Neural Network Joint Language Model: An Investigation and An Extension With Global Source Context

Ruizhongtai (Charles) Qi
Department of Electrical Engineering, Stanford University
rqi@stanford.edu

Abstract

Recent work has shown success in using a neural network joint language model that jointly model target language and its aligned source language to improve machine translation performance. In this project we first investigate a state-of-the-art joint language model by studying architectural and parametric factors through experiments and visualizations. We then propose an extension to this model that incorporates global source context information. Experiments show that the best extension setting achieves 1.9% reduction of test set perplexity on a French-English data set. [1]

1 Introduction

The construction of language model has always been an important topic in NLP. Recently, language models trained by neural networks (NNLM) have achieved state-of-the-art performance in a series of tasks like sentiment analysis and machine translation. The key idea of NNLMs is to learn distributive representation of words (aka. word embeddings) and use neural network as a smooth prediction function. In a specific application like translation, we can build a stronger NNLM by incorporating information from source sentences. A recent work from ACL 2014 (Devlin et al., 2014) achieved a 6+ BLEU score boost by using both target words and source words to train a neural network joint model (NNJM).

In this project, we implement the original NNJM and design experiments to understand the model. We also extend the current NNJM with global context of source sentences, based on the intuition that long range dependency in source language is also an important information source for modelling target language.

Our contribution mainly lies in three aspects: First, we present a deep dive into a state-of-the-art joint language model, and discuss the factors that influence the model with experimental results; second, we propose a new approach that incorporates global source sentence information into the original model, and present our experimental results on a French-English parallel dataset; third, as a side contribution, we have open-sourced[2] our implementation of both the two models, which could be run on both CPU and GPU with no additional effort.

The rest of this report is organized as follows. We first give a brief introduction on NNJM in Section 2. Then in Section 3 we present our extensions: We introduce how we compute source sentence vectors and why we make these design choices. We then spend more space present our insights on NNJM gained from experiments, and evaluation of our extended NNJM model in Section 5. We summarize related work in Section 6 and conclude in Section 7.

2 Neural Network Joint Model

For statistical machine translation, we can augment target language model with extra information from source sentence. An effective approach proposed previous work is to extend traditional NNLMs by concatenating a context window of source words into the input and train word vectors for both source and target languages. In Section 3 we will also describe another extension of NNLM of using source sentence vector as an extra source of information.

2.1 Model Description

We use a similar model as the original neural network joint model. To be concrete, we provide mathematical formulation for the model together with a model illustration in Figure 1. For more details please refer to the original BBN paper (Devlin et al., 2014).

One sample input to the model is a concatenated list of words composed of both target context words (n-1 history words for n-gram) T_i and source context words S_i. Source words are selected by looking at which source word target word t_i is aligned with, say it's s_{a_i}, then we take a context window of source words surrounding this

[1]This project is advised by Thang Luong and it is a co-project for CS 224N Natural Language Processing with Yuhao Zhang.

[2]http://goo.gl/WizzCF

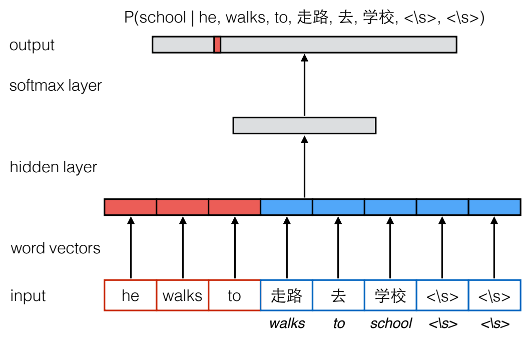

Figure 1: Neural network joint model with an example (illustrated with Chinese-English) where we use 4-gram target words (3 words history) and source context window size of 2. We want to predict the next word following *he, walks, to* and hopefully estimated probability of the next word being *school* would be high.

aligned source word. When the window width is $\frac{m-1}{2}$, we have m source words in the input.

