NEURAL NETWORK JOINT MODELING VIA CONTEXT-DEPENDENT PROJECTION

Yik-Cheung Tam, Yun Lei

Speech Technology and Research Laboratory, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025, USA
{wilson,yunlei}@speech.sri.com

ABSTRACT

Neural network joint modeling (NNJM) has produced huge improvement in machine translation performance. As in standard neural network language modeling, a context-independent linear projection is applied to project a sparse input vector into a continuous representation at each word position. Because neighboring words are dependent on each other, context-independent projection may not be optimal. We propose a context-dependent linear projection approach which considers neighboring words. Experimental results showed that the proposed approach further improves NNJM by 0.5 BLEU for English-Iraqi Arabic translation in N-best rescoring. Compared to a baseline using hierarchical phrases and sparse features, NNJM with our proposed approach has achieved a 2 BLEU improvement.

Index Terms— Neural network joint modeling, context-dependent linear projection, position-dependent linear projection, statistical machine translation

1. INTRODUCTION

Recently, neural network research has inspired a lot of interest in various areas such as automatic speech recognition and statistical machine translation. In both tasks, language modeling plays a crucial role in performance improvement. Feed-forward neural network language modeling (NNLM) [1, 2] and translation modeling [3, 4, 5] are some recent efforts producing performance improvement. In statistical machine translation (SMT), the neural network joint model (NNJM) [6] has seen dramatic improvement in machine translation, yielding 2–3 BLEU improvement over a strong SMT baseline. The success of NNJM is due to its ability to leverage a wide word context (e.g., 11-word window) from a source sentence, compared to NNLM or a recurrent neural network language model (RNNLM) [7] that only uses target word context.

Similar to NNLM [8], NNJM employs shared linear projections to map a sparse word input into a continuous representation at each contextual word position: one for the source word context and the other for the target word context. The linear projection matrices are estimated during backpropagation. Usually, the projected continuous representation is viewed as some underlying semantic/syntactic information of an input word. The rationale is similar to the conventional class-based language modeling [9], where each word is first mapped to a discrete class label. In both cases, the word-to-class mapping is context-independent and insensitive to surrounding word context. For instance, word, e.g. "bank", can have multiple meanings, namely a financial bank or a river bank, depending on the context. However, the context-independent linear projection will always map "bank" to the same point in the continuous space that may be suboptimal.

In this paper, we propose a context-dependent linear projection that takes into account surrounding word context for NNJM. Motivated by [10, 11], where a filter spanning a window of words is applied at each word position within a convolutional neural network, we employ the same idea to allow context-dependent linear projections on a window of source words and target words. With this approach, the word "bank" will be mapped onto different continuous representations, depending on the surrounding context, for more accurate modeling.

In addition, we investigate a position-dependent linear projection to study whether word positions would be useful for better modeling. First, each input word position has a dedicated linear projection matrix. This leads to significantly more model parameters to estimate and thus the model parameters may not be well estimated due to data sparsity. Therefore, all position-dependent projection matrices have a shared component and a position-dependent component. The shared component serves as a context-independent linear projection, while the position-dependent component makes further adjustment. We expect that the shared component has a regularization effect towards the position-dependent components.

The paper is organized as follows: In Section 2, we review NNJM, followed by the proposed approach in Section 3–4. We describe experiments and results in Section 5. We discuss conclusions in Section 6.

2. REVIEW OF NEURAL NETWORK JOINT MODELING

The advantage of NNJM is its ability to exploit a source sentence for predicting a target word. Similar to NNLM, NNJM employs a multi-layer architecture as shown in Figure 1. In the first layer, each word position in a context is encoded as a 1-to-V sparse vector where V denotes the size of the vocabulary. Each sparse vector is then projected into a continuous space using shared projection matrices W_0 and U_{-1} for the source language and the target language respectively.

