STRUCTURE PREDICTION WITH PYTORCH

Zekun Zhao
University of California, Santa Cruz
March 2021

A project report submitted to the department of
Computer Science and Engineering
in partial fulfillment of the requirements for
the Master's degree

The Master's Project of
Zekun Zhao is approved:

(Project Advisor)

Abstract

Currently, seq2seq models have been found to assign its global best score to the empty translation, revealing a massive failure of neural models in properly accounting for adequacy. A number of methods to fix this problem have been proposed. In this project, we focus on analysing the search errors and model errors in neural machine translation (NMT). The goal is to optimize the model by avoid generating $< STOP >$ token too early. Our experiments are based on the encoder-decoder approach with soft attention to translating between different languages. We apply the the classic structured perceptron loss to neural sequence to sequence models. Using Beam and DFS search algorithm to do an approximate and exact inference procedure for neural sequence models. Our results shows that for a small data set, an encoder-decoder approach model trained to convergence with early-stopping on a development set does not suffer this problem.

Acknowledgements

This work was done under the supervision of my advisor, Jeffrey Flanigan. Since I started working on my masters research, I have learned much about machine learning and natural language processing from him, and I appreciate his guidance and encouragement.

Contents

Abstract **ii**

Acknowledgements **iii**

1 Introduction **1**

2 Methods **2**

2.1 Stop criterion on the dev set 3

2.2 Teacher Forcing in Sequence-Level Objectives 3

3 Experiments **5**

3.1 Data used in Experiments 5

3.2 Model 6

3.3 Results 6

4 Conclusion **9**

Bibliography **10**

Chapter 1

Introduction

There has been much recent works (Meister, Vieira, & Cotterell, 2020; Edunov, Ott, Auli, Grangier, & Ranzato, 2017) on improving upon beam search for neural sequence-to-sequence models. Surprisingly, beam search fails to find these global best model scores in most cases, even with a very large beam size of 100. For more than 50% of the sentences, the model in fact assigns its global best score to the empty translation, revealing a massive failure of neural models in properly accounting for adequacy.

Most state-of-the-art results on machine translation tasks are attained using beam search despite its overwhelmingly high search error rate. The paper Stahlberg & Byrne (2019) points out that the recurrent LSTM, the convolutional SliceNet (Kaiser, Gomez, & Chollet, 2017) , and the Transformer Big systems are strong baselines from a WMT 18 shared task submission (Stahlberg, de Gispert, & Byrne, 2018) which we include in Table 1.1 and Table 1.2.

$Model(Beam_{10})$	$Bleu$	$SearchErr$
LSTM	28.6	58.4%
SliceNet	28.8	46.0%
Transformer-Base	30.3	57.7%

Table 1.1: Beam search fails to find these global best model scores in strong baselines from a WMT'18 shared task submission.

$Model(Exact)$	$Empty$
LSTM	47.7%
SliceNet	41.2%
Transformer-Base	51.8%

Table 1.2: For About 50% of the sentences, the model in fact assigns its global best score to the empty translation, revealing a massive failure of neural models in properly accounting for adequacy.

Chapter 2

Methods

In this project, we're trying to fix the problem that an exact decoder gives zero length outputs for neural machine translation (NMT) models. The decoding objective for NMT aims to find the most-probable hypothesis among all candidate hypotheses. To achieve this goal, we could optimize the loss functions that are either computed over individual tokens, over entire sequences or over a combination of tokens and sequences. An overview of two classic loss functions is given in Equation 2.1 and Equation 2.2.

Token Negative Log Likelihood (MLE) Token-level likelihood (MLE, Equation 1) minimizes the negative log likelihood of individual reference tokens $\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)$. It is the most common loss function optimized in related work and serves as a baseline for our comparison (Chen et al., 2019). This method work with the normal RNN training method, where the decoder is trained to predict based on $p(y_i|y_1 \ldots y_{i-1}, \mathbf{x})$.

$$\begin{aligned} L_{MLE} &= -\log p(\tilde{\mathbf{y}}|\mathbf{x}) \\ &= -\log \prod_{i=1}^{n} p(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) \\ &= -\sum_{i=1}^{n} \log p(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) \end{aligned} \tag{2.1}$$

Sequence-Level Objectives (Perceptron) A sequence-level training loss that enables comparisons between the entire generated and reference sequences. The Perceptron loss function (Collins, 2002) is motivated by the fact that if the training data can be perfectly classified, the function could directly minimize 0/1 prediction on the training set (Flanigan, 2018) . Let $\tilde{\mathbf{y}}$ be our reference sequence, and $\widehat{\mathbf{y}}$ be our model prediction. $(\mathcal{D}, \theta)$ are model search space and model parameters. For this loss we use the unnormalized scores computed by the model before the final softmax.

