LARGE-SCALE PRETRAINING FOR NEURAL MACHINE TRANSLATION WITH TENS OF BILLIONS OF SENTENCE PAIRS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we investigate the problem of training neural machine translation (NMT) systems with a dataset of more than 40 billion bilingual sentence pairs, which is larger than the largest dataset to date by orders of magnitude. Unprecedented challenges emerge in this situation compared to previous NMT work, including severe noise in the data and prohibitively long training time. We propose practical solutions to handle these issues and demonstrate that large-scale pretraining significantly improves NMT performance. We are able to push the BLEU score of WMT17 Chinese-English dataset to 32.3, with a significant performance boost of +3.2 over existing state-of-the-art results.

1 INTRODUCTION

End-to-end neural machine translation (NMT) (Bahdanau et al., 2014; Sutskever et al., 2014; Sennrich et al., 2015b; Vaswani et al., 2017; Britz et al., 2017; Gehring et al., 2017; Klein et al., 2017; Johnson et al., 2017; Wang et al., 2017; Hassan et al., 2018; Aharoni et al., 2019; Ng et al., 2019) has been widely adopted as the state-of-the-art approach for MT. Particularly, sequence-to-sequence models (SEQ2SEQ for short) are trained and tested on publicly available benchmark datasets, the size of which ranges from tens of thousands for low-resource languages to hundreds of millions for widely used languages.

Recent progress in natural language understanding (NLU) has proved that large-scale pretraining such as BERT (Devlin et al., 2018) or Elmo (Peters et al., 2018) often lead to a significant leap forward in SOTA results. Mostly due to the lack of paired training data (Song et al., 2019), no comparable success has been made in the field of MT. Therefore, two important questions remained unanswered: whether we can push the performance of existing neural models with more data, e.g., if we use tens of billions of bilingual training sentence pairs. And if so, how we can address the unique challenges introduced by such gigantic dataset. The answers to these questions are not immediately clear as several issues are unprecedented in this case:

- Scale: Firstly, an NMT model’s expressivity is limited by infrastructures such as GPU memory, so indefinitely increasing the size of the training data might not improve the performance. Secondly, training on a massively large dataset with tens of billion sentence pairs can be prohibitively slow.

- Noise and Out-of-domain data: A dataset with tens of billions of bilingual sentences pairs must span a wide range of domains and comes from extremely noisy sources. It is widely accepted that translation quality is very vulnerable to out-of-domain data and noisy data (Chen et al., 2016; Niehues & Waibel, 2010; Koehn & Schroeder, 2007; Eidelman et al., 2012), and a small number of noisy training instances can have negative effects on translation quality (Belinkov & Bisk, 2017). This means blindly increasing the size of training data or adding noisy training data may not necessarily lead to a better performance, but may even backfire.

In this paper, we investigate the problem of training neural machine translation systems on a dataset with more than 40 billion bilingual sentence pairs, which is by orders of magnitude larger than the largest dataset to date. To tailor existing WMT systems to the massive training dataset, we propose a
pipelined strategy which involves large-scale pretraining and domain-specific fine-tuning. We handle the trade-off between full-dataset optimality and convergence speed for pretraining and demonstrate that having large-scale pretraining significantly improves NMT performance. Combined with other existing NMT techniques (Hassan et al., 2018; Ranzato et al., 2015; Li et al., 2016), combining with other techniques such as RL and agreement reranking, the proposed model is able to reach a BLEU score of 32.3 for WMT 2017 Chinese-English translation, with a significant performance boost of +3.2 over existing SOTA results.

2 RELATED WORK

Large scale pretraining has proved to be of significant importance in NLP, from word vectors such as word2vec/Glove in the early days (Pennington et al., 2014; Mikolov et al., 2013) to recent language model pretraining such as BERT (Devlin et al., 2018) and Elmo (Peters et al., 2018). Pretraining has not achieved comparable success in MT, mostly due to the difficulty of getting a large parallel corpus. Most NMT pretraining work focuses on unsupervised NMT (Lample & Conneau, 2019) or pretraining using monolingual data (Song et al., 2019). Other relevant works for SEQ2SEQ pretraining include using an auto-encoder to pretrain the encoder-decoder network (Dai & Le, 2015; Ramachandran et al., 2016) and transfer learning from rich-source languages to low-source languages (Zoph et al., 2016; Firat et al., 2016).

