Abstract

Neural machine translation (NMT) offers a novel alternative formulation of translation that is potentially simpler than statistical approaches. However to reach competitive performance, NMT models need to be exceedingly large. In this paper we consider applying knowledge distillation approaches (Bucila et al., 2006; Hinton et al., 2015) that have proven successful for reducing the size of neural models in other domains to the problem of NMT. We demonstrate that standard knowledge distillation applied to word-level prediction can be effective for NMT, and also introduce two novel sequence-level versions of knowledge distillation that further improve performance, and somewhat surprisingly, seem to eliminate the need for beam search (even when applied on the original teacher model). Our best student model runs 10 times faster than its state-of-the-art teacher with little loss in performance. It is also significantly better than a baseline model trained without knowledge distillation: by 4.2/1.7 BLEU with greedy decoding/beam search. Applying weight pruning on top of knowledge distillation results in a student model that has $13 \times$ fewer parameters than the original teacher model, with a decrease of 0.4 BLEU.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) is a deep learning-based method for translation that has recently shown promising results as an alternative to statistical approaches. NMT systems directly model the probability of the next word in the target sentence simply by conditioning a recurrent neural sentence on the source sentence and previously generated target words.

While both simple and surprisingly accurate, NMT systems typically need to have very high capacity in order to perform well: Sutskever et al. (2014) used a 4-layer LSTM with 1000 hidden units per layer (herein $4 \times 1000$) and Zhou et al. (2016) obtained state-of-the-art results on English $\rightarrow$ French with a 16-layer LSTM with 512 units per layer. The sheer size of the models requires cutting-edge hardware for training and makes using the models on standard setups very challenging.

This issue of excessively large networks has been observed in several other domains, with much focus on fully-connected and convolutional networks for multi-class classification. Researchers have particularly noted that large networks seem to be necessary for training, but learn redundant representations in the process (Denil et al., 2013). Therefore compressing deep models into smaller networks has been an active area of research. As deep learning systems obtain better results on NLP tasks, compression also becomes an important practical issue with applications such as running deep learning models for speech and translation locally on cell phones.

Existing compression methods generally fall into two categories: (1) pruning and (2) knowledge distillation. Pruning methods (LeCun et al., 1990; He et al., 2014; Han et al., 2016) zero-out weights or entire neurons based on an importance criterion; LeCun et al. (1990) use (a diagonal approximation to)
the Hessian to identify weights whose removal minimally impacts the objective function, while Han et al. (2016) remove weights based on thresholding their absolute values. Knowledge distillation approaches (Bucila et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015) learn a smaller student network to mimic the original teacher network by minimizing the loss (typically $L_2$ or cross-entropy) between the student and teacher output.

In this work, we investigate knowledge distillation in the context of neural machine translation. We note that NMT differs from previous work which has mainly explored non-recurrent models in the multi-class prediction setting. For NMT, while the model is trained on multi-class prediction at the word-level, it is tasked with predicting complete sequence outputs conditioned on previous decisions. With this difference in mind, we experiment with standard knowledge distillation for NMT and also propose two new versions of the approach that attempt to approximately match the sequence-level (as opposed to word-level) distribution of the teacher network. This sequence-level approximation leads to a simple training procedure wherein the student network is trained on a newly generated dataset that is the result of running beam search with the teacher network.

We run experiments to compress a large state-of-the-art $4 \times 1000$ LSTM model, and find that with sequence-level knowledge distillation we are able to learn a $2 \times 500$ LSTM that roughly matches the performance of the full system. We see similar results compressing a $2 \times 500$ model down to $2 \times 100$ on a smaller data set. Furthermore, we observe that our proposed approach has other benefits, such as not requiring any beam search at test-time. As a result we are able to perform greedy decoding on the $2 \times 500$ model 10 times faster than beam search on the $4 \times 1000$ model with comparable performance. Our student models can even be run efficiently on a standard smartphone.

Finally, we apply weight pruning on top of the student network to obtain a model that has $13 \times$ fewer parameters than the original teacher model. We have released all the code for the models described in this paper.

