On Using Very Large Target Vocabulary for Neural Machine Translation

Sébastien Jean **Kyunghyun Cho**
Roland Memisevic
Université de Montréal

Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Abstract

Neural machine translation, a recently proposed approach to machine translation based purely on neural networks, has shown promising results compared to the existing approaches such as phrase-based statistical machine translation. Despite its recent success, neural machine translation has its limitation in handling a larger vocabulary, as training complexity as well as decoding complexity increase proportionally to the number of target words. In this paper, we propose a method based on importance sampling that allows us to use a very large target vocabulary without increasing training complexity. We show that decoding can be efficiently done even with the model having a very large target vocabulary by selecting only a small subset of the whole target vocabulary. The models trained by the proposed approach are empirically found to match, and in some cases outperform, the baseline models with a small vocabulary as well as the LSTM-based neural machine translation models. Furthermore, when we use an ensemble of a few models with very large target vocabularies, we achieve performance comparable to the state of the art (measured by BLEU) on both the English→German and English→French translation tasks of WMT'14.

1 Introduction

Neural machine translation (NMT) is a recently introduced approach to solving machine translation (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2015; Sutskever et al., 2014). In neural machine translation, one builds a single neural network that reads a source sentence and generates its translation. The whole neural network is jointly trained to maximize the conditional probability of a correct translation given a source sentence, using the bilingual corpus. The NMT models have shown to perform as well as the most widely used conventional translation systems (Sutskever et al., 2014; Bahdanau et al., 2015).

Neural machine translation has a number of advantages over the existing statistical machine translation system, specifically, the phrase-based system (Koehn et al., 2003). First, NMT requires a minimal set of domain knowledge. For instance, all of the models proposed in (Sutskever et al., 2014), (Bahdanau et al., 2015) or (Kalchbrenner and Blunsom, 2013) do not assume any linguistic property in both source and target sentences except that they are sequences of words. Second, the whole system is jointly trained to maximize the translation performance, unlike the existing phrase-based system which consists of many separately trained features whose weights are then tuned jointly. Lastly, the memory footprint of the NMT model is often much smaller than the existing system which relies on maintaining large tables of phrase pairs.

Despite these advantages and promising results, there is a major limitation in NMT compared to the existing phrase-based approach. That is, the number of target words must be limited. This is mainly because the complexity of training and using an NMT model increases as the number of target words increases.

A usual practice is to construct a target vocabulary of the K most frequent words (a so-called shortlist), where K is often in the range of $30k$ (Bahdanau et al., 2015) to $80k$ (Sutskever et al., 2014). Any word not included in this vocabulary is mapped to a special token representing an *unknown* word [UNK]. This approach works well when there are only a few unknown words in the target sentence, but it has been observed

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1–10,
Beijing, China, July 26-31, 2015.

that the translation performance degrades rapidly as the number of unknown words increases (Cho et al., 2014a; Bahdanau et al., 2015).

In this paper, we propose an approximate training algorithm based on (biased) importance sampling that allows us to train an NMT model with a much larger target vocabulary. The proposed algorithm effectively keeps the computational complexity during training at the level of using only a small subset of the full vocabulary. Once the model with a very large target vocabulary is trained, one can choose to use either all the target words or only a subset of them.

We compare the proposed algorithm against the baseline shortlist-based approach in the tasks of English→French and English→German translation using the NMT model introduced in (Bahdanau et al., 2015). The empirical results demonstrate that we can potentially achieve better translation performance using larger vocabularies, and that our approach does not sacrifice too much speed for both training and decoding. Furthermore, we show that the model trained with this algorithm gets the best translation performance yet achieved by single NMT models on the WMT'14 English→French translation task.

2 Neural Machine Translation and Limited Vocabulary Problem

In this section, we briefly describe an approach to neural machine translation proposed recently in (Bahdanau et al., 2015). Based on this description we explain the issue of limited vocabularies in neural machine translation.