Our work is related to a couple of previous work in domain adaptation for MT. In the context of phrase-based MT, [Hildebrand et al., 2005] select sentences similar to the topic of the test set to construct a new training corpus, which avoids topic discrepancies between the training and the test datasets. [Xiao et al., 2012] estimate word translation probabilities conditioned on topics, and then adapt lexical weights of phrases by these topic-specific probabilities. In the context of NMT, [Chen et al., 2016] provide neural networks with topic information (which are human-labeled product categories) on the decoder side. [Zhang et al., 2016b] first run topic models (Blei et al., 2003) on the training data for both sources and targets, and add topic representations to the encoder-decoder model.

Mixture models have been widely used in MT. In the context of phrase-based MT, [Foster & Kuhn, 2007] propose a three-step pipelined strategy: they first split the training corpus into different sub-corpora according to some predefined criterion, then train different MT models on different sub-corpora, and in the end combine different models for translation. Foster & Kuhn (2007)'s work was later extended to address various issues [Niehues & Waibel, 2010; Koehn & Schroeder, 2007; Eidelman et al., 2012], such as how to split the training corpus (Axelrod et al., 2011) and how to combine different models (Civera & Juan, 2007; Sennrich, 2012; Foster et al., 2010). In NMT, mixture models [Shen et al., 2019; He et al., 2018] are inspired by deep latent variable generation models [Kingma & Welling, 2013; Kim et al., 2018; Bowman et al., 2015]. Zhang et al. (2016a) augment NMT systems with a single Gaussian latent variable, and this work was further extended by Schulz et al. (2018) in which each target word is associated with a latent Gaussian variable. He et al. (2018) propose to use a soft mixture model to generate diverse translations. In addition, Shen et al. (2019) comprehensively evaluate different design choices of mixture models such as parameterization and prior distribution.

Due to the gigantic size of the dataset, we have to split it into subsets during training. Therefore, our paper is also relevant to a wide range of previous work on distributed training for deep neural networks (Dean et al., 2012; Yadan et al., 2013; Li et al., 2014; Krizhevsky, 2014; Das et al., 2016; Smith et al., 2017).

3 DATA SETUP

The most commonly used Chinese-to-English (Zh-En) translation dataset is WMT’17. The dataset consists of 332K sentence pairs from the News Commentary corpus, 15.8M sentence pairs from the UN Parallel Corpus, and 9M sentence pairs from the CWMT Corpus. We followed the pre-processing criterion in Hassan et al. (2018), resulting in about 20M training pairs. Newsdev2017 is used as the development set and Newstest2017 as the test set.

In addition to WMT2017, we collected a Chinese-English parallel dataset that significantly extends those used by previous work. The raw dataset we collected consists of roughly 50 billion sentence
pairs in total. The data comes from diverse sources such as web pages (~2 billion), digitized books (~1 billion) and private purchase from translation agencies (~46 billion).

The dataset is extremely noisy in two ways: (1) a significant proportion of files are in PDF format. (2) text is aligned at the document level rather than the sentence level. The size of large documents can be up to thousands of pages, with figures, tables and charts annoyingly inserted. For the first part, we developed a PDF document parsing system to decode PDF files. We are not diving into the details here since this part are out of the scope of the current paper. Succinctly, a lexical analyzer is first used to decompose PDF files into the basic lexical tokens according to the PostScript syntax. Next, we build a parser to analyze the abstract syntax tree (AST), and then decode the data into figures, tables, and text. For the second part, our goal is to transform doc-level alignment to sentence-level alignment. We use a hierarchical pipeline which consists of two stages: (1) aligning paragraphs and (2) aligning sentences within each aligned paragraph. We adopt many of the techniques in Uszkoreit et al. (2010): Paragraphs/sentences are discarded if both sides are identical or a language detector find them to be in the wrong language. We use a standard dynamic programming approach to implement the sentence/paragraph alignment algorithm, which takes sentence length and translation probability as features. Pairs are discarded if pairing scores are less than a certain threshold. We encourage the readers to refer to Uszkoreit et al. (2010) for details. After post-processing, we are left with 41 billion sentence pairs. We randomly select 1M instances as the development set.