---

2. https://github.com/harvardnlp/seq2seq-attn

---

2 Background

2.1 Sequence-to-Sequence with Attention

Let $s = [s_1, \ldots, s_I]$ and $t = [t_1, \ldots, t_J]$ be (random variable sequences representing) the source/target sentence, with $I$ and $J$ respectively being the source/target lengths. Machine translation involves finding the most probable target sentence given the source:

$$\arg\max_{t \in \mathcal{T}} p(t \mid s)$$

where $\mathcal{T}$ is the set of all possible sequences. NMT models parameterize $p(t \mid s)$ with an encoder neural network which reads the source sentence and a decoder neural network which produces a distribution over the target sentence (one word at a time) given the source. We employ the attentional architecture from Luong et al. (2015), which achieved state-of-the-art results on English $\rightarrow$ German translation.

2.2 Knowledge Distillation

Knowledge distillation describes a class of methods for training a smaller student network to perform better by learning from a larger teacher network (in addition to learning from the training data set). We generally assume that the teacher has previously been trained, and that we are estimating parameters for the student. Knowledge distillation suggests training by matching the student’s predictions to the teacher’s predictions. For classification this usually means matching the probabilities either via $L_2$ on the log scale (Ba and Caruana, 2014) or by cross-entropy (Li et al., 2014; Hinton et al., 2015).

Concretely, assume we are learning a multi-class classifier over a data set of examples of the form $(x, y)$ with possible classes $\mathcal{V}$. The usual training criteria is to minimize NLL for each example from the training data,

$$\mathcal{L}_{\text{NLL}}(\theta) = - \sum_{k=1}^{|\mathcal{V}|} 1\{y = k\} \log p(y = k \mid x; \theta)$$

where $1\{\cdot\}$ is the indicator function and $p$ the distribution from our model (parameterized by $\theta$).

$^3$Specifically, we use the global-general attention model with the input-feeding approach. We refer the reader to the original paper for further details.
This objective can be seen as minimizing the cross-entropy between the degenerate data distribution (which has all of its probability mass on one class) and the model distribution \( p(y \mid x; \theta) \).

In knowledge distillation, we assume access to a learned teacher distribution \( q(y \mid x; \theta_T) \), possibly trained over the same data set. Instead of minimizing cross-entropy with the observed data, we instead minimize the cross-entropy with the teacher’s probability distribution,

\[
\mathcal{L}_{\text{KD}}(\theta; \theta_T) = - \sum_{k=1}^{|V|} q(y = k \mid x; \theta_T) \times \log p(y = k \mid x; \theta)
\]

where \( \theta_T \) parameterizes the teacher distribution and remains fixed. Note the cross-entropy setup is identical, but the target distribution is no longer a sparse distribution.\(^4\) Training on \( q(y \mid x; \theta_T) \) is attractive since it gives more information about other classes for a given data point (e.g., similarity between classes) and has less variance in gradients.\(^4\)

Since this new objective has no direct term for the training data, it is common practice to interpolate between the two losses,

\[
\mathcal{L}(\theta; \theta_T) = (1 - \alpha) \mathcal{L}_{\text{NLL}}(\theta) + \alpha \mathcal{L}_{\text{KD}}(\theta; \theta_T)
\]

where \( \alpha \) is mixture parameter combining the one-hot distribution and the teacher distribution.

3 Knowledge Distillation for NMT

The large sizes of neural machine translation systems make them an ideal candidate for knowledge distillation approaches. In this section we explore three different ways this technique can be applied to NMT.

3.1 Word-Level Knowledge Distillation

NMT systems are trained directly to minimize word NLL, \( \mathcal{L}_{\text{WORD-NLL}} \), at each position. Therefore if we have a teacher model, standard knowledge distillation for multi-class cross-entropy can be applied. We define this distillation for a sentence as,

\[
\mathcal{L}_{\text{WORD-KD}} = - \sum_{j=1}^J \sum_{k=1}^{|V|} q(t_j = k \mid s, t_{<j}) \times \log p(t_j = k \mid s, t_{<j})
\]

where \( V \) is the target vocabulary set. The student can further be trained to optimize the mixture of

Figure 1: Overview of the different knowledge distillation approaches. In word-level knowledge distillation (left) cross-entropy is minimized between the student/teacher distributions (yellow) for each word in the actual target sequence (ECD), as well as between the student distribution and the degenerate data distribution, which has all of its probability mass on one word (black). In sequence-level knowledge distillation (center) the student network is trained on the output from beam search of the teacher network that had the highest score (ACF). In sequence-level interpolation (right) the student is trained on the output from beam search of the teacher network that had the highest \( \text{sim} \) with the target sequence (ECE).
\( \mathcal{L}_{\text{WORD-KD}} \) and \( \mathcal{L}_{\text{WORD-NLL}} \). In the context of NMT, we refer to this approach as word-level knowledge distillation and illustrate this in Figure 1 (left).