2.1 Neural Machine Translation

Neural machine translation is a recently proposed approach to machine translation, which uses a single neural network trained jointly to maximize the translation performance (Forcada and Ñeco, 1997; Kalchbrenner and Blunsom, 2013; Cho et al., 2014b; Sutskever et al., 2014; Bahdanau et al., 2015).

Neural machine translation is often implemented as the encoder–decoder network. The encoder reads the source sentence $x = (x_1, \dots, x_T)$ and encodes it into a sequence of hidden states $h = (h_1, \cdots, h_T)$:

$$h_t = f\left(x_t, h_{t-1}\right). \tag{1}$$

Then, the decoder, another recurrent neural network, generates a corresponding translation $y = (y_1, \cdots, y_{T'})$ based on the encoded sequence of hidden states h:

$$p(y_t \mid y_{<t}, x) \propto \exp\left\{q\left(y_{t-1}, z_t, c_t\right)\right\}, \tag{2}$$

where

$$z_t = g\left(y_{t-1}, z_{t-1}, c_t\right), \tag{3}$$
$$c_t = r\left(z_{t-1}, h_1, \dots, h_T\right), \tag{4}$$

and $y_{<t} = (y_1, \dots, y_{t-1})$.

The whole model is jointly trained to maximize the conditional log-probability of the correct translation given a source sentence with respect to the parameters $\boldsymbol{\theta}$ of the model:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n \mid y_{<t}^n, x^n),$$

where (x^n, y^n) is the n-th training pair of sentences, and T_n is the length of the n-th target sentence (y^n).

2.1.1 Detailed Description

In this paper, we use a specific implementation of neural machine translation that uses an attention mechanism, as recently proposed in (Bahdanau et al., 2015).

In (Bahdanau et al., 2015), the encoder in Eq. (1) is implemented by a bi-directional recurrent neural network such that

$$h_t = \left[\overleftarrow{h}_t; \overrightarrow{h}_t\right],$$

where

$$\overleftarrow{h}_t = f\left(x_t, \overleftarrow{h}_{t+1}\right), \overrightarrow{h}_t = f\left(x_t, \overrightarrow{h}_{t-1}\right).$$

They used a gated recurrent unit for f (see, e.g., (Cho et al., 2014b)).

The decoder, at each time, computes the context vector c_t as a convex sum of the hidden states $(h_1, \dots, h_T)$ with the coefficients $\alpha_1, \dots, \alpha_T$ computed by

$$\alpha_t = \frac{\exp\left\{a\left(h_t, z_{t-1}\right)\right\}}{\sum_k \exp\left\{a\left(h_k, z_{t-1}\right)\right\}}, \tag{5}$$

where a is a feedforward neural network with a single hidden layer.

A new hidden state z_t of the decoder in Eq. (3) is computed based on the previous hidden state z_{t-1}, previous generated symbol y_{t-1} and the computed

context vector c_t. The decoder also uses the gated recurrent unit, as the encoder does.

The probability of the next target word in Eq. (2) is then computed by

$$p(y_t \mid y_{<t}, x) = \frac{1}{Z} \exp\left\{\mathbf{w}_t^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_t\right\}, \quad (6)$$

where ϕ is an affine transformation followed by a nonlinear activation, and $\mathbf{w}_t$ and b_t are respectively the *target word vector* and the target word bias. Z is the normalization constant computed by

$$Z = \sum_{k:y_k \in V} \exp\left\{\mathbf{w}_k^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_k\right\}, \quad (7)$$

where V is the set of all the target words.

For the detailed description of the implementation, we refer the reader to the appendix of (Bahdanau et al., 2015).

