Semi-Supervised Neural Machine Translation with Language Models

Ivan Skorokhodov iskorokhodov@gmail.com

Anton Rykachevskiy anton.rykachevskiy@skolkovotech.ru
Skolkovo Intsitute of Science and Technology, Moscow, Russia

Dmitry Emelyanenko dimdi-y@ya.ru
National Research University Higher School of Economics, Moscow, Russia

Sergey Slotin me@sereja.me
Moscow Institute of Physics and Technology, Dolgoprudnyy, Russia

Anton Ponkratov anton.ponkratov@skolkovotech.ru
Skolkovo Intsitute of Science and Technology, Moscow, Russia

Abstract

Training neural machine translation models is notoriously slow and requires abundant parallel corpora and computational power. In this work we propose an approach of transferring knowledge from separately trained language models to translation systems, also we investigate several techniques to improve translation quality when there is a lack of parallel data and computational resources. Our method is based on fusion between translation system and language model, and initialization of translation system with weights from pretrained language models. We show that this technique gives +2.2 BLEU score on En–Fr pair of `WMT europarl-7` dataset and allows us to reach 20.7 BLEU on 20k parallel sentences in less than 8 hours of training on a single NVIDIA GeForce GTX 1080 Ti. We specifically note, that for this advance we use nothing but monolingual corpora for source and target languages. Significant part of presented results was obtained during DeepHack.Babel hackathon on low-resource machine translation organized by iPavlov Lab.

1 Introduction

Neural Machine Translation (NMT) is now a state-of-the-art approach for building translation systems. The reason behind this is both the invention of new techniques (Bahdanau et al., 2014) and availability of massive amounts of training data. Despite of abundance of parallel datasets for some popular language pairs we still lack such learning opportunities for many others. That's why there is a growing interest in techniques which allow us to train translation systems in semi-supervised or completely unsupervised fashion.

Several notable approaches (Conneau et al., 2017; Artetxe et al., 2017; Lample et al., 2017) in this direction were provided in recent years, but supervised techniques still highly outperform unsupervised ones. In our work we investigate a semi-supervised approach of combining translation model with language models in two ways. First, we propose an approach of initializing a translation model with language models and show how it can be used to advance learning of the whole translation system. Second, we revise the technique of shallow fusion by Gülçehre et al. (2015) by using two separate gates to combine predictions of the translation model with predictions of the language model. Using both of these techniques we were able to obtain +2.2 BLEU on `WMT europarl-7` data for En–Fr pair in comparison to the strong baseline model.

Most of the current work was preformed during the DeepHack.Babel hackathon[1] on unsupervised machine translation, organized by iPavlov Lab[2]. Participants were given the task to build semi-supervised translation system. Dataset consisted of 1M monolingual sentences for each of two languages (source and target) and 50k parallel sentences. The hackathon had an unusual format: instead of Kaggle-like submissions, where participants are given both training and test data and are asked to send predictions for the test set, we had to pack our models into Docker[3] containers and send it to the submission server, which did all the training and testing on its own side. The purpose of this was to prevent participants from deviating from the hackathon rules by using additional parallel datasets or incorporating some language-dependent techniques.

As the datasets used during the hackathon are private and not available for public use, we rerun all our experiments on a dataset, specially generated from WMT'14 En–Fr data (see section 4), and all the reported results are obtained from it. In such a way, we manage to build a system which learns to translate from English to French in less than 8 hours of training on NVIDIA GeForce GTX 1080 Ti by using only 20k parallel sentences and 300k monolingual corpora. Our system obtains 20.7 BLEU score, showing improvement of 2.2 BLEU in comparison to the strong baseline model, which does not use monolingual corpora.

2 Related work

Large amounts of monolingual corpora makes it very appealing to incorporate unsupervised methods into machine translation techniques, and in recent years this trend is becoming more and more prominent.