### 3.2 Sequence-Level Knowledge Distillation

Word-level knowledge distillation allows transfer of these local word distributions. Ideally however, we would like the student model to mimic the teacher’s actions at the sequence-level. The sequence distribution is particularly important for NMT, because wrong predictions can propagate forward at test-time.

First, consider the sequence-level distribution specified by the model over all possible sequences \( t \in T \),

\[
p(t \mid s) = \prod_{j=1}^{J} p(t_j \mid s, t_{< j})
\]

for any length \( J \). The sequence-level negative log-likelihood for NMT then involves matching the one-hot distribution over all complete sequences,

\[
\mathcal{L}_{\text{SEQ-NLL}} = - \sum_{t \in T} \mathbb{1}\{t = y\} \log p(t \mid s)
\]

\[
= - \sum_{j=1}^{J} \sum_{k=1}^{|V|} \mathbb{1}\{y_j = k\} \log p(t_j = k \mid s, t_{< j})
\]

\[
= \mathcal{L}_{\text{WORD-NLL}}
\]

where \( y = [y_1, \ldots, y_J] \) is the observed sequence. Of course, this just shows that from a negative log likelihood perspective, minimizing word-level NLL and sequence-level NLL are equivalent in this model.

But now consider the case of sequence-level knowledge distillation. As before, we can simply replace the distribution from the data with a probability distribution derived from our teacher model. However, instead of using a single word prediction, we use \( q(t \mid s) \) to represent the teacher’s sequence distribution over the sample space of all possible sequences,

\[
\mathcal{L}_{\text{SEQ-KD}} = - \sum_{t \in T} q(t \mid s) \log p(t \mid s)
\]

Note that \( \mathcal{L}_{\text{SEQ-KD}} \) is inherently different from \( \mathcal{L}_{\text{WORD-KD}} \), as the sum is over an exponential number of terms. Despite its intractability, we posit that this sequence-level objective is worthwhile. It gives the teacher the chance to assign probabilities to complete sequences and therefore transfer a broader range of knowledge. We thus consider an approximation of this objective.

Our simplest approximation is to replace the teacher distribution \( q \) with its mode,

\[
q(t \mid s) \sim \mathbb{1}\{t = \text{argmax}_t q(t \mid s)\}
\]

Observing that finding the mode is itself intractable, we use beam search to find an approximation. The loss is then

\[
\mathcal{L}_{\text{SEQ-KD}} \approx - \sum_{t \in T} \mathbb{1}\{t = \hat{y}\} \log p(t \mid s)
\]

\[
= - \log p(t = \hat{y} \mid s)
\]

where \( \hat{y} \) is now the output from running beam search with the teacher model.

Using the mode seems like a poor approximation for the teacher distribution \( q(t \mid s) \), as we are approximating an exponentially-sized distribution with a single sample. However, previous results showing the effectiveness of beam search decoding for NMT lead us to belief that a large portion of \( q \)’s mass lies in a single output sequence. In fact, in experiments we find that with beam of size 1, \( q(\hat{y} \mid s) \) (on average) accounts for 1.3% of the distribution for German \( \rightarrow \) English, and 2.3% for Thai \( \rightarrow \) English (Table 1: \( p(t = \hat{y}) \))\(^5\).

To summarize, sequence-level knowledge distillation suggests to: (1) train a teacher model, (2) run beam search over the training set with this model, (3) train the student network with cross-entropy on this new dataset. Step (3) is identical to the word-level NLL process except now on the newly-generated data set. This is shown in Figure 1 (center).