$$p(t_i \mid T_i, S_i)$$

$$T_i = t_{i-1}, ..., t_{i-n+1}$$

$$S_i = s_{a_i - \frac{m-1}{2}}, ..., s_{a_i}, ..., s_{a_i + \frac{m-1}{2}}$$

Here we regard t_i as output, i.e. $y \in \mathbf{R}$ as one of the target words, and concatenation of T_i and S_i as input, i.e. $x \in \mathbf{R}^{n+m-1}$ of $n-1$ target words and m source words. The mathematical relation between input and output is as follows, where $\Theta = \{L, W, b^{(1)}, U, b^{(2)}\}$. Linear embedding layer $L \in \mathbf{R}^{d \times (V_{src}+V_{tgt})}$ which converts words to word vectors by lookup, where d is word vector dimension. In hidden layer, $W \in \mathbf{R}^{h \times (d*(n+m-1))}$, $b^{(1)} \in \mathbf{R}^h$. In softmax layer, $U \in \mathbf{R}^{V_{tgt} \times h}$, $b^{(2)} \in \mathbf{R}^{V_{tgt}}$ and

$$g_i(v) = \frac{\exp(v_i)}{\sum_{k=1}^{V_{tgt}} \exp(v_k)}$$

$$p(y = i \mid x; \Theta) = g_i(Uf(WL(x) + b^{(1)}) + b^{(2)})$$

Optimization objective is to maximize the log-likelihood of the model.

$$\ell(\Theta) = \sum_{i=1}^{m} log(p(y^{(i)} \mid x^{(i)}; \Theta))$$

2.2 Evaluation Metric

We use *perplexity* as the metric to evaluate quality of a language model.

$$PP(W) = p(w_1, w_2, ..., w_N)^{\frac{-1}{N}}$$

3 Neural Network Joint Model with Global Source Context

In this section, we show our attempts in pushing the state-of-the-art of NNJM by utilizing global source context (global source sentence information). For simplicity, we will use NNJM-Global to refer to this extension in the following sections.

3.1 Weighted Sum Source Sentence

Our first attempt is to include sentence vector directly into the input layer of the neural network. We calculate weighted sum of word vectors in source sentence, and feed the result into our input layer, as shown in Figure 2. Specifically, we experimented with two different approaches:

1. **Uniform weights** We assign each word a uniform weight in the source sentence. In another word, we take the mean of all the word vectors to form the global context vector.

2. **Zero weights for stop words** Instead of giving all words the same weight, we identify top *N* frequent words in the vocabulary as stop words, and assign each of them with a zero weight. For all the rest words in the vocabulary, we still assign them with a uniform weight. The intuition is depress possible noises introduced from those irrelevant words.

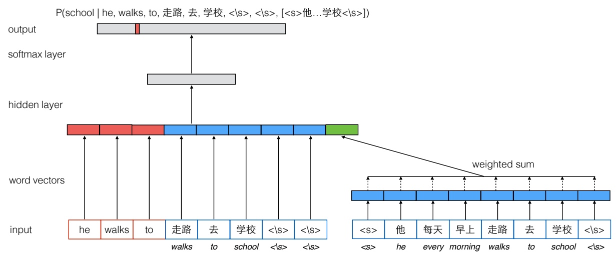

Figure 2: An example for the NNJM with global context, where an additional source sentence is fed into the model.

3.2 Splitting Source Sentence Into Sections

In order to increase expressibility of sentence vectors, we propose to split the source sentence into sections before taking weighted sum, as shown in Figure 3. We treat the number of sections as a hyper-parameter for this model. Specifically, we experimented with two variants of this approach:

1. **Fixed section length splitting** The sentence vector is first extended with end-of-sentence tokens so that all the input source sentences are of the same length.

Then the splitting is done on the populated source sentences.

2. **Adaptive section length splitting** We use the original source sentence instead of extending all sentence vectors into a uniform length. Thus, each section will have a variable length dependent on the length of the entire sentence.