$$
\begin{aligned}
L_{Perceptron}(\mathcal{D}, \theta) &= -score(\tilde{y}|\mathbf{x}) + \max_{\widehat{\mathbf{y}} \in \mathcal{D}} score(\widehat{\mathbf{y}}|\mathbf{x}) \\
&= \sum_{i=1}^{n} -score(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) + \max_{\widehat{\mathbf{y}} \in \mathcal{D}} \sum_{i=1}^{n} score(\widehat{y}_i|\widehat{y}_1, \ldots, \widehat{y}_{i-1}, \mathbf{x})
\end{aligned} \tag{2.2}
$$

In the greedy approach for percetron loss, we could use the beam size equal to one for the training process.

$$
\begin{aligned}
L_{Perceptron-Greedy}(\mathcal{D}, \theta) &= \sum_{i=1}^{n} -score(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) + \max_{\widehat{y}_i \in \mathcal{D}(\widehat{y}_i)} \sum_{i=1}^{n} score(\widehat{y}_i|\widehat{y}_1, \ldots, \widehat{y}_{i-1}, \mathbf{x}) \\
&= \sum_{i=1}^{n} -score(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) + \sum_{i=1}^{n} \max_{\widehat{y}_i \in \mathcal{D}(\widehat{y}_i)} score(\widehat{y}_i|\widehat{y}_1, \ldots, \widehat{y}_{i-1}, \mathbf{x})
\end{aligned} \tag{2.3}
$$

2.1 Stop criterion on the dev set

Despite the decoder is optimized to give the highest score or probability to most-probable hypothesis in the model search space, somehow the model is still incorrectly predicting $score(< \text{START} >, < \text{STOP} > |\mathbf{x})$ too high. It is possible that the model hasn't been trained to convergence, and that is why $score(< \text{STOP} > | < \text{START} >)$ is higher than it should be. For an RNN, $w^T_{<\text{STOP}>}$ and $v^T_{<\text{STOP}>}$ are both the layer weights, $b_{<\text{STOP}>}$ is the bias term for the $< \text{STOP} >$ symbol.

$$
score(< \text{STOP} > |h_t) = \sigma\left(w^T_{<\text{STOP}>} x_{t-1} + v^T_{<\text{STOP}>} h_{t-1} + b_{<\text{STOP}>}\right)
$$

In inference time, simply adjusting the model $< \text{STOP} >$ bias by a constant value does not help resolve the empty translation issue, since every complete sentence is ended by $< \text{STOP} >$. The score for every possible output sentence is adjusted by reducing same amount of values.

To fix this, we have to train the model further to adjust the layer weights by using early-stopping and dynamically changing the learning rate based on the performance of dev set. For the evaluation metric, we care about the highest BLEU score model instead of the loss value.

2.2 Teacher Forcing in Sequence-Level Objectives

By far the most popular training strategy is via the maximum likelihood principle. In the RNN literature, this form of training is also known as Teacher forcing (Williams & Zipser, 1989), due to the use of the ground-truth samples $\tilde{y}_i$ being fed back into the model to be conditioned on for the

prediction of later outputs. These fed back samples force the RNN to stay close to the ground-truth sequence and improve model skill and stability.

In equation 2.2, we need to find the highest score among all candidate hypotheses. However, the NMT search space is vast as it grows exponentially with the sequence length. For example, for a common vocabulary size of $|\tau| = 32,000$, there are already more possible translations with 20 words or less than atoms in the observable universe ($32000^{20} \gg 10^{82}$). Thus, complete enumeration of the search space is impossible. We have to use a simple search heuristic, beam search to decode models. However, there is no formal guarantee that beam search will return or even approximate the highest score candidate under a model. In practice, beam search often fail to find a higher score than the model score of our reference sequence when the beam size is not big enough.