\(^5\) Additionally there are simple ways to better approximate \( q(t \mid s) \). One way would be to consider a \( K \)-best list from beam search and renormalizing the probabilities,

\[
q(t \mid s) \sim \frac{q(t \mid s)}{\sum_{t \in \mathcal{T}_K} q(t \mid s)}
\]

where \( \mathcal{T}_K \) is the \( K \)-best list from beam search. This would increase the training set by a factor of \( K \). A beam of size 5 captures 2.8% of the distribution for German \( \rightarrow \) English, and 3.8% for Thai \( \rightarrow \) English. Another alternative is to use a Monte Carlo estimate and sample from the teacher model (since \( \mathcal{L}_{\text{SEQ-KD}} = \mathbb{E}_{t \sim q(t \mid s)}[- \log p(t \mid s)] \)). However in practice we found the (approximate) mode to work well.
3.3 Sequence-Level Interpolation

Next we consider integrating the training data back into the process, such that we train the student model as a mixture of our sequence-level teacher-generated data ($\mathcal{L}_{\text{SEQ-KD}}$) with the original training data ($\mathcal{L}_{\text{SEQ-NLL}}$),

$$
\mathcal{L} = (1 - \alpha) \mathcal{L}_{\text{SEQ-NLL}} + \alpha \mathcal{L}_{\text{SEQ-KD}}
$$

$$
= -(1 - \alpha) \log p(y \mid s) - \alpha \sum_{t \in T} q(t \mid s) \log p(t \mid s)
$$

where $y$ is the gold target sequence.

Since the second term is intractable, we could again apply the mode approximation from the previous section,

$$
\mathcal{L} = -(1 - \alpha) \log p(y \mid s) - \alpha \log p(\hat{y} \mid s)
$$

and train on both observed ($y$) and teacher-generated ($\hat{y}$) data. However, this process is non-ideal for two reasons: (1) unlike for standard knowledge distribution, it doubles the size of the training data, and (2) it requires training on both the teacher-generated sequence and the true sequence, conditioned on the same source input. The latter concern is particularly problematic since we observe that $y$ and $\hat{y}$ are often quite different.

As an alternative, we propose a single-sequence approximation that is more attractive in this setting. This approach is inspired by local updating (Liang et al., 2006), a method for discriminative training in statistical machine translation (although to our knowledge not for knowledge distillation). Local updating suggests selecting a training sequence which is close to $y$ and has high probability under the teacher model,

$$
\hat{y} = \arg\max_{t \in T} sim(t, y)q(t \mid s)
$$

where $sim$ is a function measuring closeness (e.g. Jaccard similarity or BLEU (Papineni et al., 2002)). Following local updating, we can approximate this sequence by running beam search and choosing

$$
\hat{y} \approx \arg\max_{t \in \hat{\mathcal{T}}_K} sim(t, y)
$$

where $\hat{\mathcal{T}}_K$ is the $K$-best list from beam search. We take $sim$ to be smoothed sentence-level BLEU (Chen and Cherry, 2014).

We justify training on $\hat{y}$ from a knowledge distillation perspective with the following generative process: suppose that there is a true target sequence (which we do not observe) that is first generated from the underlying data distribution $\mathcal{D}$. And further suppose that the target sequence that we observe ($y$) is a noisy version of the unobserved true sequence: i.e. (i) $t \sim \mathcal{D}$, (ii) $y \sim \epsilon(t)$, where $\epsilon(t)$ is, for example, a noise function that independently replaces each element in $t$ with a random element in $\mathcal{V}$ with some small probability. In such a case, ideally the student’s distribution should match the mixture distribution,

$$
\mathcal{D}_{\text{SEQ-Inter}} \sim (1 - \alpha)\mathcal{D} + \alpha q(t \mid s)
$$

In this setting, due to the noise assumption, $\mathcal{D}$ now has significant probability mass around a neighborhood of $y$ (not just at $y$), and therefore the argmax of the mixture distribution is likely something other than $y$ (the observed sequence) or $\hat{y}$ (the output from beam search). We can see that $\hat{y}$ is a natural approximation to the argmax of this mixture distribution between $\mathcal{D}$ and $q(t \mid s)$ for some $\alpha$. We illustrate this framework in Figure 1 (right) and visualize the distribution over a real example in Figure 2.

4 Experimental Setup

To test out these approaches, we conduct two sets of NMT experiments: high resource (English $\rightarrow$ German) and low resource (Thai $\rightarrow$ English).