Cheng et al. (2016) and Sennrich et al. (2015) propose an approach of *backtranslation*, which is training two translation models: source→target and target→source, and then generating synthetic training dataset from monolingual corpora to improve the models. In such a way we incorporate the dual nature of the translation problem. Authors report significant improvement up to +3.7 BLEU on English→German pair on IWSLT'14 dataset (Sennrich et al., 2015).

Gülçehre et al. (2015) show how one can improve their translation model by shallow or deep fusion of separately trained language model. Let $p(\boldsymbol{y_t} = k|\boldsymbol{x}, \boldsymbol{y}_{<t})$ be a probability that t-th word of output sequence $\boldsymbol{y}$ is the k-th word of the vocabulary under some sequence-to-sequence model. Here $\boldsymbol{x}$ is the input sentence, $\boldsymbol{y}_{<t}$ are previous $t-1$ tokens. In shallow fusion we combine probabilities from target language model and translation model in the following way:

$$\log p(\boldsymbol{y}_t = k|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \log p_{\text{trans}}(\boldsymbol{y}_t = k|\boldsymbol{x}, \boldsymbol{y}_{<t}) + \beta \log p_{\text{trg}}(\boldsymbol{y}_t = k|\boldsymbol{y}_{<t}),$$

where hyperparameter β denotes how much influence language model has for the prediction. In deep fusion authors just concatenate hidden states of the translation model and language model and fine-tune the whole thing, keeping parameters of the language model freeze.

Mikolov et al. (2013) used distributed representations of words to learn a linear mapping between vector spaces of languages and showed that this mapping can serve as a good dictionary between the languages. They pick 5k most frequent words from the source language $(\boldsymbol{x}_i)_{i=1}^{5000}$

[1]`http://babel.tilda.ws`
[2]`http://ipavlov.ai`
[3]`https://www.docker.com/`

and looked up their translations $(\boldsymbol{y}_i)_{i=1}^{5000}$ via Google Translate. Afterwards they used them to find a linear mapping W which minimizes $\sum_{i=1}^{5000} \parallel W\boldsymbol{x}_i - \boldsymbol{y}_i \parallel$. This linear mapping W was later utilized as the translation mapping to generate a dictionary between two vocabularies and proved to be rather accurate, giving almost 90% top-5 precision.

Lample et al. (2017) extended the approach of (Mikolov et al., 2013) and trained a Generative Adversarial Network (GAN) model to find this mapping without any supervised signal whatsoever. Generator was set to be this linear mapping, while discriminator should distinct between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}} = W\boldsymbol{x}$. This approach worked out: learning random bijection was impossible because of linearity and learning a bad linear mapping was impossible, because many source words would be mapped to nowhere, which is heavily penalized by discriminator. Authors report 83.3% top-1 precision, which is a significant result for purely unsupervised approach.

Artetxe et al. (2017) built upon described methods to train translation model without any parallel corpora at all. They trained a shared encoder which should encode sentences into the language-agnostic representations and then two separate decoders to reconstruct them into the desired language. To make the encoding task non-trivial authors add noise to the input sentence: they randomly swap words, forcing encoder to learn internal structure of the sentence. They also use backtranslation procedure to make model learn to translate. This approach obtained 15.56 BLEU on Fr-En pair on WMT'14 dataset.

Artetxe et al. (2017) goes further and use adversarial loss to train their translation system. They build a single shared encoder and a single shared decoder, using both denoising autoencoder loss and adversarial loss. Corrupted version of the sentence is given to the encoder and its original form is reconstructed by the decoder. Discriminator takes encoder's outputs and tries to guess which language was given to the encoder. Backtranslation is also used to teach model to translate. Such an approach shows striking performance, obtaining 32.8 BLEU on English-French pair of Multi30k-Task1 dataset.

Zoph et al. (2016) experimented with transferring different components of the translation model trained on a rich language pair (parent model) to a low-resource NMT system (child model). Such a pretraining proved to be a very strong prior for the child model parameters and improved performance by an average of 5.6 BLEU.