4 MODELS AND ARCHITECTURES

4.1 PIPELINES: PRETRAINING AND FINE-TUNING

Translation quality is very vulnerable to out-of-domain data and noisy data (Chen et al., 2016; Niehues & Waibel, 2010; Koehn & Schroeder, 2007; Eidelman et al., 2012). Since dataset we use comes from a different domain from WMT2017, it would be less favorable if we directly apply a trained model on this large but noisy dataset to the WMT17 test set. One option to handle this issue is to do data selection and filtering before training (Belinkov & Bisk, 2017; Hassan et al., 2018). Hassan et al. (2018) proposes to first learn sentence representations from the provided training data in WMT2017 (target domain), and then reject training instances if their similarity with the target sentences is below a prespecified threshold. We did not choose this method for two reasons: (1) Hassan et al. (2018) select different training instances for different target domains. This means every time we encounter a new domain, we have to retrain the model; (2) the value of data filtering threshold is crucial but hard to decide: it is unrealistic to tune its value since each threshold value corresponds to a different filtered training set, on which a brand new model has to be trained.

Inspired by large-scale pretraining strategies such as BERT (Devlin et al., 2018) and Elmo (Peters et al., 2018), we used a pipelined approach: we first pretrain an NMT model on the massive dataset, and then fine-tune the model on the training set of the target domain. This strategy naturally addresses the aforementioned two caveats of data pre-filtering approach: the pretrained model can be easily adapted to a set of training data of an arbitrary domain, and it is no longer needed to find the optimal data selection threshold. Moreover, since the model will be fine-tuned at later stage, it is more immune to noise in the data at the first stage. We combine the WMT 20M data with our new 40B data to do the pretraining, and then fine-tune the model on the WMT 20M data.

4.2 TRADEOFF BETWEEN FULL-DATASET OPTIMALITY AND CONVERGENCE SPEED

There is a tradeoff between the optimality on the full dataset and the convergence speed. At one end of the spectrum, running a single model on the full dataset may achieve full-dataset optimality. However, this strategy suffers from prohibitively long training time. At the other end of the spectrum, splitting the full dataset into smaller subsets and running independent models on different datasets until convergence as in Foster & Kuhn (2007) saves a lot of training time. But each individual model only finds the optimal solution for its own subset dataset, so their combination is not guaranteed to be the optimal model for the full dataset. We harvest the best of both worlds by model communication and data communication (see later sections for details).

With a single V100 GPU, a single update on a single batch with about 2-5K tokens takes about 2-6 seconds. Even with 512 parallel GPUs, it takes months for an epoch to finish.
4.3 Model Details

We use the Transformer architecture (Vaswani et al., 2017) as a backbone. The encoder and the decoder both have 6 blocks. The number of attention heads, embedding dimension and inner-layer dimension are set to 16, 1,024 and 2,048. We use the same transformer structure for all experiments. All experiments were run using 512 Nvidia V100 GPUs with mini-batches of approximately 1M tokens. Models are optimized with Adam (Kingma & Ba, 2014) in which $\beta_1$ is set to 0.9, $\beta_2$ is set to 0.98 and $\epsilon$ is set to $10^{-8}$. For the Chinese language, instead of using segmented words or byte-pair encoding (BPE) (Sennrich et al., 2015b), we use characters as the basic units and maintain a character vocabulary size of 10,000. We will get back to this in the ablation study section. On the English side, we segment text into subword symbols using BPE (Sennrich et al., 2015b) and maintain a vocabulary size of 40K.

4.3.1 Pretraining

We explore the following different strategies for model pretraining.

**Single-Model** We use a single transformer model to fit all the training data. We use 512 Nvidia V100 GPUs with mini-batches of approximately 1M tokens. Models are saved every 0.1 epoch. Upon the submission of this paper, training has lasted for three months, 2 epochs in total, and perplexity on the development set is still dropping.

**Uniform-data-split** The disadvantages of single-model are obvious: (1) It is prohibitively slow and (2) it is unclear whether a single-model is powerful enough to model the full training dataset. We thus followed Foster & Kuhn (2007) to build mixture models, in which we first split the full dataset into a few subsets, and then train independent model components on different subsets. Here “component” refers to the individual transformer in the mixture model setup. Using multiple components naturally increase the model’s capacity and expressivity.