The English-German data comes from WMT 2014. The training set has 4m sentences and we take newstest2012/newstest2013 as the dev set and newstest2014 as the test set. We keep the top 50k most frequent words, and replace the rest with UNK. The teacher model is a $4 \times 1000$ LSTM (as in Luong et al. (2015)) and we train two student models: $2 \times 300$ and $2 \times 500$. The Thai-English data comes from IWSLT 2015. There are 90k sentences in the

Footnotes:

---

6While we employ a simple (unrealistic) noise function for illustrative purposes, the generative story is quite plausible if we consider a more elaborate noise function which includes additional sources of noise such as phrase reordering, replacement of words with synonyms, etc. One could view translation having two sources of variance that should be modeled separately: variance due to the source sentence ($t \sim \mathcal{D}$), and variance due to the individual translator ($y \sim \epsilon(t)$).

7http://statmt.org/wmt14

8https://sites.google.com/site/iwsltevaluation2015/mt-track
Figure 2: Visualization of sequence-level interpolation on an example German → English sentence: Bis 15 Tage vor Anreise sind Zimmer-Annullationen kostenlos. We run beam search, plot the final hidden state of the hypotheses using t-SNE and show the corresponding (smoothed) probabilities with contours. In the above example, the sentence that is at the top of the beam after beam search (green) is quite far away from gold (red), so we train the model on a sentence that is on the beam but had the highest sim (e.g. BLEU) to gold (purple).

training set and we take 2010/2011/2012 data as the dev set and 2012/2013 as the test set, with a vocabulary size is 25k. Size of the teacher model is 2 × 500 (which performed better than 4 × 1000, 2 × 750 models), and the student model is 2 × 100. Other training details mirror Luong et al. (2015).

We evaluate on tokenized BLEU with multi-bleu.perl, and experiment with the following variations:

**Word-Level Knowledge Distillation (Word-KD)**
Student is trained on the original data and additionally trained to minimize the cross-entropy of the teacher distribution at the word-level. We tested $\alpha \in \{0.5, 0.9\}$ and found $\alpha = 0.5$ to work better.

**Sequence-Level Knowledge Distillation (Seq-KD)**
Student is trained on the teacher-generated data, which is the result of running beam search and taking the highest-scoring sequence with the teacher model. We use beam size $K = 5$ (we did not see improvements with a larger beam).

**Sequence-Level Interpolation (Seq-Inter)**
Student is trained on the sequence on the teacher’s beam that had the highest BLEU (beam size $K = 35$). We adopt a fine-tuning approach where we begin training from a pretrained model (either on original data or Seq-KD data) and train with a smaller learning rate (0.1). For English-German we generate Seq-Inter data on a smaller portion of the training set ($\sim 50\%$) for efficiency.

The above methods are complementary and can be combined with each other. For example, we can train on teacher-generated data but still include a word-level cross-entropy term between the teacher/student (Seq-KD + Word-KD in Table 1), or fine-tune towards Seq-Inter data starting from the baseline model trained on original data (Baseline + Seq-Inter in Table 1).

5 Results and Discussion

Results of our experiments are shown in Table 1. We find that while word-level knowledge distillation (Word-KD) does improve upon the baseline, sequence-level knowledge distillation (Seq-KD) does better on English → German and performs similarly on Thai → English. Combining them (Seq-KD + Word-KD) results in further gains for the $2 \times 300$ and $2 \times 100$ models (although not for the $2 \times 500$ model), indicating that these methods provide orthogonal means of transferring knowledge from the teacher to the student: Word-KD is transferring knowledge at the the local (i.e. word) level while Seq-KD is transferring knowledge at the global (i.e. sequence) level.

Sequence-level interpolation (Seq-Inter), in addition to improving models trained via Word-KD and Seq-KD, also improves upon the original teacher model that was trained on the actual data but fine-tuned towards Seq-Inter data (Baseline + Seq-Inter). In fact, greedy decoding with this fine-tuned model has similar performance (19.6) as beam search with the original model (19.5), allowing for faster decoding even with an identically-sized model.

We hypothesize that sequence-level knowledge distillation is effective because it allows the student network to only model relevant parts of the teacher distribution (i.e. around the teacher’s mode) instead of ‘wasting’ parameters on trying to model the entire

---

9For instance, ‘Seq-KD + Seq-Inter + Word-KD’ in Table 1 means that the model was trained on Seq-KD data and fine-tuned towards Seq-Inter data with the mixture cross-entropy loss at the word-level.
space of translations. Our results suggest that this is indeed the case: the probability mass that Seq-KD models assign to the approximate mode is much higher than is the case for baseline models trained on original data (Table 1: $p(t = \hat{y})$). For example, on English → German the (approximate) argmax for the $2 \times 500$ Seq-KD model (on average) accounts for 16.9% of the total probability mass, while the corresponding number is 0.9% for the baseline.