We introduce a variant of the structured Perceptron algorithm 2.4 but trained with Teacher forcing. We observe that this algorithm is actually optimizing towards the token level after applying the Teacher forcing on the perceptron loss. Instead of searching higher score among all candidate hypotheses, we can use the model output from a prior time step as an input and choose the highest output score as current step score. This approach ensures that we will get a higher score than the model score of our reference sequence since the hidden state for each step will be the same and we always pick the highest score for each step rather than the score for the next correct output token.

$$\begin{aligned} L_{Perceptron-TeachForce}(\mathcal{D}, \theta) &= \sum_{i=1}^{n} -score(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) + \sum_{i=1}^{n} \max_{\widehat{y}_i} score(\widehat{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) \\ &= \sum_{i=1}^{n} (-score(\tilde{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x}) + \max_{\widehat{y}_i} score(\widehat{y}_i|\tilde{y}_1, \ldots, \tilde{y}_{i-1}, \mathbf{x})) \end{aligned} \tag{2.4}$$

By applying this technique, we show we can reduce the training time of perceptron model dramatically, and ensure the loss is no-negative value.

Chapter 3

Experiments

The motivation for our experiments is to investigate the reason for the empty translation and possible approaches for alleviating this issue. We use the BLEU score as our stopping criterion for training which means the highest BLEU score model will become our best choice. We also have a comparison of Token-level likelihood model (MLE) and Sequence-level perceptron model. In order to improve the computational convergence rates and statistical efficiency, we introduce a variant of the structured Perceptron algorithm 2.4 but trained with Teacher forcing as an improvement upon the vanilla Perceptron algorithm for training sequence-to-sequence models.

3.1 Data used in Experiments

The data for this project is English to French translation pairs which are collected from the open translation site Tatoeba.[1] It is a large database of sentences and translations. Its content is ever-growing and results from the voluntary contributions of thousands of members. You can also find the extra work of splitting language pairs into individual text files here.[2] We sampled sentence pairs from the original dataset with the length constrain of 10 tokens per sentence and prefixes constrains. And the total number of counted french tokens is 4345; the total number of counted english token is 2803.

Language	*Corpus*	*Sentences*	$Tokens_{fr}$	$Tokens_{en}$
French-English	Train	8479	2587	3944
French-English	Dev	1600	975	1183
French-English	Test	1600	934	1174

Table 3.1: Corpora

[1] `https://tatoeba.org/`
[2] `https://www.manythings.org/anki/`

3.2 Model

The neural network architecture in our experiments is the Encoder Decoder network with soft attention, which is a model consisting of two RNNs called the encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder reads that vector to produce an output sequence. For our small dataset we can use relatively small networks of 256 hidden nodes and a single GRU layer. For each model, we restricted the number of training epoch to 20 epochs and set initial learning rate as 0.01. We used the library *torch.optim.lr_scheduler* from pytorch to provide several methods to adjust the learning rate based on the number of epochs. Decays the learning rate of each parameter group by gamma 0.95 every step size epochs. Notice that such decay can happen simultaneously with other changes to the learning rate from outside this scheduler. For optimizer, we use the SGD with momentum 0.5.

3.3 Results

Our experiments in Table 3.2 show that MLE models have a better Bleu score than Perceptron models under the constrain of 20 training epochs. And the Figure 3.1 shows that perceptron model has a slow convergence rate which could suggest that the perceptron model is not fully trained yet after 20 epochs, even though the teacher forcing technique works for both loss functions at first 20 epochs. Also, compared with results from Table 3.2, perceptron models have a lower Bleu score in the exact decoding scenario in Table 3.3. Since our dataset is very small, we could relatively easier train it and test the model under the exact decoding method. In fact, the empty translation is not an issue in our experiments under the exact decoding scenario.

$Model(lr = 0.01)$	$Bleu$	$LengthRatio$
MLE-noTF	49.8_{Greedy}	1.003
MLE-onlyTF	51.3_{Greedy}	1.002
MLE-Combined	49.7_{Greedy}	0.977
Perceptron-noTF	38.4_{Beam12}	1.033
Perceptron-onlyTF	37.8_{Beam12}	1.007
Perceptron-Combined	38.2_{Beam12}	1.005

Table 3.2: NMT with approximate inference method on models with Token Negative Log Likelihood (MLE) and Sequence Negative Log Likelihood (SeqNLL). Due to the slow training speed, we only trained each model for about 20 epochs. The results suggest Perceptron model is harder to train and suffers from slow convergence.