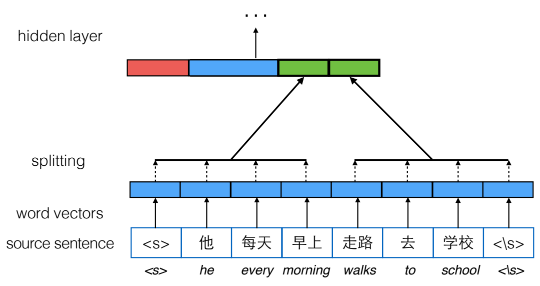

Figure 3: An example of splitting source sentence into 2 sections before calculating the global context vectors. Weighted sum is calculated on the first half of the sentence to form the first global context vector, and then on the second half.

3.3 Global-only Non-linear Layer

We can also add non-linearity to the model by adding another global-only non-linear layer between the global linear layer and the downstream hidden layer, as it is illustrated in Figure 4. Note that this non-linear layer is only added for the global part of the model, and has no effect on the local part. We use the same non-linear function for this layer as in other layers of the model.

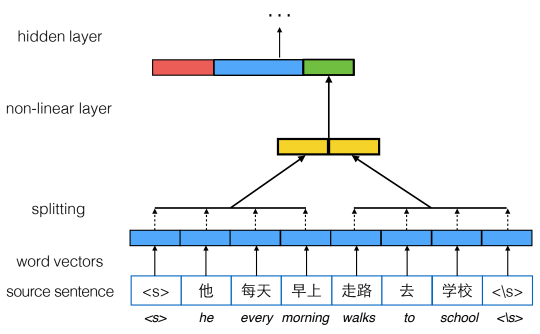

Figure 4: An example for the non-linearity on the global source sentence. A weighted sum is calculated to form the intermediate global context vectors (yellow), and then these intermediate vectors are fed into a global-only non-linear layer.

4 Model Training

Following a similar strategy with BBN paper, we use mini-batch gradient descent to maximize log-likelihood on training set. Each batch contains 128 input samples, each of which is a sequence of target words plus source context words. There are around 22K mini-batches per epoch. Model parameters are randomly initialized in the range of [-0.05, 0.05]. We use early stopping to pick the model with least validation set perplexity. At the end of every epoch we do a validation set test and see if the validation set perplexity becomes worse from last time, if it is worse we halve the learning rate.

The data set we use is from European Parallel Corpus. Our training set contains 100K pairs of parallel French-English sentences. Validation and test set each contains 1K pairs of French-English sentences. For NNJM model analyzing, we use the entire data set. However, due to limit of time, we also use a 25K subset of the data for some of the experiments such as NNJM and NNJM+Global comparisons.

Both training and testing are implemented using Python. We use Theano Library for neural network modeling. The training runs on a single Tesla GPU. Training speed is around 1,500 samples/second and training one epoch of data (128*22K) takes around half an hour. For reference, total training time for a basic NNJM model over the entire corpus is around 2.5 hours when the full GPU power is utilized.

5 Experimental Results

5.1 NNJM

In this subsection, we focus on showing our understanding of the joint language model. More evaluation results will be combined with NNJM+Global model in Subsection 5.2.

5.1.1 Effects of Hyperarameters

Tuning is on the entire 100K training set. Since a full grid search is too time consuming we will start from a default hyper parameter setting and change one of them each time. In default, learning rate is 0.3, target n-gram size is 5, source window width is 5 (11 source words), vocab size is 20K for both target and source language, epoch number is 5, word vector size is 96 and there is one hidden layer of 128 units.

As shown in Figure 5, hyper-parameters can have a big difference on model performance. Generally it helps to increase word vector dimensions and hidden layer size. Source window width 3 or 5 is good and target n-gram size of 4 is optimal in our default setting.