2.2 Limited Vocabulary Issue and Conventional Solutions

One of the main difficulties in training this neural machine translation model is the computational complexity involved in computing the target word probability (Eq. (6)). More specifically, we need to compute the dot product between the feature $\phi\left(y_{t-1}, z_t, c_t\right)$ and the word vector w_t as many times as there are words in a target vocabulary in order to compute the normalization constant (the denominator in Eq. (6)). This has to be done for, on average, 20–30 words per sentence, which easily becomes prohibitively expensive even with a moderate number of possible target words. Furthermore, the memory requirement grows linearly with respect to the number of target words. This has been a major hurdle for neural machine translation, compared to the existing non-parametric approaches such as phrase-based translation systems.

Recently proposed neural machine translation models, hence, use a shortlist of 30k to 80k most frequent words (Bahdanau et al., 2015; Sutskever et al., 2014). This makes training more feasible, but comes with a number of problems. First of all, the performance of the model degrades heavily if the translation of a source sentence requires many words that are not included in the shortlist (Cho et al., 2014a). This also affects the performance evaluation of the system which is often measured by BLEU. Second, the first issue becomes more problematic with languages that have a rich set of words such as German or other highly inflected languages.

There are two *model-specific* approaches to this issue of large target vocabulary. The first approach is to stochastically approximate the target word probability. This has been proposed recently in (Mnih and Kavukcuoglu, 2013; Mikolov et al., 2013) based on noise-contrastive estimation (Gutmann and Hyvarinen, 2010). In the second approach, the target words are clustered into multiple classes, or hierarchical classes, and the target probability $p(y_t|y_{<t}, x)$ is factorized as a product of the class probability $p(c_t|y_{<t}, x)$ and the intra-class word probability $p(y_t|c_t, y_{<t}, x)$. This reduces the number of required dot-products into the sum of the number of classes and the words in a class. These approaches mainly aim at reducing the computational complexity during training, but do not often result in speed-up when decoding a translation during test time.[1]

Other than these model-specific approaches, there exist *translation-specific* approaches. A translation-specific approach exploits the properties of the rare target words. For instance, Luong et al. proposed such an approach for neural machine translation (Luong et al., 2015). They replace rare words (the words that are not included in the shortlist) in both source and target sentences into corresponding $\langle \text{OOV}_n \rangle$ tokens using the word alignment model. Once a source sentence is translated, each $\langle \text{OOV}_n \rangle$ in the translation will be replaced based on the source word marked by the corresponding $\langle \text{OOV}_n \rangle$.

It is important to note that the model-specific approaches and the translation-specific approaches are often complementary and can be used together to further improve the translation performance and reduce the computational complexity.

3 Approximate Learning Approach to Very Large Target Vocabulary

3.1 Description

In this paper, we propose a *model-specific* approach that allows us to train a neural machine translation model with a very large target vocabulary. With the proposed approach, the compu-

[1] This is due to the fact that the beam search requires the conditional probability of *every* target word at each time step regardless of the parametrization of the output probability.

tational complexity of training becomes constant with respect to the size of the target vocabulary. Furthermore, the proposed approach allows us to efficiently use a fast computing device with limited memory, such as a GPU, to train a neural machine translation model with a much larger target vocabulary.

As mentioned earlier, the computational inefficiency of training a neural machine translation model arises from the normalization constant in Eq. (6). In order to avoid the growing complexity of computing the normalization constant, we propose here to use only a small subset V' of the target vocabulary at each update. The proposed approach is based on the earlier work of (Bengio and Sénécal, 2008).

Let us consider the gradient of the log-probability of the output in Eq. (6). The gradient is composed of a positive and negative part:

$$\begin{aligned}&\nabla \log p(y_t \mid y_{<t}, x) \qquad (8)\\ &\quad = \nabla \mathcal{E}(y_t) - \sum_{k:y_k \in V} p(y_k \mid y_{<t}, x) \nabla \mathcal{E}(y_k),\end{aligned}$$

where we define the energy $\mathcal{E}$ as

$$\mathcal{E}(y_j) = \mathbf{w}_j^\top \phi\left(y_{j-1}, z_j, c_j\right) + b_j.$$

The second, or negative, term of the gradient is in essence the expected gradient of the energy:

$$\mathbb{E}_P\left[\nabla \mathcal{E}(y)\right], \qquad (9)$$

where P denotes $p(y \mid y_{<t}, x)$.