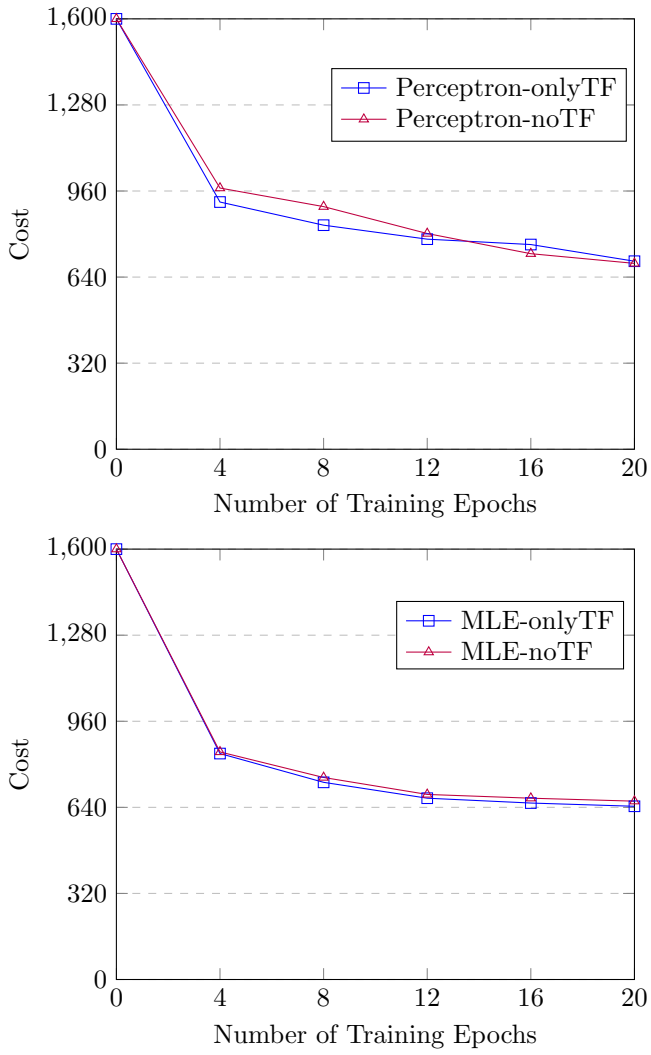

Figure 3.1: Cost changes during training in terms of the number of epochs. Cost represents the sum of 1 - score(Sentence-blue) for the whole validation set. The result shows that the teacher forcing technique works for both loss functions at first 20 epochs.

$Model(lr = 0.01)$	$Bleu(Exact_{appro})$
MLE-noTF	50.2
MLE-onlyTF	52.4
MLE-Combined	50.4
Perceptron-noTF	-
Perceptron-onlyTF	31.2
Perceptron-Combined	30.2

Table 3.3: A model trained to convergence with early-stopping on a development set has a better BLEU score under exact search and will not prefers the empty translation. $Bleu(Exact_{appro})$ is the exact decoding under the time constrain for 5 seconds per sentence.

Chapter 4

Conclusion

This work is motivated by the problem in training Seq2Seq models: the model often prefers the empty translation – an evidence of NMT's failure to properly model adequacy. Inspired by Teacher forcing technique, we propose the usage of the prior time step's ground truth as input for training a sequence-level loss to improve Seq2Seq learning. There are some important observations in this project:

1. Empty translation should not be an issue for a model trained to convergence with early-stopping on a development set.

2. Teacher Forcing can be applied in the Sequence-Level Objectives (Perceptron) and helps the model converge faster.

3. Teacher Forcing technique can mitigate the issue of empty translation by improving the model computational convergence rates and statistical efficiency.

Bibliography

Chen, L., Zhang, Y., Zhang, R., Tao, C., Gan, Z., Zhang, H., ... Carin, L. (2019). Improving sequence-to-sequence learning via optimal transport. *arXiv preprint arXiv:1901.06283*.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 conference on empirical methods in natural language processing (emnlp 2002)* (pp. 1–8).

Edunov, S., Ott, M., Auli, M., Grangier, D., & Ranzato, M. (2017). Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*.

Flanigan, J. (2018). Parsing and generation for the abstract meaning representation. *Language Technologies Institute School of Computer Science Carnegie Mellon University. Ph.D. Thesis*.

Kaiser, L., Gomez, A. N., & Chollet, F. (2017). Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*.

Meister, C., Vieira, T., & Cotterell, R. (2020). Best-first beam search. *Transactions of the Association for Computational Linguistics*, *8*, 795–809.

Stahlberg, F., & Byrne, B. (2019). On nmt search errors and model errors: Cat got your tongue? *arXiv preprint arXiv:1908.10090*.

Stahlberg, F., de Gispert, A., & Byrne, B. (2018). The university of cambridge's machine translation systems for wmt18. *arXiv preprint arXiv:1808.09465*.

Williams, R. J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, *1*(2), 270–280.