Given a source sentence $F = f_1...f_i...f_I$, a target sentence $E = e_1...e_j...e_J$, and the corresponding word alignment sequence A=$a_1...a_j...a_J$ where a_j denotes a set of source word positions aligned to e_j, NNJM has the following mathematical form:

$$P(e_j|e_{j-1}...e_1, F, A) \approx P(e_j|e_{j-1}...e_{j-n+1}, F_{[\bar{a}_j-t,\bar{a}_j+t]})$$

where e_j is the predicted target word at position j. $\bar{a}_j$ denotes the averaged source position that is aligned with e_j. Since a target word may align to multiple source words, the averaged source position is computed for simplicity. Following the convention in [6], if e_j does not align to any source word, an alignment variable of the next word e_{j+1} is inherited. The averaged source word position helps to locate a block of source words centered at $\bar{a}_j$ for target word prediction.

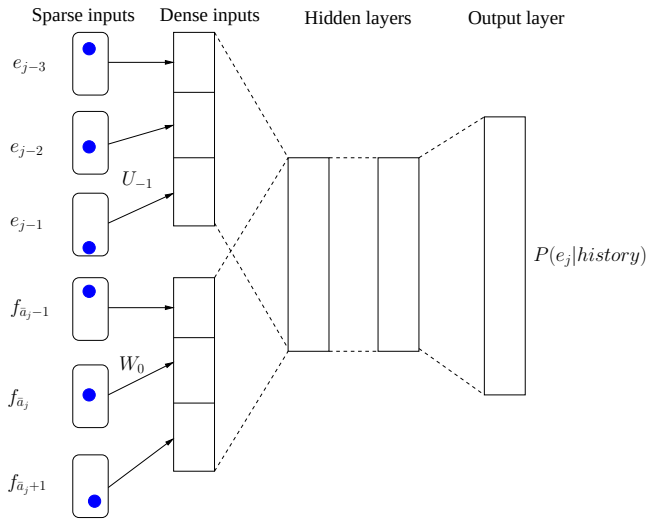

Fig. 1. Architecture of NNJM.

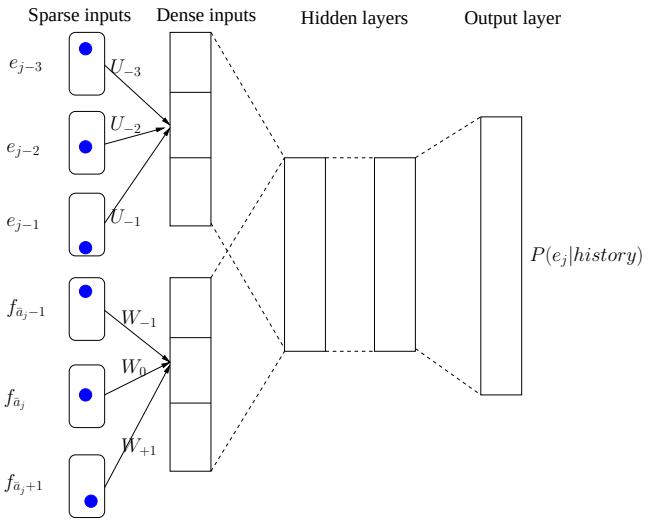

Fig. 2. Architecture of NNJM with context dependent linear projections: $U = \{U_{-3}, U_{-2}, U_{-1}\}$ and $W = \{W_{-1}, W_0, W_{+1}\}$.

With $n = 4$ and $t = 5$, the above NNJM takes a 14-gram context: 3-gram target word context and 11-gram source word context. To train NNJM, the standard backpropagation algorithm can be applied to estimate the network weights, including the projection matrices, using maximum likelihood. We perform a simple count-cutoff strategy to limit the size of the source and target vocabulary by mapping singleton words to an unknown token.

3. NNJM WITH CONTEXT-DEPENDENT PROJECTION

One assumption in NNLM/NNJM modeling is the use of a context-independent linear projection matrix:

$$x_i = W_0 \cdot f_i \quad (1)$$
$$y_{j-1} = U_{-1} \cdot e_{j-1} \quad (2)$$

where f_i and e_{j-1} denote sparse vectors on the source and target language. x_i and y_{j-1} are the corresponding dense vectors. This implies that, for instance, the word "bank" as in "river bank" or "financial bank" is projected into the same continuous representation. For more accurate modeling, word context should be taken into consideration during linear projection as shown in Figure 2. For instance, to project a source word f_i, a source context window $[f_{i-1}, f_i, f_{i+1}]$ is applied:

$$x_i = W_{-1} \cdot f_{i-1} + W_0 \cdot f_i + W_{+1} \cdot f_{i+1} \quad (3)$$

To project a target history word e_{j-1}, an N-gram target history window $[e_{j-3}, e_{j-2}, e_{j-1}]$ can be employed:

$$y_{j-1} = U_{-3} \cdot e_{j-3} + U_{-2} \cdot e_{j-2} + U_{-1} \cdot e_{j-1} \quad (4)$$

