Neural Machine Translation with Additional Denoise Function (??)

Chen Xia
chenxia@cs.cmu.edu
Siyuan Wang
siyuanw@cs.cmu.edu

December 11, 2018

Abstract

TODO

1 Introduction

TODO:

- noisy mt backgroud

- mtnt dataset

- related work for the denoise idea

1.1 Datasets

The standard datasets used for pre-training NMT models consist of europarl-v7 and news-commentary-v10 corpora for the en/fr language pair, from which we created our train data and dev data after Moses corpus cleaning. There are 2,092,069 train samples and 1571 dev samples. We evaluate the model on the newstest2014 test set to compare with the results reported in [2]. The noisy text dataset we use is the MTNT dataset, which is built from noisy comments on Reddit1 and professionally sourced translations. We use the provided train split for direct finetuning and denoise layer training, and evaluate on the provided test split.

2 Model Description

2.1 Baseline Model

All our models are implemented with the faiseq toolkit [1] by Facebook Research [1]. We train several standard NMT models on the standard datasets mentioned above to get the baseline results.

First, we run a simple seq2seq NMT model, largely based on the model configuration in the MTNT paper [2]. The encoder is a bidirectional LSTM with 2 layers, and the decoder is a 2 layered LSTM. However, we use simply dot-product attention. The encoder embedding dimension is 512, all other dimensions are 1024. We tie the target word embeddings and the output projection weights. We train with Adam optimizer, learning rate 0.001 and dropout probability 0.3. We use subword-nmt toolkit [2] for BPE, with each side 16,000 BPE size following the same setting as MTNT paper.

[1] https://github.com/pytorch/fairseq
[2] https://github.com/rsennrich/subword-nmt

	Model	en-fr	fr-en
newstest2014	BiLSTM	34.8	29.8
	FConv	33.9	29.8
	BiLSTM(std) + finetune(mtnt)	34.3 (1ep) 33.6 (10ep) 33.3 (20ep)	29.8 (1ep) 29.5 (10ep) 29.3 (20ep)
MTNT	BiLSTM	24.6	23.9
	FConv	23.4	24.4
	BiLSTM(std) + finetune(mtnt)	29.5 (1ep) 32.6 (10ep) 33.6 (20ep)	31.2 (1ep) 34.6 (10ep) 35.4 (20ep)
	BiLSTM(std) + denoise(mtnt)	29.0	27.5
	BiLSTM(std) + res-denoise(mtnt)	29.4	27.4

Table 1: BLEU scores for baseline and finetuned models on newstest2014 and MTNT test

The second baseline model we run is a convolutional sequence to sequence model (FConv), as described in [1]. The configuration details we use can be found here [3]. For this model, we use a combined BPE model on both source and target languages with size 40,000, as recommended by the paper.

2.2 Naive Adaptation

With an NMT model trained on the standard datasets, we do naive domain adaptation by simply finetuning with the MTNT train data. We train with Adam optimizer but with 0.0001 learning rate until convergence.

2.3 Denoise Function

With a pre-trained seq2seq model, we fix all the parameters and add a denoise layer after the embedding layer in the encoder. We then only train the denoise layer with MTNT train data.

Our hypothesis is that this denoise function can learn how to transform a noisy source sentence to a clean one, so that the NMT model trained on clean corpora can have better and more robust performance.

The denoise function can be a simpel LSTM, bidirectional LSTM or tranformer layer etc. that keeps the dimension of the word embeddings. We also experiment with adding a residual layer by adding the input to the output of the denoise layer, which we hope can help restrain the denoise layer output as word embeddings.

3 Experiment Results

Baseline and Naive Adaptation As shown in Table 1, BiLSTM and FConv trained on the standard dataset (coded "std" in the table) are comparable in performance on newstest2014 and MTNT test. As BiLSTM has fewer parameters than the FConv model and takes less time to train on a single GPU, we decided to use the BiLSTM as baseline for other experiments.

Denoise Function To find a denoise function that fits in our settings, it should have the following properties: (1)handle variable length (2)have kind of encoding capability to normalize the input noisy text embedding. We choose a **bidirectional lstm with 1 layer as our denoise function**. Input size is the same as the embedding size, and the hidden size is half of the input embedding size, so it can output the same size of vector in our pretrained nmt model.