At test time, we use a single transformer model to fit all the training data. We use 512 Nvidia V100 GPUs with mini-batches of approximately 1M tokens. Models are saved every 0.1 epoch. Upon the submission of this paper, training has lasted for three months, 2 epochs in total, and perplexity on the development set is still dropping.

We randomly split the 40B training set into $K = 10$ subsets, denoted by $D_1, D_2, \ldots, D_K$. At training, different transformers are independently trained on different subsets using parallel GPUs until convergence. At test time, the probability of $p(y|x)$ can be written as follows:

$$p(y|x) = \sum_z p(y|z, x)p(z|x)$$

where $z$ can be simply thought as the index of different subsets. $p(y|z, x)$ is characterized by the $\text{SEQ2SEQ}$ component trained on $D_z$. $z \in [1, K]$. We assume that $p(z|x) = 1/K$ is uniform. The generation of target $y$ is thus the ensemble of the $K$ models.

**Topic-data-split** The issue with uniform-data-split is that subsets are randomly generated. It would be more desirable if each $D_z$ represents a specific domain and model components trained on different $D_z$ are separate and have their own specialization and domain expertise. Domain-specific $D_z$ comes with the advantages of having fewer vocabularies, fewer low-frequency words and more sentences with similar topics and language expression patterns. We thus propose to split the full dataset into a more elegant way using topic models (Blei et al., 2003). Topic models are widely used in data split and selection in phrase-based MT (Hildebrand et al., 2005; Zhao & Xing, 2006, 2008). One tricky issue here is that each sentence pair consists of two different languages and we need to extract bilingual topics. To handle this issue, Zhao & Xing (2008) proposed a generative model, in which each source sentence is first sampled based on its topic, as in standard LDA, and then for each position in the source sentence, a target word is sampled based on a topic-specific translation lexicon. Variational EM is used for inference. We refer the readers to Zhao & Xing (2008) for details. We followed Zhao & Xing (2008) and mined the topics distribution from the bilingual corpus. Each sentence pair is assigned to the subset that is most likely (has the highest probability) to contain the pair.

When data split is done, different $\text{SEQ2SEQ}$ components are independently trained on different $D_z$. At test time, we use $p(y|x) = \sum_z p(y|z, x)p(z|x)$ for inference. $p(y|z, x)$ is characterized by the $\text{SEQ2SEQ}$ component trained on $D_z$. Unlike uniform-data-split, $p(z|x)$ is not uniform, but predicted by a $K$-class classification model using BiLSTMs. The classification model first maps $x$ to a vector representation, which is then fed to a $K$-class softmax function.
Dynamic-data-split For topic-data-split and uniform-data-split strategies, subsets are generated beforehand and then fixed during the training. We are thus unable to dynamically adjust the data during training. This means that if a training pair is assigned to a wrong subset, there is no way we can correct it and it will negatively affect the training process permanently. Inspired by mixture models for NMT (Shen et al., 2019; He et al., 2018), we propose to dynamically assign training instances to different model components, and update different components according to the examples assigned.

\[
\nabla_{\theta} \log \sum_z p(y|z,x)p(z|x) = \mathbb{E}_{p(z|x,y)} \nabla_{\theta} \log p(y,z|x)
\]

(2)

We choose to use hard-EM instead of vanilla-EM due to the concern about the training speed: for vanilla-EM, all components need to run a forward-backward step on each training instance. This is computationally intensive and we would like to avoid the slow convergence issue. Using hard-EM, models are trained by iteratively applying the following two steps:

**E-step**: For a given sentence pair \((x_i, y_i)\), estimate \(z_i\) by: \(z_i = \arg\max_z \log p(y_i|z_i, x_i)\).

**M-step**: update parameters with gradients \(\nabla_{\theta} \log p(y_i, z_i|x_i)\).

We need to extend the single-instance EM step above to batched computation. There are two issues that requires special design: (1) the notorious “richer gets richer” issue (Eigen et al., 2013; Shazeer et al., 2017; Shen et al., 2019), i.e., once one component is slightly better than the others, it will always be selected and other components will never get trained. Then the latent variable \(z\) becomes useless and the entire system degenerates to the vanilla SEQ2SEQ model that models \(p(y|x)\); (2) to accelerate training, we need all components to get updated all the time. Recall that each component have hundreds of parallel GPUs for computing. If some components do not get enough data, which is possible in dynamic-data-split, it would be a waste of computational resources.