The main idea of the proposed approach is to approximate this expectation, or the negative term of the gradient, by importance sampling with a small number of samples. Given a predefined proposal distribution Q and a set V' of samples from Q, we approximate the expectation in Eq. (9) with

$$\mathbb{E}_P\left[\nabla \mathcal{E}(y)\right] \approx \sum_{k:y_k \in V'} \frac{\omega_k}{\sum_{k':y_{k'} \in V'} \omega_{k'}} \nabla \mathcal{E}(y_k), \qquad (10)$$

where

$$\omega_k = \exp\left\{\mathcal{E}(y_k) - \log Q(y_k)\right\}. \qquad (11)$$

This approach allows us to compute the normalization constant during training using only a small subset of the target vocabulary, resulting in much lower computational complexity for each parameter update. Intuitively, at each parameter update, we update only the vectors associated with the correct word $\mathbf{w}_t$ and with the sampled words in V'. Once training is over, we can use the full target vocabulary to compute the output probability of each target word.

Although the proposed approach naturally addresses the computational complexity, using this approach naively does not guarantee that the number of parameters being updated for each sentence pair, which includes multiple target words, is bounded nor can be controlled. This becomes problematic when training is done, for instance, on a GPU with limited memory.

In practice, hence, we partition the training corpus and define a subset V' of the target vocabulary for each partition prior to training. Before training begins, we sequentially examine each target sentence in the training corpus and accumulate unique target words until the number of unique target words reaches the predefined threshold τ. The accumulated vocabulary will be used for this partition of the corpus during training. We repeat this until the end of the training set is reached. Let us refer to the subset of target words used for the i-th partition by V'_i.

This may be understood as having a separate proposal distribution Q_i for each partition of the training corpus. The distribution Q_i assigns equal probability mass to all the target words included in the subset V'_i, and zero probability mass to all the other words, i.e.,

$$Q_i(y_k) = \begin{cases} \frac{1}{|V'_i|} & \text{if } y_t \in V'_i \\ 0 & \text{otherwise.} \end{cases}$$

This choice of proposal distribution cancels out the correction term $-\log Q(y_k)$ from the importance weight in Eqs. (10)–(11), which makes the proposed approach equivalent to approximating the exact output probability in Eq. (6) with

$$\begin{aligned}&p(y_t \mid y_{<t}, x)\\ &\quad = \frac{\exp\left\{\mathbf{w}_t^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_t\right\}}{\sum_{k:y_k \in V'} \exp\left\{\mathbf{w}_k^\top \phi\left(y_{t-1}, z_t, c_t\right) + b_k\right\}}.\end{aligned}$$

It should be noted that this choice of Q makes the estimator biased.

The proposed procedure results in speed up against usual importance sampling, as it exploits the advantage of modern computers in doing matrix-matrix vs matrix-vector multiplications.

3.1.1 Informal Discussion on Consequence

The parametrization of the output probability in Eq. (6) can be understood as arranging the vectors associated with the target words such that the dot product between the most likely, or correct, target word's vector and the current hidden state is maximized. The exponentiation followed by normalization is simply a process in which the dot products are converted into proper probabilities.

As learning continues, therefore, the vectors of all the likely target words tend to align with each other but not with the others. This is achieved exactly by moving the vector of the correct word in the direction of $\phi(y_{t-1}, z_t, c_t)$, while pushing all the other vectors away, which happens when the gradient of the logarithm of the exact output probability in Eq. (6) is maximized. Our approximate approach, instead, moves the word vectors of the correct words and of only a subset of sampled target words (those included in V').

3.2 Decoding

Once the model is trained using the proposed approximation, we can use the full target vocabulary when decoding a translation given a new source sentence. Although this is advantageous as it allows the trained model to utilize the whole vocabulary when generating a translation, doing so may be too computationally expensive, e.g., for real-time applications.