This also explains the success of greedy decoding for Seq-KD models—since we are only modeling around the teacher’s mode, the student’s distribution is more peaked and therefore the argmax is much easier to find. Seq-Inter offers a compromise between the two, with the greedily-decoded sequence accounting for 7.6% of the distribution.

Finally, although past work has shown that models with lower perplexity generally tend to have

<table>
<thead>
<tr>
<th>Model</th>
<th>$\text{BLEU}_{K=1}$</th>
<th>$\Delta_{K=1}$</th>
<th>$\text{BLEU}_{K=5}$</th>
<th>$\Delta_{K=5}$</th>
<th>PPL</th>
<th>$p(t = \hat{y})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher Baseline $4 \times 1000$ (Params: 221m)</td>
<td>17.7</td>
<td>19.5</td>
<td>6.7</td>
<td>1.3%</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Baseline + Seq-Inter</td>
<td>19.6</td>
<td>19.8</td>
<td>10.4</td>
<td>8.2%</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Student Baseline $2 \times 500$ (Params: 84m)</td>
<td>14.7</td>
<td>17.6</td>
<td>8.2</td>
<td>0.9%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Word-KD</td>
<td>15.4</td>
<td>17.7</td>
<td>8.0</td>
<td>1.0%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD</td>
<td>18.9</td>
<td>19.0</td>
<td>22.7</td>
<td>16.9%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Baseline + Seq-Inter</td>
<td>18.5</td>
<td>18.7</td>
<td>11.3</td>
<td>5.7%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Word-KD + Seq-Inter</td>
<td>18.3</td>
<td>18.5</td>
<td>11.8</td>
<td>6.3%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD + Seq-Inter</td>
<td>18.9</td>
<td>19.3</td>
<td>15.8</td>
<td>7.6%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD + Word-KD</td>
<td>18.7</td>
<td>18.9</td>
<td>10.9</td>
<td>4.1%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD + Seq-Inter + Word-KD</td>
<td>18.8</td>
<td>19.2</td>
<td>14.8</td>
<td>7.1%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Student Baseline $2 \times 300$ (Params: 49m)</td>
<td>14.1</td>
<td>16.9</td>
<td>10.3</td>
<td>0.6%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Word-KD</td>
<td>14.9</td>
<td>18.1</td>
<td>10.9</td>
<td>0.7%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD</td>
<td>18.1</td>
<td>18.1</td>
<td>64.4</td>
<td>14.8%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Baseline + Seq-Inter</td>
<td>17.6</td>
<td>17.9</td>
<td>13.0</td>
<td>10.0%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Word-KD + Seq-Inter</td>
<td>17.8</td>
<td>18.0</td>
<td>14.5</td>
<td>4.3%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD + Seq-Inter</td>
<td>18.2</td>
<td>18.5</td>
<td>40.8</td>
<td>5.6%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD + Word-KD</td>
<td>17.9</td>
<td>18.8</td>
<td>44.1</td>
<td>3.1%</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Seq-KD + Seq-Inter + Word-KD</td>
<td>18.5</td>
<td>18.9</td>
<td>97.1</td>
<td>5.9%</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Results on English-German (newstest2014) and Thai-English (2012/2013) test sets. $\text{BLEU}_{K=1}$: BLEU score with beam size $K = 1$ (i.e. greedy decoding); $\Delta_{K=1}$: BLEU gain over the baseline model without any knowledge distillation with greedy decoding; $\text{BLEU}_{K=5}$: BLEU score with beam size $K = 5$; $\Delta_{K=5}$: BLEU gain over the baseline model without any knowledge distillation with beam size $K = 5$; PPL: perplexity on the test set; $p(t = \hat{y})$: Probability of output sequence from greedy decoding (averaged over the test set). Params: number of parameters in the model. Best results (as measured by improvement over the baseline) within each category are highlighted in bold.
higher BLEU, our results indicate that this is not necessarily the case. The perplexity of the baseline 2 × 500 English → German model is 8.2 while the perplexity of the corresponding Seq-KD model is 22.7, despite the fact that Seq-KD model does significantly better for both greedy (+4.2 BLEU) and beam search (+1.4 BLEU) decoding.