We propose the **batched-E-step** strategy to deal with these two issues: suppose that the batch size for the mixture model component is \(B\) (the value of which is approximately 1M). Since we have \(K\) components, we feed \(K \times B\) training instances to the model for each E-step. We ensure that each component is **guaranteed** to be assigned \(B\) instances in the E-step in order to get sufficiently updated

---

**Figure 1**: Illustrations of different strategies for training.

---

For different values of \(z_i\), each \(p(y_i|z_i, x_i)\) will be updated by gradients \(\nabla_{\theta} \log p(y_i, z_i|x_i)\) weighted by the corresponding responsibility \(p(z_i|x_i)\).
Training Data | Setting | Performance |
--- | --- | --- |
20M | Transformer (Hassan et al., 2018) | 24.4 |
20M | Sogou (Wang et al., 2017) | 26.4 |
20M | Microsoft (Hassan et al., 2018) | 27.4 |
20M | Teacher Forcing (He et al., 2019) | 29.1 |
20M+100M | Microsoft (Hassan et al., 2018) | 28.4 |
20M+40B (pretrain only) | single-model (1 epoch) | 22.1 |
20M+40B (pretrain only) | single-model (2 epoch) | 24.7 |
20M+40B (pretrain only) (ensemble) | Uniform Data Split | 27.2 |
20M+40B (pretrain only) (ensemble) | Topic Data Split | 27.7 |
20M+40B (pretrain only) (ensemble) | Dynamic Data Split | 28.4 |
20M+40B (pretrain+finetune) | single-model (1 epoch) | 27.4 |
20M+40B (pretrain+finetune) | single-model (2 epoch) | 28.7 |
20M+40B (pretrain+finetune) (ensemble) | Uniform Data Split | 31.1 |
20M+40B (pretrain+finetune) (ensemble) | Topic Data Split | 31.5 |
20M+40B (pretrain+finetune) (ensemble) | Dynamic Data Split | 32.0 |

Table 1: Main results of different models and different settings on the WMT 2017 Chinese-English test set.

in the subsequent M step. The batch-level assignment $z$ are computed as follows:

$$
\max_{z_1, z_2, \ldots, z_B} \sum_{i=1}^{BK} \log p(y_i | z_i, x_i) \\
\text{s.t. } \sum!_{i=1}^{BK} \mathbb{I}(z_i = z) = B \text{ for } z = 1, 2, \ldots, K
$$

(3)

Eq. [3] is an integer linear programming (ILP) problem. ILP is NP-hard and is solved using Hill Climbing (Russell & Norvig, 2016), a heuristic method to find the optimal solution. The proposed strategy naturally simultaneously avoid the aforementioned “richer gets richer” issue and the potential waste of computational resources. It is worth noting that the proposed batched-E-step is not specific to our scenario, but a general solution to the long-standing degeneracy issue of neural mixture models for text generation (Shazeer et al., 2017; Shen et al., 2019).

4.3.2 Fine-Tuning

At the model fine-tuning stage, we maintain the structure of the original pretrained model and fine-tune the model on the 20M WMT17 dataset. The number of iterations is treated as a hyper-parameter, which is tuned on the development set of WMT17.

For single-model, we maintain the transformer structure and run additional iterations. For uniform-data-split and topic-data-split, we fine-tune each component on the WMT17 dataset. At test time, constituent components are combined for decoding. Translations from uniform-data-split and topic-data-split are generated by the ensemble of $K$ models.

For dynamic-data-split, at the fine-tuning stage we run mixture models on the WMT17 dataset with minor adjustments: we replace the hard-EM and the batched-E-step with vanilla soft-EM, in which all components get updated with each training instance. We do this because: (1) The WMT17 dataset is significantly smaller so the computational cost is no longer a concern; (2) Mixture models have already been sufficiently trained during the pretraining, so we are less concerned about the “richer gets richer” issue. Instead, it is even desirable that some components get more fine-tuning if they are more relevant to the target domain.