[3] https://github.com/pytorch/fairseq/blob/9dd8724577fd01511a59df962e856a7e30be4618/fairseq/models/fconv.py#L737

source	mais bon la je re stresse car j'ai aucune idée de comment contribuer car j'ai fait quasi que de la théorie jusqu'a maintenant
target	But anyway I'm stressed again because I have no idea how to contribute since I mostly did theory up until now.
baseline	But I am **happy** because I have no idea how to contribute, because I have made almost the same theory **as before**.
+finetune 20 epochs	But I'm **tired** because I have no idea how to contribute because I did almost like theory **until now <no period>**

Table 2: Naive doman adaptation examples

source	mais bon la je re stresse car j'ai aucune idée de comment contribuer car j'ai fait quasi que de la théorie jusqu'a maintenant
target	But anyway I'm stressed again because I have no idea how to contribute since I mostly did theory up until now.
baseline+finetune 20 epochs	But **I'm tired** because I have no idea how to contribute because I did almost like theory **until now <no period>**
denoise + nmt + residual	But **I'm sorry** because I have no idea how to contribute because I have made almost the theory **until now**.
cotrain + denoise + nmt + residual	However, **I am very sorry**, because I have no idea how to contribute, because I have almost had theory **up to now**.

Table 3: Naive domain adaptation vs denoise function

4 Discussion and Analysis

4.1 What improved with finetuning and denoising?

Translation result comparison (baseline vs finetune) We pick up the sentence that is generated by the following model:**(a)**pretrain on standard datasets, finetune on MTNT datasets **(b)**pretrain on standard datasets, then only train denoise function on MTNT dataset **(c)**cotrain nmt + denoise function on standard datasets, then finetune only denoise function. From [Table 3], among three models, model(c) generates very interesting results, which turn

- hand-picked examples

See [Table 2]

- formal/informal n-grams stats

4.2 Denoising Effects

Top K scores with cosine similarity To evaluate whether the denoise function is really doing the job, we perform the following analysis: we pick out the trained denoise function on MTNT data, and then use this part for the analysis. First, we find some sentence that has noisy text, such as *'sutecthat'*,

reference	"I'm": 50, "it's": 96, "I don't": 31, "doesn't": 20, 'I do not': 1
nmt	'I do not': 33, 'it is not': 19, 'not have': 11, 'That is': 9
nmt + 20 epoch	"I'm": 63, "it's": 134, ": 19, "I don't": 41, "doesn't": 17 'I do not': 2, 'not have': 1
co-train denoise + nmt	"I'm": 25, 'I do not': 37, 'it is not': 24, 'That is': 6, "I don't": 9 'not have': 11
denoise + residual + nmt	'I do not': 30, "it's": 3, "I'm": 30, 'That is': 6, "I don't": 12 'it is not': 19, 'not have': 8

Table 4: n-grams statistics under/over-generated

target	noisy text	Top 1 word	Top 2 word
Pretty sutecthat was an error though and some orders didnt get filled.	sutecthat	su ch that	sud that
😂 is this in reference to the hair?	😂	"	'
I go to Meetup groups at a Panera and I love the broccoli cheese soup but not the 800mgs of sodium	800mgs	800m grams	8Mgs

Table 5: Top k similar word embeddings using cosine similarity

which should be *"such that"*. Then we put this sentence into the denoise function, and get the hidden size of each time step. We then compare the **cosine similarity** of each word in the sentence with the whole embedding. Finally, we find top k scores subwords. Then we pick the top1 and top2 scores and concatenate the subword(which is sequential, and easy to concatenate). From the result, we could see that *"sutecthat"* could be converted to *"su ch that"*, which is quite good.

5 Conclusion

Despite

References

[1] J. GEHRING, M. AULI, D. GRANGIER, D. YARATS, AND Y. N. DAUPHIN, *Convolutional Sequence to Sequence Learning*, in Proc. of ICML, 2017.

[2] P. MICHEL AND G. NEUBIG, *Mtnt: A testbed for machine translation of noisy text*, arXiv preprint arXiv:1809.00388, (2018).