With a window of size 3, the number of parameters in the projection matrices is 3 times larger than the conventional NNJM. With $n = 4$ and $t = 5$, 5-gram target word context and 13-gram source word context are utilized for linear projection. However, the number of dense vectors after linear projection is still identical to conventional NNJM.

4. NNJM WITH POSITION-DEPENDENT PROJECTION

Another way to precisely model input projection is to have separate linear projection matrix per input word position. However, this will increase the number of model parameters proportional to the number of input context and thus insufficient training data may be an issue. To alleviate this, we enforce the position-dependent linear projection matrices to have a shared matrix component:

$$U'_j = U_j + U \quad (5)$$
$$W'_i = W_i + W \quad (6)$$

where i and j denote the input word positions, and U and W are the shared matrices across word positions on the target and source side respectively. We anticipate that the shared matrices are essential to enforce some structure during learning. For analysis, we assume that all matrices are initialized with zeros and the learning rate is 1. After processing the first mini batch, the updated shared matrix U is calculated as the summation of gradients over all input positions in gradient ascent:

$$U^{(new)} = \sum_j \Delta U_j \quad (7)$$

Due to linearity, we can see that the updated U'_j is:

$$U_j^{'(new)} = \Delta U_j + \sum_j \Delta U_j \quad (8)$$
$$= 2 \cdot \Delta U_j + \sum_{j' \neq j} \Delta U_{j'} \quad (9)$$

Eqn 9 means that the projection matrix at the j-th position is calculated similarly as the shared matrix in Eqn 7 but the corresponding position gradient is boosted by 2. The result is intuitive. Each input word is first projected into a space using the context-independent linear projection, then followed by a position-dependent adjustment.

5. EXPERIMENTAL SETUP

Our translation engine was built on data from the DARPA TRANSTAC program, a speech-to-speech translation initiative targeting tactical

military communication [12]. The source language was conversational English, and the target language was Iraqi Arabic. This MT direction is more challenging because valid morphology and word order in the MT output must be maintained, and data scarcity for LM training is a greater problem in Iraqi Arabic. We had $760K$ parallel sentence pairs as training data and 6985 sentence pairs for tuning the log-linear weights for dense and sparse features. The tuning set had a single reference, and all test sets had 4 references. We filtered the tuning set by skipping short dialogues that contained less than three sentences/turns; many of them were simple sentences such as "thank you" or "you are welcome." Details are shown in Table 1.

Table 1. Sizes of translation data sets used.

Data	Sentences	Source words
Train	760200	7207779
Tune	6985	64193
Test1	567	6855
Test2	655	10652
Test3	617	9203

We applied a word segmenter on the Iraqi Arabic text to segment affixes on words. All models were built using the segmented data, and translations were post-processed into word forms for BLEU score computation. In our Hiero SMT baseline, we incorporated 12 dense features for each bilingual stochastic context-free grammar (SCFG) rule after the Hiero grammar in [13], including IBM Model-1 scores in both source-to-target and target-to-source directions, relative frequencies in both directions, count of phrases, count of Hiero rules, number of source content words aligned to target spontaneous words, number of target spontaneous words aligned to source content words, three binned frequencies, and the number of unaligned source words. We further computed lexical pairs seen in a dictionary, affix sequences/ngrams, fertility for each word in the SCFG rule, and additional spontaneous/content word mismatches as sparse features. In total, we had $368,524$ sparse features. Optimization methods such as MIRA [14] or PRO [15], which can optimize millions of sparse features, were employed.