3 Proposed approach

Our method is built upon Transformer (Vaswani et al., 2017) architecture, which proved to be a fast and powerful machine translation model. In the original paper, Transformer was trained entirely on a parallel data. In our work we propose several improvements over it which allows us to exploit monolingual corpora and learn in semi-supervised fashion.

3.1 Transformer model

Transformer (Vaswani et al., 2017) has general encoder-decoder architecture, but unlike RNN-based models, it is purely attentional, i.e. it does not keep any internal hidden state, which is recurrently updated, and all computations are done with tokens representations. Transformer takes a sequence as an input, makes several self-attentional iterations to encode it and then several attentional iterations to decode it. Attention mechanism (scaled dot-product attention) in its general form is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d}}\Big)V,$$

where Q is some query matrix (individual queries are packed into a query matrix), K is the keys which are used to process the query, V are the values to be retrieved. Transformer uses multiple attention heads:

$$\text{MultiHead}(Q, K, V) = [\text{head}_1, ..., \text{head}_h]W^O,$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. For translation setup we usually set $Q = K = V = [z_1, ..., z_k]^\top$ where z_i is the i-th sequence element on the current iteration. Scaling factor $\sqrt{d}$ improves numerical stability and does not allow model to be pushed to regions with flat gradients.

Each time attentional layer is computed, we apply a ReLU non-linearity with residual connections.

As Transformer does not possess any recurrent state, it needs to know a position of each token some other way. For this purpose we add a positional embedding $\boldsymbol{p}_i$ to each token $\boldsymbol{z}_i$, which elements are computed as:

$$p_i[j] = \begin{cases} \sin\left(i/10000^{\frac{2j}{d}}\right), & j \text{ is even;} \\ \cos\left(i/10000^{\frac{2j}{d}}\right), & j \text{ is odd.} \end{cases}$$

3.2 Initializing transformer with language models

Originally, transformer is trained in a purely supervised fashion, which limits its use to problems with abundant parallel corpora. We mitigate this problem the following way.

We take two monolingual corpora for each language, source and target, separately train Transformer's encoder as a language model for the source language, its decoder — as a language model for the target language, and then fine-tune the whole thing on a parallel corpus to learn a translation task. In this way, our method can be seen as a transfer learning technique.

Language model (LM) is tasked to predict the next token given the sequence of previous ones. Let $p_{\text{src}}(\boldsymbol{x}_s = l|\boldsymbol{x}_{<s}; \boldsymbol{\theta}_{\text{src}}), p_{\text{trg}}(\boldsymbol{y}_t = k|\boldsymbol{y}_{<t}; \boldsymbol{\theta}_{\text{trg}})$ denote language models for source and target languages, $p_{\text{trans}}(\boldsymbol{y}_t|\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{\theta})$ denote our Transformer. Source LM is parametrized as Transformer's encoder but with additional linear output transformation to generate logits, target LM — as its decoder but without encoder's outputs attention weights. One can notice that a lot of weights between $\boldsymbol{\theta}$ and $\{\boldsymbol{\theta}_{\text{src}}, \boldsymbol{\theta}_{\text{trg}}\}$ can be shared: $\boldsymbol{\theta}$ does not need final output connection from hidden state to logits of the source LM which $\boldsymbol{\theta}_{\text{src}}$ disposes and additionally it needs attention weights for encoder outputs from the decoder which $\boldsymbol{\theta}_{\text{trg}}$ lacks. All other weights can be transferred from language models to the translation model right away. In such a way we transfer knowledge, extracted by language models from large monolingual corpora, to our translation system. Transformer architecture with proposed initialization approach is represented on figure .

