaluating rare word embeddings
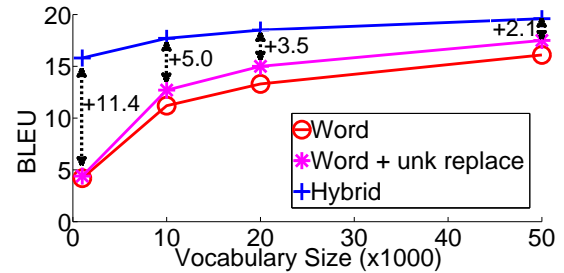

Figure 3: **Vocabulary size effect** – shown are the performances of different systems as we vary their vocabulary sizes. We highlight the improvements obtained by our hybrid models over word-based systems which already handle unknown words.

as well as examining sample translations.

6.1 Effects of Vocabulary Sizes

As shown in Figure 3, our hybrid models offer large gains of +2.1-11.4 BLEU points over strong word-based systems which already handle unknown words. With only a small vocabulary, e.g., 1000 words, our hybrid approach can produce systems that are better than word-based models that possess much larger vocabularies. While it appears from the plot that gains diminish as we increase the vocabulary size, we argue that our hybrid models are still preferable since they understand word structures and can handle new complex words at test time as illustrated in Section 6.3.

6.2 Rare Word Embeddings

We evaluate the *source* character-level model by building representations for rare words and measuring how good these embeddings are.

Quantitatively, we follow Luong et al. (2013) in using the word similarity task, specifically on the *Rare Word* dataset, to judge the learned representations for complex words. The evaluation metric is the Spearman's correlation ρ between similarity scores assigned by a model and by human annotators. From the results in Table 3, we can see that source representations produced by our hybrid[7] models are significantly better than those of the word-based one. It is noteworthy that our deep recurrent character-level models can outperform the model of (Luong et al., 2013), which uses recursive neural networks and requires a complex morphological analyzer, by a large margin. Our performance is also competitive to the best Glove

---

[7]We look up the encoder embeddings for frequent words and build representations for rare word from characters.
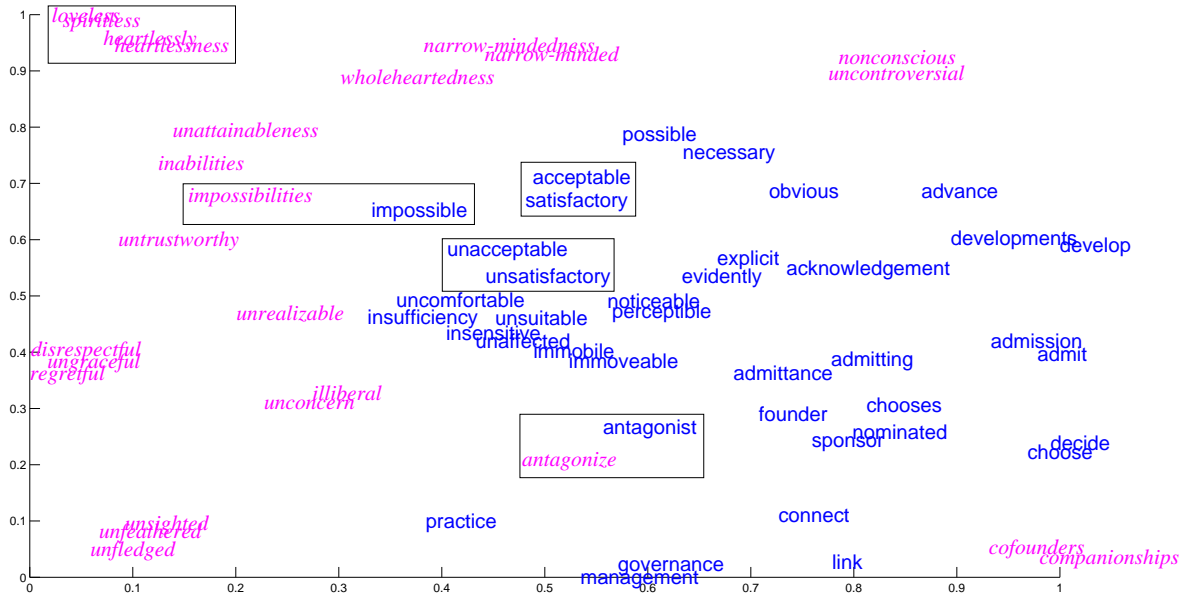
Figure 4: **Barnes-Hut-SNE visualization of source word representations** – shown are sample words from the *Rare Word* dataset. We differentiate two types of embeddings: frequent words in which encoder embeddings are looked up directly and *rare* words where we build representations from characters. Boxes highlight examples that we will discuss in the text. We use the hybrid model *(l)* in this visualization.

embeddings (Pennington et al., 2014) which were trained on a much larger dataset.

	System	Size	$|V|$	ρ
	(Luong et al., 2013)	1B	138K	34.4
	Glove (Pennington et al., 2014)	6B	400K	38.1
		42B	400K	**47.8**
	Our NMT models			
(d)	Word-based	0.3B	50K	20.4
(k)	Hybrid	0.3B	10K	42.4
(l)	Hybrid	0.3B	50K	***47.1***

Table 3: **Word similarity task** – shown are Spearman's correlation ρ on the *Rare Word* dataset of various models (with different vocab sizes $|V|$).