5 Experimental Results

Following Hassan et al. (2018); He et al. (2019), we use sacreBLEU for evaluation. Results are reported in Table 1.

https://github.com/awslabs/sockeye/tree/master/contrib/sacrebleu
5.1 Results

**Baselines** We copied baseline results from the Sogou best WMT17 system (Wang et al., 2017), the microsoft system (Hassan et al., 2018), and the current SOTA result using teaching forcing (He et al., 2019).

**Pretrain Only** We first take a look at results of the pretrain-only setting, in which we directly apply the pretrained models to the test set (it is worth noting that the pretrained training data contains the WMT17 training set). For the single-model, due to the prohibitively long training time, we are only able to finish two epochs upon the submission of this paper, which has been running for 3 months. Though the model has not fully converged, its BLEU score (24.7) is slightly higher than the performance of the same model trained on only the WMT17 dataset (24.4).

Mixture models with different dataset split strategies outperform single-model by a huge margin. This is due to the reasons that (1) The single-model has not fully converged yet; (2) Model capacities of mixture models are significantly larger (scaled by K) than the single-model; and most importantly (3) uniform-data-split is actually an ensemble of multiple models and the comparison is not completely fair. By comparing uniform-data-split, topic-data-split and dynamic-data-split, we can see that the way the full dataset is split has a significant effect on the performance. Topic-data-split significantly outperforms uniform-data-split. This is because subsets in topic-data-split contain fewer low-frequency words and more sentences with similar language expression patterns. The dynamic-data-split strategy dynamically adjusts the training data, which lead to more coherent subsets, and thus better performances.

**Pretrain+Finetune** displays similar patterns as Pretrain-Only: the single-model achieves a BLEU score of 29.7, already outperforming the current best system. This demonstrates the great power of large-scale pretraining. The gap between single-model-1-epoch, single-model-2-epoch and uniform-data-split is narrowed in the Pretrain+Finetune setting than in Pretrain-Only. The best setting, dynamic-data-split, achieves a BLEU score of 32.0.

![Figure 2: Performances of each of the K components in different data split strategies.](image)

6 Ablation Studies and Analyses

**Data Split Strategy** In previous phrase-based MT work, both how the full set is split (Axelrod et al., 2011) and how the mixture models are combined (Civera & Juan, 2007; Sennrich, 2012; Foster et al., 2010) have a significant impact on the final performance. This is confirmed in our study. The first row of Figure 2 corresponds to the BLEU scores achieved by each of the K components with the

---

A completely fair comparison would be to use an ensemble of 20 single-model, each of which is trained on the 40B dataset. But this is very computationally prohibitive for us.
three data-split strategies (pretrain-only). As can be seen, performances for different components in uniform-data-split are very similar. This is expected since the dataset is randomly split and all subsets come from the same distribution. The Variances in topic-data-split and dynamic-data-split are much larger. This is because subsets in topic-data-split and dynamic-data-split have more centralized topic/domain distribution. Models trained on more specialized subsets perform much better than models trained on less specialized ones. The second row of Figure 2 corresponds to the average weight of components in different mixture models. For uniform-data-split, weight for each component is identical. For topic-data-split and dynamic-data-split, we can find a high correlation between weight and performance for each mixture component. This further explains the superiority of the two models.

### Model Size and Data Size

It would be interesting to compare the performances of the following cases: (1) a large model trained on the full dataset (40B) but has not fully converged, i.e., large-model-large-data, (2) a converged large model on a subset of the dataset (4 Billion), i.e., large-model-small-data and (3) a model of a smaller size, which runs much faster on the full set (40B), i.e., small-model-large-data. Table 2 represents the results. As can be seen, for the pretrain-only and the pretrain-finetune setting, small-model-large-data performs the worst. When comparing Table 2 with Table 1, small-model-large-data even underperforms a larger transformer trained only on the WMT17 dataset. This demonstrates the significant importance of model size and capacity. Interestingly, comparing large-model-large-data with large-model-small-data, the former performs a bit better on the pretrain-only setting, but performs a bit worse on the pretrain-fine-tune setting. Our explanation is as follows: for the pretrain-only setting, large-model-large-data has not fully converged, and performs a bit worse on the pretrain-fine-tune setting. In the pretrain-fine-tune setting, both models have fully converged on the WMT17 dataset. Since large-model-large-data was exposed to more data during the pretraining process, it has more generalization capability and achieves better performances during fine-tuning.