3.3 Gated shallow fusion

We further improve our model by incorporating language knowledge in the process of generating output sequence. In such a way we follow a strategy of shallow fusion proposed by Gülçehre et al. (2015), but instead of adding hyperparameter β we use *gated* shallow fusion. Let $\log p_{\text{trans}}(\boldsymbol{y}_t = k|\boldsymbol{x}, \boldsymbol{y}_{<t}), \log p_{\text{trg}}(\boldsymbol{y}_t = k|\boldsymbol{y}_{<t})$ denote logits for t-th output token produced by translation transformer and LM of the target language respectively. We generate final predictions $\log p(\boldsymbol{y}_t = k|\boldsymbol{x}, \boldsymbol{y}_{<t})$ in the following manner:

$$\log p(\boldsymbol{y}_t = k|\boldsymbol{x}, \boldsymbol{y}_{<t}) = \sigma_{\text{trans}}^{(t)} \cdot \log p_{\text{trans}}(\boldsymbol{y}_t = k|\boldsymbol{x}, \boldsymbol{y}_{<t}) + \sigma_{\text{lm}}^{(t)} \cdot \log p_{\text{trg}}(\boldsymbol{y}_t = k|\boldsymbol{y}_{<t}),$$

where $\sigma_{\text{trans}}^{(t)}, \sigma_{\text{lm}}^{(t)} \in (0, 1)$ are gates which let the overall system choose between two models. Both gates are produced by a feed-forward network with one hidden layer and two output neurones:

$$[\sigma_{\text{trans}}^{(t)}, \sigma_{\text{lm}}^{(t)}]^\top = \text{FFN}(\boldsymbol{s}_{\text{trans}}^{(t)})$$

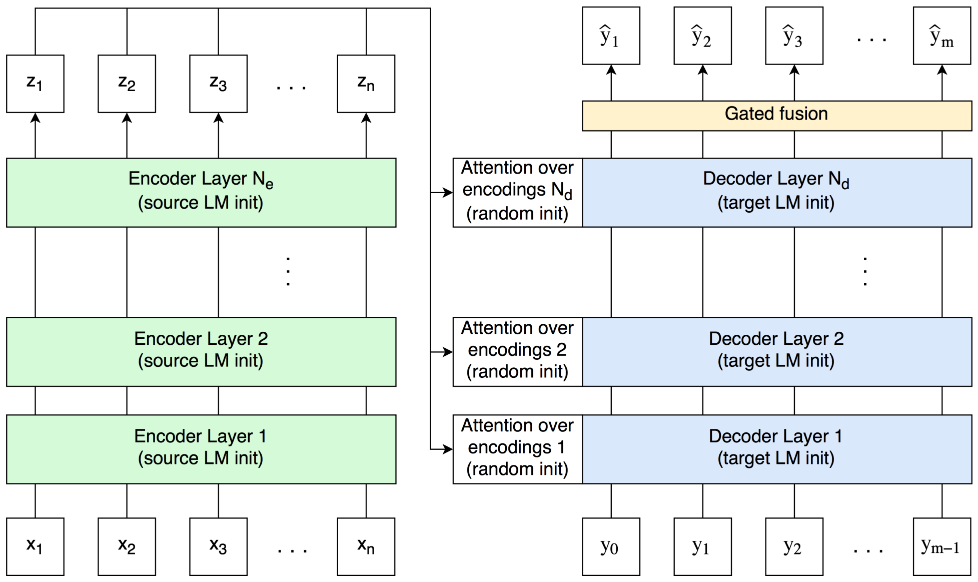
z_1
z_2
z_3
...
z_n
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
...
$\hat{y}_m$
Gated fusion
Encoder Layer N_e (source LM init)
Attention over encodings N_d (random init)
Decoder Layer N_d (target LM init)
Encoder Layer 2 (source LM init)
Attention over encodings 2 (random init)
Decoder Layer 2 (target LM init)
Encoder Layer 1 (source LM init)
Attention over encodings 1 (random init)
Decoder Layer 1 (target LM init)
x_1
x_2
x_3
...
x_n
y_0
y_1
y_2
...
y_{m-1}

Figure 1: Proposed Transformer architecture. We color components whose weights are initialized with source LM weights in green, and with target LM — in blue.

We use two separate gates, σ_{trans} and σ_{lm}, instead of using a single one and gating with σ and $(1 - \sigma)$ because it gives more accurate calibration between the language model and translation model.

During training in this setup we freeze language model weights and translation model should learn when and how much it should trust language model.