The training data were aligned using the grow-diag-final option with GIZA word alignment in both directions. Then the aligned sentence pairs were fed into NNJM training while the target side was fed into NNLM and RNNLM training. The vocabulary sizes of Iraqi Arabic and English were 30k and 20k respectively after mapping singleton words into an unknown token. 10% of the training data was kept for cross-validation on word perplexity to ensure that training was on the right track, although word perplexity had little correlation with translation performance using BLEU [16]. We compared the following neural network modeling approaches:

- NNLM
- RNNLM
- Bilingual RNN [17]
- NNJM with 11-word source window and 3-word target window [6]
- NNJM with context-dependent (CD) projection (this paper)
- NNJM with position-dependent (PD) projection (this paper)

All neural network models employed 600 hidden nodes trained on the same data and vocabulary.

For baseline RNNLM and bilingual RNN training, we employed 100 output classes and used backpropagation through time using the flag "-bptt 4 -bptt-block 10". These are the default settings in the RNNLM toolkit [7]. The same training and cross-validation sets were used, but with a sequential sentence order to allow RNN to capture dialogue-level discourse via the recurrent hidden vector. RNNLM training took 5 days to finish on a single CPU. The initial learning rate was chosen as 0.1; the learning rate started to halve when the reduction in cross-validation perplexity was small enough. For bilingual RNN [17], a bag-of-words (BOW) representation of a source sentence was used as additional input for RNN training, similar to [18]. Meanwhile, we performed NNJM training using a GPU with a minibatch size of 128 samples and an initial learning rate of 0.06. We randomized training samples before training. As with RNNLM training, the learning rate started to halve when the reduction in cross-validation perplexity was small enough. NNJM training took 2 days to finish using Theano [19]. Motivated by [6], the output sizes of the projection matrices were set to 192. In addition, NNJM had 2 hidden layers with 600 hidden nodes in each layer. Empirically, this only contributed 0.1 BLEU improvement compared to a single hidden layer architecture.

For N-best list reranking, we applied our baseline translation engine to generate up to 2000 N-best hypotheses per source sentence. The combined weighted score was associated with each hypothesis so that the score was further combined with a score from NNJM for reranking:

$$\text{score(rerank)} \quad = \quad \text{score(base)} + \lambda_{nn} \cdot f_{nn}(F, E)$$

where $f_{nn}(F, E)$ denotes different kinds of neural network modeling scores for a sentence pair F and E. λ_{nn} was optimized using a simple grid search.

5.1. Reranking results

Table 2 shows N-best reranking performance on BLEU using various neural network models. A 4-gram NNLM yielded 0.3 BLEU improvement compared to the SMT baseline with sparse features. Context-dependent input projection on NNLM provided an additional 0.1 BLEU improvement, achieving the same performance as RNNLM. Bilingual RNN with BOW representation performed better than RNNLM by 0.4 BLEU overall. This showed that the bag-of-word representation of a source sentence provided useful information for SMT. NNJM outperformed bilingual RNN with an additional 0.7 BLEU improvement. This implies that the word ordering of a source sentence provided additional information compared to bag-of-word representation. Since RNN training employed factorized output classes, the estimated probabilities may be suboptimal compared to NNJM using the full output vocabulary. With position-dependent linear projection, we observed additional 0.3 BLEU improvement compared to the NNJM (CI) baseline. This may suggest that relative word position within an input window provides useful information for better projection. To show the importance of the shared matrices in Eqn 5–6, we removed the shared components and retrained the model. We observed slight degradation compared to the NNJM baseline. This may be due to the increased number of parameters (7 times more projection matrices for a 14-gram NNJM) without proper regularization.

With context-dependent linear projection using 3-word source and target windows in NNJM, we achieved additional 0.5 BLEU improvement compared to conventional NNJM. Result showed that contextual information was important for more accurate linear projection. When we apply position-dependent modeling on context-dependent projection matrices, we did not observe further gain. This

may imply that word context window and word positions are not complementary.

Table 2. Reranking test results using various neural network models.