5.1 Decoding Speed
Run-time complexity for beam search grows linearly with beam size. Therefore, the fact that sequence-level knowledge distillation allows for greedy decoding is significant, with practical implications for running NMT systems across various devices. To test the speed gains, we run the teacher/student models on GPU, CPU, and smartphone, and check the average number of source words translated per second (Table 2). We use a GeForce GTX Titan X for GPU and a Samsung Galaxy 6 smartphone. We find that we can run the student model 10 times faster with greedy decoding than the teacher model with beam search on GPU (1051.3 vs 101.9 words/sec), with similar performance.

5.2 Weight Pruning
Although knowledge distillation enables training faster models, the number of parameters for the student models is still somewhat large (Table 1: Params), due to the word embeddings which dominate most of the parameters.\(^\text{10}\) For example, on the 2 × 500 English → German model the word embeddings account for approximately 63% (50m out of 84m) of the parameters. The size of word embeddings have little impact on run-time as the word embedding layer is a simple lookup table that only affects the first layer of the model.

We therefore focus next on reducing the memory footprint of the student models further through weight pruning. Weight pruning for NMT was recently investigated by See et al. (2016), who found that up to 80 – 90% of the parameters in a large NMT model can be pruned with little loss in performance. We take our best English → German student model (2 × 500 Seq-KD + Seq-Inter) and prune \(x\)% of the parameters by removing the weights with the lowest absolute values. We then retrain the pruned model on Seq-KD data with a learning rate of 0.2 and fine-tune towards Seq-Inter data with a learning rate of 0.1. As observed by See et al. (2016), retraining proved to be crucial. The results are shown in Table 3.

Our findings suggest that compression benefits achieved through weight pruning and knowledge distillation are orthogonal.\(^\text{11}\) Pruning 80% of the weight in the 2 × 500 student model results in a model with 13× fewer parameters than the original teacher model with only a decrease of 0.4 BLEU. While pruning 90% of the weights results in a more appreciable decrease of 1.0 BLEU, the model is

<table>
<thead>
<tr>
<th>Model Size</th>
<th>GPU</th>
<th>CPU</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Beam = 1 (Greedy)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 × 1000</td>
<td>425.5</td>
<td>15.0</td>
<td>–</td>
</tr>
<tr>
<td>2 × 500</td>
<td>1051.3</td>
<td>63.6</td>
<td>8.8</td>
</tr>
<tr>
<td>2 × 300</td>
<td>1267.8</td>
<td>104.3</td>
<td>15.8</td>
</tr>
<tr>
<td><strong>Beam = 5</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 × 1000</td>
<td>101.9</td>
<td>7.9</td>
<td>–</td>
</tr>
<tr>
<td>2 × 500</td>
<td>181.9</td>
<td>22.1</td>
<td>1.9</td>
</tr>
<tr>
<td>2 × 300</td>
<td>189.1</td>
<td>38.4</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Table 2: Number of source words translated per second across GPU (GeForce GTX Titan X), CPU, and smartphone (Samsung Galaxy 6) for the various English → German models. We were unable to open the 4 × 1000 model on the smartphone.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prune %</th>
<th>Params</th>
<th>BLEU</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 × 1000</td>
<td>0%</td>
<td>221 m</td>
<td>19.5</td>
<td>1×</td>
</tr>
<tr>
<td>2 × 500</td>
<td>0%</td>
<td>84 m</td>
<td>19.3</td>
<td>3×</td>
</tr>
<tr>
<td>2 × 500</td>
<td>50%</td>
<td>42 m</td>
<td>19.3</td>
<td>5×</td>
</tr>
<tr>
<td>2 × 500</td>
<td>80%</td>
<td>17 m</td>
<td>19.1</td>
<td>13×</td>
</tr>
<tr>
<td>2 × 500</td>
<td>85%</td>
<td>13 m</td>
<td>18.8</td>
<td>18×</td>
</tr>
<tr>
<td>2 × 500</td>
<td>90%</td>
<td>8 m</td>
<td>18.5</td>
<td>26×</td>
</tr>
</tbody>
</table>

Table 3: Performance of student models with varying % of the weights pruned. Top two rows are models without any pruning. Params: number of parameters in the model; Prune %: Percentage of weights pruned based on their absolute values; BLEU: BLEU score with beam search decoding (\(K = 5\)) after retraining the pruned model; Ratio: Ratio of the number of parameters versus the original teacher model (which has 221m parameters).