Since training puts the target word vectors in the space so that they align well with the hidden state of the decoder only when they are likely to be a correct word, we can use only a subset of candidate target words during decoding. This is similar to what we do during training, except that at test time, we do not have access to a set of correct target words.

The most naïve way to select a subset of candidate target words is to take only the top-K most frequent target words, where K can be adjusted to meet the computational requirement. This, however, effectively cancels out the whole purpose of training a model with a very large target vocabulary. Instead, we can use an existing word alignment model to align the source and target words in the training corpus and build a dictionary. With the dictionary, for each source sentence, we construct a target word set consisting of the K-most frequent words (according to the estimated unigram probability) and, using the dictionary, at most K' likely target words for each source word. K and K' may be chosen either to meet the computational requirement or to maximize the translation performance on the development set. We call a subset constructed in either of these ways a *candidate list*.

3.3 Source Words for Unknown Words

In the experiments, we evaluate the proposed approach with the neural machine translation model called RNNsearch (Bahdanau et al., 2015) (see Sec. 2.1.1). In this model, as a part of decoding process, we obtain the alignments between the target words and source locations via the alignment model in Eq. (5).

We can use this feature to infer the source word to which each target word was most aligned (indicated by the largest α_t in Eq. (5)). This is especially useful when the model generated an [UNK] token. Once a translation is generated given a source sentence, each [UNK] may be replaced using a translation-specific technique based on the aligned source word. For instance, in the experiment, we try replacing each [UNK] token with the aligned source word or its most likely translation determined by another word alignment model. Other techniques such as transliteration may also be used to further improve the performance (Koehn, 2010).

4 Experiments

We evaluate the proposed approach in English→French and English→German translation tasks. We trained the neural machine translation models using only the bilingual, parallel corpora made available as a part of WMT'14. For each pair, the datasets we used are:

- English→French:[2]
  - Common Crawl
  - News Commentary
  - Gigaword
  - Europarl v7
  - UN
- English→German:
  - Common Crawl
  - News Commentary
  - Europarl v7

[2]The preprocessed data can be found and downloaded from `http://www-lium.univ-lemans.fr/~schwenk/nnmt-shared-task/README`.

	English-French		English-German	
	Train	Test	Train	Test
15k	93.5	90.8	88.5	83.8
30k	96.0	94.6	91.8	87.9
50k	97.3	96.3	93.7	90.4
500k	99.5	99.3	98.4	96.1
All	100.0	99.6	100.0	97.3

Table 1: Data coverage (in %) on target-side corpora for different vocabulary sizes. "All" refers to all the tokens in the training set.

To ensure fair comparison, the English→French corpus, which comprises approximately 12 million sentences, is identical to the one used in (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2015; Sutskever et al., 2014). As for English→German, the corpus was preprocessed, in a manner similar to (Peitz et al., 2014; Li et al., 2014), in order to remove many poorly translated sentences.

We evaluate the models on the WMT'14 test set (news-test 2014),[3] while the concatenation of news-test-2012 and news-test-2013 is used for model selection (development set). Table 1 presents data coverage w.r.t. the vocabulary size, on the target side.

Unless mentioned otherwise, all reported BLEU scores (Papineni et al., 2002) are computed with the multi-bleu.perl script[4] on the cased tokenized translations.

4.1 Settings

As a baseline for English→French translation, we use the **RNNsearch** model proposed by (Bahdanau et al., 2015), with 30k source and target words.[5] Another RNNsearch model is trained for English→German translation with 50k source and target words.

For each language pair, we train another set of RNNsearch models with much larger vocabularies of 500k source and target words, using the proposed approach. We call these models **RNNsearch-LV**. We vary the size of the shortlist used during training (τ in Sec. 3.1). We tried 15k and 30k for English→French, and 15k and 50k for English→German. We later report the results for the best performance on the development set, with models generally evaluated every twelve hours. The training speed is approximately the same as for RNNsearch. Using a 780 Ti or Titan Black GPU, we could process 100k mini-batches of 80 sentences in about 29 and 39 hours respectively for $\tau = 15k$ and $\tau = 50k$.