Qualitatively, we visualize embeddings produced by the hybrid model *(l)* for selected words in the Rare Word dataset. Figure 4 shows the two-dimensional representations of words computed by the Barnes-Hut-SNE algorithm (van der Maaten, 2013).[8] It is extremely interesting to observe that words are clustered together not only by the word structures but also by the meanings. For example, in the top-left box, the *character-based representations for "loveless", "spiritless", "heartlessly", and "heartlessness" are nearby, but clearly separated into two groups. Similarly, in the

---

[8] We run Barnes-Hut-SNE algorithm over a set of 91 words, but filter out 27 words for displaying clarity.

center boxes, *word*-based embeddings of "acceptable", "satisfactory", "unacceptable", and "unsatisfactory", are close by but separated by meanings. Lastly, the remaining boxes demonstrate that our character-level models are able to build representations comparable to the word-based ones, e.g., "impossibilities" vs. "impossible" and "antagonize" vs. "antagonist". All of this evidence strongly supports that the source character-level models are useful and effective.

6.3 Sample Translations

We show in Table 4 sample translations between various systems. In the first example, our hybrid model translates perfectly. The word-based model fails to translate "diagnosis" because the second $<unk>$ was incorrectly aligned to the word "after". The character-based model, on the other hand, makes a mistake in translating names.

For the second example, the hybrid model surprises us when it can capture the long-distance reordering of "fifty years ago" and "před padesáti lety" while the other two models do not. The word-based model translates "Jr." inaccurately due to the incorrect alignment between the second $<unk>$ and the word "said". The character-based model literally translates the name "King" into "král" which means "king".

Lastly, both the character-based and hybrid

1	source	The author **Stephen Jay Gould** died 20 years after **diagnosis** .
	human	Autor **Stephen Jay Gould** zemřel 20 let po **diagnóze** .
	word	Autor Stephen Jay *<unk>* zemřel 20 let po *<unk>* .
		Autor **Stephen Jay Gould** zemřel 20 let po **po** .
	char	Autor **Stepher Stepher** zemřel 20 let po **diagnóze** .
	hybrid	Autor *<unk>* *<unk>* *<unk>* zemřel 20 let po *<unk>*.
		Autor **Stephen Jay Gould** zemřel 20 let po **diagnóze** .
2	source	As the Reverend **Martin Luther King Jr.** said **fifty years ago** :
	human	Jak **před padesáti lety** řekl reverend **Martin Luther King Jr** . :
	word	Jak řekl reverend Martin *<unk>* King *<unk>* před padesáti lety :
		Jak řekl reverend **Martin Luther King** **řekl** před padesáti lety :
	char	Jako reverend **Martin Luther** **král říkal** před padesáti lety :
	hybrid	Jak před *<unk>* lety řekl *<unk>* Martin *<unk>* *<unk>* *<unk>* :
		Jak **před padesáti lety** řekl reverend **Martin Luther King** Jr. :
3	source	Her **11-year-old** daughter , **Shani Bart** , said it felt a " little bit **weird** " [..] back to school .
	human	Její **jedenáctiletá** dcera **Shani Bartová** prozradila , že " je to trochu **zvláštní** " [..] znova do školy .
	word	Její *<unk>* dcera *<unk>* *<unk>* řekla , že je to " trochu divné " , [..] vrací do školy .
		Její **11-year-old** dcera **Shani** **,** řekla , že je to " trochu divné " , [..] vrací do školy .
	char	Její **jedenáctiletá** dcera , **Shani Bartová** , říkala , že cítí trochu divně , [..] vrátila do školy .
	hybrid	Její *<unk>* dcera , *<unk>* *<unk>* , řekla , že cítí " trochu *<unk>* " , [..] vrátila do školy .
		Její **jedenáctiletá** dcera , **Graham** Bart , řekla , že cítí " trochu divný " , [..] vrátila do školy .

Table 4: **Sample translations on newstest2015** – for each example, we show the *source*, *human* translation, and translations of the following NMT systems: *word* model *(d)*, *char* model *(g)*, and *hybrid* model *(k)*. We show the translations before replacing *<unk>* tokens (if any) for the word-based and hybrid models. The following formats are used to highlight ***correct***, **wrong**, and <u>close</u> translation segments.

models impress us by their ability to translate compound words exactly, e.g., "11-year-old" and "jedenáctiletá"; whereas the identity copy strategy of the word-based model fails. Of course, our hybrid model does make mistakes, e.g., it fails to translate the name "Shani Bart". Overall, these examples highlight how challenging translating into Czech is and that being able to translate at the character level helps improve the quality.

7 Conclusion

We have proposed a novel *hybrid* architecture that combines the strength of both word- and character-based models. Word-level models are fast to train and offer high-quality translation; whereas, character-level models help achieve the goal of open vocabulary NMT. We have demonstrated these two aspects through our experimental results and translation examples.

Our best hybrid model has surpassed the performance of both the best word-based NMT system and the best non-neural model to establish a new state-of-the-art result for English-Czech translation in WMT'15 with 20.7 BLEU. Moreover, we have succeeded in replacing the standard unk replacement technique in NMT with our character-level components, yielding an improvement of +2.1−11.4 BLEU points. Our analysis has shown that our model has the ability to not only generate well-formed words for Czech, a highly inflected language with an enormous and complex vocabulary, but also build accurate representations for English source words.

Additionally, we have demonstrated the potential of purely character-based models in producing good translations; they have outperformed past word-level NMT models. For future work, we hope to be able to improve the memory usage and speed of purely character-based models.

Acknowledgments

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *EMNLP*.

Ondřej Bojar and Aleš Tamchyna. 2015. CUNI in WMT15: Chimera Strikes Again. In *WMT*.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell. In *ICASSP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. 9(8):1735–1780.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015a. On using very large target vocabulary for neural machine translation. In *ACL*.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015b. Montreal neural machine translation systems for WMT'15. In *WMT*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *AAAI*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *NAACL*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan Black. 2015b. Character-based neural machine translation.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *IWSLT*.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *ICFHR*.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Laurens van der Maaten. 2013. Barnes-Hut-SNE. In *ICLR*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. abs/1409.2329.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.