For learning rate, we train the model until convergence (around 25 epochs yet the decrease of valid set perplexity is marginal after 5 to 10 epochs). We observe that while

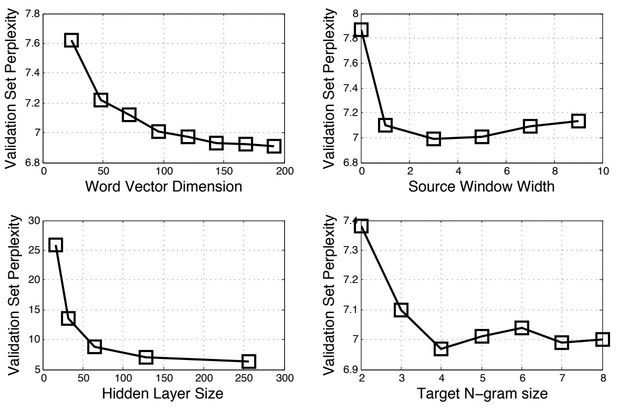

Figure 5: Effects of Hyper-parameters

very large learning rate such as 1.0 and 3.0 leads to quick convergence yet tend to rest at unsatisfactory local minimums, very small learning rate such as 0.03 converges too slow. We think lr=0.3 achieves balance between convergence speed and stability.

5.1.2 Visualizations and Insights

In this subsection we use network parameter visualization to show how the neural network take advantage of source context. Specifically, we look at the linear transformation matrix W in the hidden layer, which can be thought as a way to measure how much certain part of input contribute to predicting the next word. In Figure **??**, we observe that the center source word, i.e. the one aligned with the next target word, contributes most to the prediction, even more than the last history target word (index 14). There is a trend of attenuating importance for source words far from the middle one.

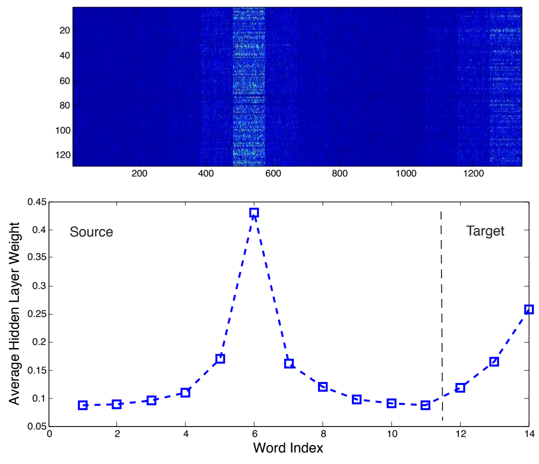

Figure 6: Top: Heat map of absolute values of hidden layer matrix $W \in \mathbf{R}^{h\times(d*(n+m-1))}$. Bottom: Average of W's elements corresponding to each input words. Words 1 to 11 are from source with word 6 as the center one. Words 12 to 14 are history words in the target n-gram.

5.2 NNJM-Global

In this subsection we demonstrate experimental results for each variant of the NNJM-Global model, and compare their results with the vanilla NNJM model. Note that all the models in this part are trained with the same strategy described in previous section. By default, we use a vocabulary size of 10K, a source window size of 3, a target n-gram size of 4, an embedding dimension of 96, a hidden layer size of 128, and a learning rate of 0.3 to train the models.

5.2.1 Comparing NNJM-Global with NNJM

The resulting perplexity achieved by different models on the test set is shown in Table 1. Note that we also include the result for a basic neural network language model (NNLM) where only target words are utilized for making predictions, to demonstrate the effect of global source context information. It is observed that for each setting of source window size, the NNJM-Global model achieves smaller (better) test set complexity compared to its corresponding NNJM model. The best performance is achieved when the source window size 3. Under this setting, a slightly better result is achieved when we use a zero-weights-for-stop-words weighted sum strategy. There is no noticeable difference between the different settings of number of stop words in the NNJM-Global model.