4 Setup

In this section we describe datasets we used for testing our models, preprocessing steps and model parameters. Results are presented in the next section.

4.1 Datasets

To introduce repeatable results, we reproduced all experiments on famous `WMT europarl-v7` for En–Fr pair. This corpus consists of almost 2M parallel sentences, so we were able to extract parallel data, test set, and monolingual corporas for language models from it. Important thing here is that we choose monolingual corporas from different parts of original parallel corpora, so they do not contain similar information. Also we made experiments with different sizes of parallel corpora. We name datasets according to the size of the parallel corpus and language pair, see table 1. At the same time we used private dataset from DeepHack hackathon, which consists of descriptions of hotels in English and Russian as parallel corporas and comments about hotels as monolingual ones. The size of the parallel corpora for this set was originally 50k, and monolingual corporas were 1M each, but we reduced them to be the same size as En–Fr datasets.

We use validation set to prevent overfitting and reduce training time. Training process stops when the BLEU score on validation set does not improve for five epochs. The model with best validation BLEU at that moment is a final one. We report final result on 5k parallel sentences used as test set.

Dataset	Corpus	En	Fr
En–Fr-20k	Parallel train	0 - 20k	0 - 20k
	Parallel validation	47.5 - 48.5k	47.5 - 48.5k
	Parallel test	50 - 55k	50 - 55k
	Monolingual En	200 - 500k	-
	Monolingual Fr	-	1M - 1.3M
Fr-10k	Parallel train	0 - 10k	0 - 10k
	Parallel validation	47.5 - 48.5k	47.5 - 48.5k
	Parallel test	50 - 55k	50 - 55k
	Monolingual En	200 - 500k	-
	Monolingual Fr	-	1M - 1.3M

Table 1: Data splits from original `europarl-7` for En–Fr pair

4.2 Preprocessing

We use the following preprocessing procedure.

1. First, we tokenize all data with standard NLTK[4] package for python.
2. Second, we apply byte-pair encoding (Sennrich et al., 2015) to reduce the size of the dictionary. This step is done using Subword-NMT[5] library. For En–Fr pair we use joint vocabulary of size 32k. And for En-Ru we use separate vocabularies 4k each. Vocabularies are fixed for all En–Fr datasets and for all En-Ru datasets.

4.3 Default model

For translation model we follow general transformer architecture (Vaswani et al., 2017), but as we were highly constrained by time and computational resources during the hackathon (we had 8 hours on NVIDIA GeForce GTX 1080 Ti for preprocessing, training and inference), we had to reduce several hyper-parameters values.

Model we used had the following parameters for all experiments: All embeddings had size $d = 512$, hidden layer of FFN, applied to embeddings in between attentional layers, had 2048 neurons. We used 4 attentional heads and 4 attentional layers. Our dropout rates were chosen to be 0.1 for attention, ReLU and residual connections. As an optimizer we used Adam optimizer with initial learning rate being equal $1e - 4$ and $\beta_2 = 0.98$. Weights are randomly initialized with normal distribution with mean 0 and standard deviation being equal $1/\sqrt{d}$.

5 Results

In this section we evaluate all variations of model on several datasets, to see how our models profits from fusion and initialization with language models.

5.1 20k experiments[6]

We start from evaluating four main models on En–Fr-20k and En-Ru-20k datasets. The training progress for En–Fr pair is shown on Fig. 2. The final results for both pairs on a test set are listed in table 2. We can notice, that models with initialization perform reasonably better than models without it. Also we see that initialization is essential if we use fusion. All training process for 20k data fits in 5 hours on GTX 1080 Ti, and a huge part of this time is used by validation after each epoch.