For both language pairs, we also trained new models, with $\tau = 15k$ and $\tau = 50k$, by reshuffling the dataset at the beginning of each epoch. While this causes a non-negligible amount of overhead, such a change allows words to be contrasted with different sets of other words each epoch.

To stabilize parameters other than the word embeddings, at the end of the training stage, we freeze the word embeddings and tune only the other parameters for approximately two more days after the peak performance on the development set is observed. This helped increase BLEU scores on the development set.

We use beam search to generate a translation given a source. During beam search, we keep a set of 12 hypotheses and normalize probabilities by the length of the candidate sentences, as in (Cho et al., 2014a).[6] The candidate list is chosen to maximize the performance on the development set, for $K \in \{15k, 30k, 50k\}$ and $K' \in \{10, 20\}$. As explained in Sec. 3.2, we test using a bilingual dictionary to accelerate decoding and to replace unknown words in translations. The bilingual dictionary is built using *fast_align* (Dyer et al., 2013). We use the dictionary only if a word starts with a lowercase letter, and otherwise, we copy the source word directly. This led to better performance on the development sets.

Note on ensembles For each language pair, we began training four models from each of which two points corresponding to the best and second-best performance on the development set were collected. We continued training from each point, while keeping the word embeddings fixed, until the best development performance was reached, and took the model at this point as a single model in an ensemble. This procedure resulted in a total of eight models from which we averaged the length-normalized log-probabilities. Since much of training had been shared, the composition of

[3] To compare with previous submissions, we use the filtered test sets.

[4] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

[5] The authors of (Bahdanau et al., 2015) gave us access to their trained models. We chose the best one on the validation set and resumed training.

[6] These experimental details differ from (Bahdanau et al., 2015).

	RNNsearch	RNNsearch-LV	Google	Phrase-based SMT	
Basic NMT	29.97 (26.58)	32.68 (28.76)	30.6*		
+Candidate List	–	33.36 (29.32)	–		
+UNK Replace	33.08 (29.08)	34.11 (29.98)	33.1°	33.3*	37.03$^\bullet$
+Reshuffle (τ=50k)	–	34.60 (30.53)	–		
+Ensemble	–	37.19 (31.98)	37.5°		

(a) English→French

	RNNsearch	RNNsearch-LV	Phrase-based SMT
Basic NMT	16.46 (17.13)	16.95 (17.85)	
+Candidate List	–	17.46 (18.00)	
+UNK Replace	18.97 (19.16)	18.89 (19.03)	20.67$^\diamond$
+Reshuffle	–	19.40 (19.37)	
+Ensemble	–	21.59 (21.06)	

(b) English→German

Table 2: The translation performances in BLEU obtained by different models on (a) English→French and (b) English→German translation tasks. RNNsearch is the model proposed in (Bahdanau et al., 2015), RNNsearch-LV is the RNNsearch trained with the approach proposed in this paper, and Google is the LSTM-based model proposed in (Sutskever et al., 2014). Unless mentioned otherwise, we report single-model RNNsearch-LV scores using $\tau = 30k$ (English→French) and $\tau = 50k$ (English→German). For the experiments we have run ourselves, we show the scores on the development set as well in the brackets. ($\star$) (Sutskever et al., 2014), ($\circ$) (Luong et al., 2015), ($\bullet$) (Durrani et al., 2014), ($*$) Standard Moses Setting (Cho et al., 2014b), ($\diamond$) (Buck et al., 2014).

such ensembles may be sub-optimal. This is supported by the fact that higher cross-model BLEU scores (Freitag et al., 2014) are observed for models that were partially trained together.