\(^\text{10}\)Word embeddings scale linearly while RNN parameters scale quadratically with the dimension size.

\(^\text{11}\)To our knowledge combining pruning and knowledge distillation has not been investigated before.
drastically smaller with 8m parameters, which is 26× fewer than the original teacher model.

5.3 Further Observations

- For models trained with word-level knowledge distillation, we also tried regressing the student network’s top-most hidden layer at each time step to the teacher network’s top-most hidden layer as a pretraining step, noting that [Romero et al. (2015)] obtained improvements with a similar technique on feed-forward models. We found this to give comparable results to standard knowledge distillation and hence did not pursue this further.

- There have been promising recent results on eliminating word embeddings completely and obtaining word representations directly from characters with character composition models, which have many fewer parameters than word embedding lookup tables [Ling et al., 2015a; Kim et al., 2016; Ling et al., 2015b; Jozefowicz et al., 2016; Costa-Jussa and Fonollosa, 2016]. Combining such methods with knowledge distillation/pruning to further reduce the memory footprint of NMT systems remains an avenue for future work.

6 Related Work

Compressing deep learning models is an active area of current research. Pruning methods involve pruning weights or entire neurons/nodes based on some criterion. LeCun et al. (1990) prune weights based on an approximation of the Hessian, while Han et al. (2016) show that a simple magnitude-based pruning works well. Prior work on removing neurons/nodes include Srinivas and Babu (2015) and Mariet and Sra (2016). See et al. (2016) were the first to apply pruning to Neural Machine Translation, observing that that different parts of the architecture (input word embeddings, LSTM matrices, etc.) admit different levels of pruning. Knowledge distillation approaches train a smaller student model to mimic a larger teacher model, by minimizing the loss between the teacher/student predictions (Bucila et al., 2006; Ba and Caruana, 2014; Li et al., 2014; Hinton et al., 2015). Romero et al. (2015) additionally regress on the intermediate hidden layers of the student/teacher network as a pretraining step, while Mou et al. (2015) obtain smaller word embeddings from a teacher model via regression. There has also been work on transferring knowledge across different network architectures: Chan et al. (2015b) show that a deep non-recurrent neural network can learn from an RNN; Geras et al. (2016) train a CNN to mimic an LSTM for speech recognition. Kuncoro et al. (2016) recently investigated knowledge distillation for structured prediction by having a single parser learn from an ensemble of parsers.

Other approaches for compression involve low rank factorizations of weight matrices (Denton et al., 2014; Jaderberg et al., 2014; Lu et al., 2016; Prabhavalkar et al., 2016), sparsity-inducing regularizers (Murray and Chiang, 2015), binarization of weights (Courbariaux et al., 2016; Lin et al., 2016), and weight sharing (Chen et al., 2015; Han et al., 2016). Finally, although we have motivated sequence-level knowledge distillation in the context of training a smaller model, there are other techniques that train on a mixture of the model’s predictions and the data, such as local updating (Liang et al., 2006), hope/fear training (Chiang, 2012), SEARN (Daumé III et al., 2009); DAgger (Ross et al., 2011), and minimum risk training (Och, 2003; Shen et al., 2016).

7 Conclusion

In this work we have investigated existing knowledge distillation methods for NMT (which work at the word-level) and introduced two sequence-level variants of knowledge distillation, which provide improvements over standard word-level knowledge distillation.

We have chosen to focus on translation as this domain has generally required the largest capacity deep learning models, but the sequence-to-sequence framework has been successfully applied to a wide range of tasks including parsing (Vinyals et al., 2015a), summarization (Rush et al., 2015), dialogue (Vinyals and Le, 2015) Serban et al., 2016; Li et al., 2016), NER/POS-tagging (Gillick et al., 2016), image captioning (Vinyals et al., 2015b; Xu et al., 2015), video generation (Srivastava et al., 2015), and speech recognition (Chan et al., 2015a). We anticipate that methods described in this paper can be used to similarly train smaller models in other domains.
References