[4]http://www.nltk.org

[5]https://github.com/rsennrich/subword-nmt

[6]20k parallel train sentences, 1k parallel validation sentences, 300k monolingual corporas

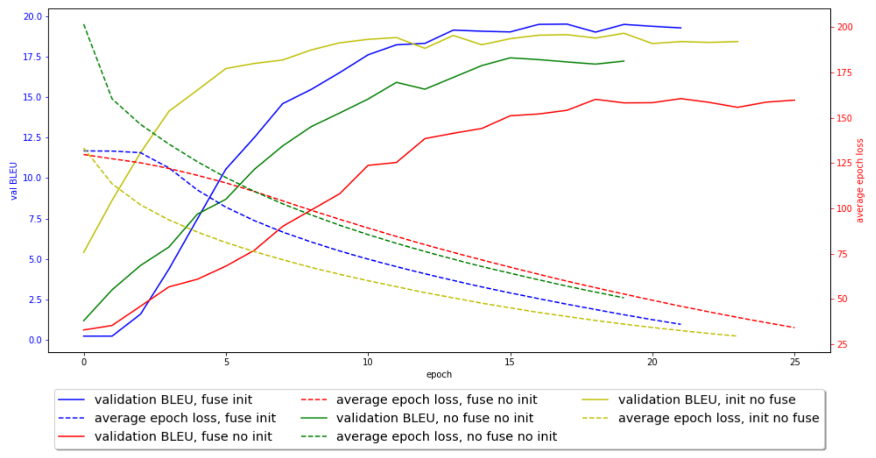

Figure 2: Training process for En–Fr-20k dataset

Model	BLEU En–Fr	BLEU En-Ru
Transformer	18.5	26.3
Transformer with LM initialization	19.7	27.7
Transformer without LM initialization, with fusion	16.1	26.0
Transformer with LM initialization and fusion	20.7	27.2

Table 2: Results on 20k datasets

5.2 10k experiments

During this series of experiments we see again that proposed approach improves BLEU score. Also it is worth to mention that for fused model this initialization is essential to achieve any reasonable score. Finally it is important to notice, that due to extremely small parallel corpora models tend to do a lot of factual mistakes in translations, which is shown well by BLEU score.

Model	BLEU En–Fr	BLEU En-Ru
Transformer	13.6	21.8
Transformer with LM initialization	16.1	23.0
Transformer without LM initialization, with fusion	<5	21.8
Transformer with LM initialization and fusion	15.9	22.4

Table 3: Results on 10k datasets

6 Discussion and Conclusion

In this work we presented an approach of initializing translation model with language models and a new technique for shallow fusion of LM into translation system. These approaches allow to improve existing NMT in the situations when there is a lack of parallel data by only using monolingual corpora. We experimentally showed an improvement up to +2.5 BLEU for the Transformer model on one of the setups. However we must notice that human review of translations for En-Ru pair where our model outperformed standard transformer by 1.4 BLEU, does not show any significant advantage of our approach. All of the proposed models perform quite well on short sentences and provide almost the same translations but on the sentences longer than 15 words, translations become notably different and with different mistakes. Also, we see that the performance of our version of fusion is not always good so this approach is yet to be analyzed.

Acknowledgment

We thank iPavlov laboratory for organizing DeepHack.Babel hackathon and providing us data and computational resources to do the work.

References

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2017). Unsupervised neural machine translation. *Computing Research Repository*, abs/1710.11041.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *Computing Research Repository*, abs/1409.0473.

Cheng, Y., Xu, W., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Semi-supervised learning for neural machine translation. *Computing Research Repository*, abs/1606.04596.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *Computing Research Repository*, abs/1710.04087.

Gülçehre, Ç., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H., Bougares, F., Schwenk, H., and Bengio, Y. (2015). On using monolingual corpora in neural machine translation. *Computing Research Repository*, abs/1503.03535.

Lample, G., Denoyer, L., and Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *Computing Research Repository*, abs/1711.00043.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *Computing Research Repository*, abs/1309.4168.

Sennrich, R., Haddow, B., and Birch, A. (2015). Improving neural machine translation models with monolingual data. *Computing Research Repository*, abs/1511.06709.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Computing Research Repository*, abs/1706.03762.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer learning for low-resource neural machine translation. *CoRR*, abs/1604.02201.