Model	SrcWin	Perplexity
NNLM	-	95.06
NNLM-Global	-	94.73
NNJM	7	10.09
NNJM-Global	7	10.05
NNJM	5	9.89
NNJM-Global	5	9.71
NNJM	3	9.51
NNJM-Global	3	9.45
NNJM-Global + SW-10	3	**9.44**
NNJM-Global + SW-25	3	**9.44**

Table 1: Test set perplexity for different models. *SrcWin* represents the source window size that is used in the model. *SW-N* represents that N most frequent stop words are removed from the global sentence vector. Results for the NNLM model where only target words are used for prediction are also included.

5.2.2 Effect of Splitting Source Sentence

Both the two approaches for splitting the global source sentence vectors are evaluated and compared to the basic NNJM and NNJM-Global models. The results are shown in Table 2. The fixed section length splitting strategy with section number of 2 gives reduction of the test set perplexity when compared to the basic NNJM-Global model, while the adaptive section length splitting strategy gives almost the same result as the basic NNJM-Global model, and also achieves better result compared to the original NNJM model. The performance is observed to deteriorate when the section number increases.

Model	NumSec	Perplexity
NNJM	-	9.51
NNJM-Global	1	9.45
NNJM-Global + FixSplit	4	9.54
NNJM-Global + FixSplit	2	**9.38**
NNJM-Global + AdaSplit	2	9.46

Table 2: Test set perplexity for models with different global context vector section numbers. *NumSec* is for section number in for global context vector. *FixSplit* denotes the model with fixed section length and *AdaSplit* denotes the model with adaptive section length.

5.2.3 Effect of Global-only Non-linear Layer

Generally, adding a non-linear layer could add expression power to the neural network. Table 3 shows the effect of adding non-linear layer for generating sentence vector under various settings. The best perplexity is achieved with FixSplit of 2 sections and non-linear size 192, which is 1.9% lower than the basic NNJM model. One possible explanation for the improvement is that with the help of section splitting the non-linear model can gain additional expressive power from this combination of architecture settings that is not possible without splitting or non-liearn layer.

6 Related Work

While BBN's work (Devlin et al., 2014) on neural network join model focused on efficiency and MT result presentation, we investigate deep into the original NNJM by study on hyper parameters and visualization of hidden layer weights. We also extend the model with global source context and achieves improvement in terms of perplexity scores. Our project have taken similar strategy in generating sentence vector with previous work on using sentence vector to learning word embeddings with multiple representations per word (Huang et al., 2012). However, we have also developed more complex models and focus more on designing good architecture to improve joint language model quality.

Model	NS	NLSize	Perp
NNJM	-	-	9.51
NNJM-Global	1	-	9.45
NNJM-Global + NL	1	96	9.45
NNJM-Global + NL	1	192	9.45
NNJM-Global + FixSplit	2	-	9.38
NNJM-Global + FixSplit + NL	2	96	9.61
NNJM-Global + FixSplit + NL	2	192	**9.33**
NNJM-Global + AdaSplit	2	-	9.46
NNJM-Global + AdaSplit + NL	2	96	9.55
NNJM-Global + AdaSplit + NL	2	192	9.47

Table 3: Test set perlexity for models with global-only non-linear layers. Results for models with no global vector splitting, with fixed section length splitting, and with adaptable section length splitting are shown. *NL* is for non-linear layer in the global part. *NLSize* represents the size of the global-only non-linear layer.

7 Conclusion

In this report we present our work in investigating a neural network joint language model and extending it with global source context. Our experimental analysis demonstrates how hyperparameters influence performance of the model. We also use visualization of network weights to show how source words influence prediction. Furthermore, we have shown that incorporating global source context (sentence vector) can further improve the performance of the language model in terms of perplexity. Finally, we have open-sourced our implementation of both the original model and the extended model.

Acknowledgements

The author sincerely acknowledge Thang Luong in Stanford NLP Group for his advising and thank Yuhao Zhang for his collaboration and thank CS 229 staff for bringing us such a rewarding class.

References

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, June*.

[Huang et al.2012] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.