4.2 Translation Performance

In Table 2, we present the results obtained by the trained models with very large target vocabularies, and alongside them, the previous results reported in (Sutskever et al., 2014), (Luong et al., 2015), (Buck et al., 2014) and (Durrani et al., 2014). Without translation-specific strategies, we can clearly see that the RNNsearch-LV outperforms the baseline RNNsearch.

In the case of the English→French task, RNNsearch-LV approached the performance level of the previous best single neural machine translation (NMT) model, even without any translation-specific techniques (Sec. 3.2–3.3). With these, however, the RNNsearch-LV outperformed it. The performance of the RNNsearch-LV is also better than that of a standard phrase-based translation system (Cho et al., 2014b). Furthermore, by combining 8 models, we were able to achieve a translation performance comparable to the state of the art, measured in BLEU.

For English→German, the RNNsearch-LV outperformed the baseline before unknown word replacement, but after doing so, the two systems performed similarly. We could reach higher large-vocabulary single-model performance by reshuffling the dataset, but this step could potentially also help the baseline. In this case, we were able to surpass the previously reported best translation result on this task by building an ensemble of 8 models.

With $\tau = 15k$, the RNNsearch-LV performance worsened a little, with best BLEU scores, without reshuffling, of 33.76 and 18.59 respectively for English→French and English→German.

The English→German ensemble described in this paper has also been used for the shared translation task of the 10$^{\text{th}}$ Workshop on Statistical Machine Translation (WMT'15), where it was ranked first in terms of BLEU score. The translations by this ensemble can be found online.[7]

4.3 Analysis

4.3.1 Decoding Speed

In Table 3, we present the timing information of decoding for different models. Clearly, decoding from RNNsearch-LV with the full target vocab-

[7] http://matrix.statmt.org/matrix/output/1774?run_id=4079

	CPU*	GPU$^\circ$
RNNsearch	0.09 s	0.02 s
RNNsearch-LV	0.80 s	0.25 s
RNNsearch-LV +Candidate list	0.12 s	0.05 s

Table 3: The average per-word decoding time. Decoding here does not include parameter loading and unknown word replacement. The baseline uses 30k words. The candidate list is built with $K = 30k$ and $K' = 10$. ($\star$) i7-4820K (single thread), ($\circ$) GTX TITAN Black

ulary is slowest. If we use a candidate list for decoding each translation, the speed of decoding substantially improves and becomes close to the baseline RNNsearch.

A potential issue with using a candidate list is that for each source sentence, we must re-build a target vocabulary and subsequently replace a part of the parameters, which may easily become time-consuming. We can address this issue, for instance, by building a common candidate list for multiple source sentences. By doing so, we were able to match the decoding speed of the baseline RNNsearch model.

4.3.2 Decoding Target Vocabulary

For English→French ($\tau = 30k$), we evaluate the influence of the target vocabulary when translating the test sentences by using the union of a fixed set of $30k$ common words and (at most) K' likely candidates for each source word according to the dictionary. Results are presented in Figure 1. With $K' = 0$ (not shown), the performance of the system is comparable to the baseline when not replacing the unknown words (30.12), but there is not as much improvement when doing so (31.14). As the large vocabulary model does not predict [UNK] as much during training, it is less likely to generate it when decoding, limiting the effectiveness of the post-processing step in this case. With $K' = 1$, which limits the diversity of allowed uncommon words, BLEU is not as good as with moderately larger K', which indicates that our models can, to some degree, correctly choose between rare alternatives. If we rather use $K = 50k$, as we did for testing based on validation performance, the improvement over $K' = 1$ is approximately 0.2 BLEU.

When validating the choice of K, we found it to be correlated with the value of τ used during training. For example, on the English→French validation set, with $\tau = 15k$ (and $K' = 10$), the BLEU score is 29.44 with $K = 15k$, but drops to 29.19 and 28.84 respectively for $K = 30k$ and $50k$. For $\tau = 30k$, the score increases moderately from $K = 15k$ to $K = 50k$. A similar effect was observed for English→German and on the test sets. As our implementation of importance sampling does not apply the usual correction to the gradient, it seems beneficial for the test vocabularies to resemble those used during training.