### Character, Word or BPE

In large-scale neural network learning, the necessity of Chinese word segmentation (CWS) is still under huge debate (Meng et al., 2019). The answer is not immediately clear because, on the one hand, the data sparsity issue with word-based models is less severe when massive training data is available, so word-based models might be useful. One the other hand, “word” is a human-defined linguistic concept, characterized by labeled CWS datasets (Xia 2000, Yu et al. 2001). It is widely accepted that large-scale text learning can automatically learn and encode linguistic structures (Hinoshita et al. 2011, Williams et al. 2018), which means CWS might be less useful with large training data. Using the 4B subset on which the best performance is achieved in the topic-data-split setting, we examine the performances of the word-based model (vocabulary set to 50K), the subword BPE model (vocabulary set to 50K) and the char-based model (vocabulary set to 10K). The three models only differ in the encoding stage. Results are shown in Table 3. Combined with fine-tuning, the three models achieve BLEU scores of 29.4, 29.8 and 30.1, respectively. This confirms the finding in Meng et al. (2019) that CWS is not necessary in NMT when large-scale training data is used.

### Examination of Existing NMT Techniques

It would be interesting to see which existing widely-used NMT techniques are still effective. Since our bilingual dataset is extremely larger than previous datasets, existing data augmentation strategies such as monolingual language model fusion (Sennrich et al. 2015a) or back translation (Hassan et al. 2018, Edunov et al. 2018) are expected to be no
Table 3: (a) Results for word-based, BPE-based and char-based models. (b) Results for different existing NMT techniques.

<table>
<thead>
<tr>
<th>Word, BPE and Char</th>
<th>Currently best system</th>
<th>Monolingual LM Fusion</th>
<th>Back Translation</th>
<th>Agreement Reranking</th>
<th>RL</th>
<th>Diverse Decoding</th>
<th>Agreement Reranking + RL+ Diverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>29.4</td>
<td>30.16</td>
<td>30.87</td>
<td>32.22</td>
<td>32.20</td>
<td>32.11</td>
<td>32.29</td>
</tr>
<tr>
<td>BPE</td>
<td>29.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Char</td>
<td>30.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Agreement Reranking: [Hassan et al., 2018] reranks the N-best list using models that generate sources and targets from different directions, i.e., S2T-L2R (target sentence is generated from left to right), S2T-R2L, T2S-L2R and T2SR2L. Due to the computational cost, we only retain S2T-R2L, T2S-L2R and T2SR2L on the 4B dataset and then fine-tune them on WMT2017.

Reinforce Learning [Ranzato et al., 2015; Wu et al., 2016]: refining the SEQ2SEQ objective with the RL objective to directly optimize the translation BLEU scores. We apply RL at the fine-tuning stage.

Diverse Decoding [Li et al., 2017; Vijayakumar et al., 2018]: using a diversity-promoting beam search, in which inter-sibling scores are penalized in order to generate more diverse N-best list.

Results are shown in Table 3. As can be seen, monolingual ML fusion and back-translation actually harm the performance. This is expected since the monolingual dataset is significantly smaller than the pretraining bilingual datasets. Agreement reranking introduces a +0.17 BLEU boost, confirming the importance of the order by which sequences are generated. The improvement from diverse decoding is also tiny. We think that this might because the model is already good enough for using beam search to find the global optimal. Another significant improvement comes from the RL strategy, leading to a roughly +0.2 BLEU boost. The combination of RL, agreement-ranking and diverse decoding pushes the performance up to 32.3.

7 Conclusion

In this paper, we empirically study training NMT systems with 40B training instances, the size of which is by orders of magnitude larger than the largest dataset to date. We provide practical solutions to handle the tradeoff between full-dataset level optimality and fast training speed, and demonstrate that large-scale pretraining significantly improves NMT performances. We are able to achieve a BLEU score of 32.3 on WMT17 Chinese-English dataset, with a significant performance boost of +3.2 over existing SOTA results.

References


Yuxian Meng, Xiaoya Li, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. Is word segmentation necessary for deep learning of chinese representations? arXiv preprint arXiv:1905.05526, 2019.