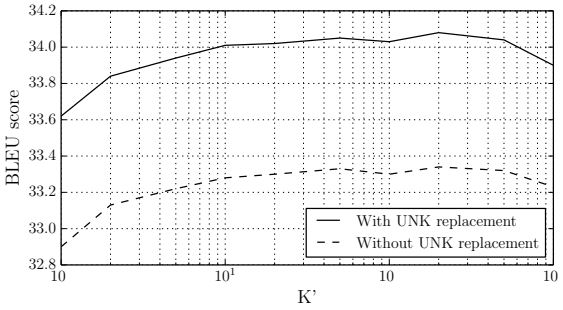

Figure 1: Single-model test BLEU scores (English→French) with respect to the number of dictionary entries K' allowed for each source word.

5 Conclusion

In this paper, we proposed a way to extend the size of the target vocabulary for neural machine translation. The proposed approach allows us to train a model with much larger target vocabulary without any substantial increase in computational complexity. It is based on the earlier work in (Bengio and Sénécal, 2008) which used importance sampling to reduce the complexity of computing the normalization constant of the output word probability in neural language models.

On English→French and English→German translation tasks, we observed that the neural machine translation models trained using the proposed method performed as well as, or better than, those using only limited sets of target words, even when replacing unknown words. As performance of the RNNsearch-LV models increased when only a selected subset of the target vocabulary was used during decoding, this makes the proposed learning algorithm more practical.

When measured by BLEU, our models showed translation performance comparable to the

state-of-the-art translation systems on both the English→French task and English→German task. On the English→French task, a model trained with the proposed approach outperformed the best single neural machine translation (NMT) model from (Luong et al., 2015) by approximately 1 BLEU point. The performance of the ensemble of multiple models, despite its relatively less diverse composition, is approximately 0.3 BLEU points away from the best system (Luong et al., 2015). On the English→German task, the best performance of 21.59 BLEU by our model is higher than that of the previous state of the art (20.67) reported in (Buck et al., 2014).

Finally, we release the source code used in our experiments to encourage progress in neural machine translation.[8]

Acknowledgments

The authors would like to thank the developers of Theano (Bergstra et al., 2010; Bastien et al., 2012). We acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs, CIFAR and Samsung.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR'2015, arXiv:1409.0473*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19(4):713–722.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.

[8] https://github.com/sebastien-j/LV_groundhog

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavík, Iceland, May.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–Decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, October.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, October.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. 2014. Edinburgh's phrase-based machine translation systems for WMT-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 97–104. Association for Computational Linguistics Baltimore, MD, USA.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 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*, pages 644–648, Atlanta, Georgia, June. Association for Computational Linguistics.

Mikel L. Forcada and Ramón P. Ñeco. 1997. Recursive hetero-associative memories for translation. In José Mira, Roberto Moreno-Díaz, and Joan Cabestany, editors, *Biological and Artificial Computation: From Neuroscience to Technology*, volume 1240 of *Lecture Notes in Computer Science*, pages 453–462. Springer Berlin Heidelberg.

Markus Freitag, Stephan Peitz, Joern Wuebker, Hermann Ney, Matthias Huck, Rico Sennrich, Nadir Durrani, Maria Nadejde, Philip Williams, Philipp Koehn, et al. 2014. Eu-bridge MT: Combined machine translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 105–113.

M. Gutmann and A. Hyvarinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Liangyou Li, Xiaofeng Wu, Santiago Cortes Vaillo, Jun Xie, Andy Way, and Qun Liu. 2014. The DCU-ICTCAS MT system at WMT 2014 on German-English translation task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 136–141, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations: Workshops Track*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephan Peitz, Joern Wuebker, Markus Freitag, and Hermann Ney. 2014. The RWTH Aachen German-English machine translation system for WMT 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 157–162, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

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