Setup	Overall
Baseline	34.7
NNLM	35.0
NNLM (CD)	35.1
RNNLM	35.1
Bilingual RNN (BOW)	35.5
NNJM (CI)	36.2
NNJM (PD w/o shared matrix)	36.1
NNJM (PD)	36.5
NNJM (CD)	**36.7**
NNJM (PD+CD)	**36.7**

5.2. Discussion

One may argue that context-dependent linear projection actually exploits more source and target words. With a projection window of size three, two extra contextual words were used at the boundary positions. The source of the gain, whether the extra contextual words or the context-dependent projection, was unclear. To understand this better, we trained NNJM with 13 contextual words on the source side and 5 contextual words on the target side; context-independent projection matrices were still employed. Table 3 shows that although the number of parameters was increased after enlarging the context sizes, this only brought a marginal gain of 0.1 BLEU. Comparing the number of model parameters, a 3-window context-dependent projection matrix has $O(|V||Y|)$ additional parameters, where Y is the dense vector after linear projection, i.e., $|Y| = 192$ throughout our experiments. $|V|$ is the input vocabulary size. On the other hand, NNJM with 13 contextual words on the source side and 5 contextual words on the target side has $O(|H||Y|)$ additional parameters, where $|H| = 600$ hidden nodes. Since $|V| >> |H|$, NNJM with context-dependent linear projection has more parameters to fit the training data. Increasing the number of parameters in this way was more fruitful.

Table 3. BLEU performance of NNJM with different numbers of contextual words.

Setup	# Source context	# Target context	Overall
NNJM (CI)	11	3	36.2
NNJM (CI)	13	5	36.3
NNJM (CD)	11	3	**36.7**

Another question is the effect of the size of the projection window on MT performance. Table 4 shows the BLEU results. Increasing the projection window size from 1 to 3 on the source side yielded 0.2 BLEU improvement. Applying context-dependent projections on both sides yielded additive improvement. On the other hand, further increasing the window sizes did not improve performance: a degradation was observed with window size of 5. This may be due to the significant increase in the number of model parameters and there were insufficient training data.

Table 4. BLEU performance of NNJM with different projection window sizes.

Setup	Source window	Target window	Overall
NNJM (CI)	1	1	36.2
NNJM (CD)	3	1	36.4
NNJM (CD)	3	3	**36.7**
NNJM (CD)	3	4	**36.7**
NNJM (CD)	5	5	36.3

6. CONCLUSIONS

In this paper, we have presented two approaches of linear projections of sparse inputs with an application to neural network joint modeling for statistical machine translation. The results showed that both context-dependent and position-dependent linear projections yielded consistent improvement in machine translation performance as compared to context-independent projections. In the future, we will investigate using rich input representation such as part-of-speech tags and discourse context for NNJM.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0016. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for Public Release, Distribution Unlimited.

8. REFERENCES

[1] H. Schwenk, A. Rousseau, and M. Attik, "Large, pruned or continuous space language models on a GPU for statistical machine translation," in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Montréal, Canada, June 2012, pp. 11–19, Association for Computational Linguistics.

[2] H. Schwenk, "Continuous-space language models for statistical machine translation," in *The Prague Bulletin of Mathematical Linguistics*, 2010, pp. (93): 137–146.

[3] H. S. Le, A. Allauzen, and F. Yvon, "Continuous space translation models with neural networks.," in *HLT-NAACL*, 2012, pp. 39–48.

[4] H. S. Le, T. Lavergne, A. Allauzen, M. Apidianaki, L. Gong, A. Max, A. Sokolov, G. Wisniewski, and F. Yvon, "LIMSI @ wmt12.," in *WMT*, 2012, pp. 330–337.

[5] J. Gao, X. He, W. Yih, and L. Deng, "Learning continuous phrase representations for translation modeling," in *ACL*, 2014.

[6] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *ACL*, 2014.

[7] T. Mikolov, K. Martin, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, 2010, pp. 1045–1048.

[8] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[9] P. F. Brown, V. J. D Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based N-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[10] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *ICML*, 2008.

[11] P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling," in *ASRU*, 2013.

[12] N. F. Ayan, A. Mandal, M. Frandsen, J. Zheng, A. Kathol, F. Bechet, B. Favre, A. Marin, T. Kwiatkowski, M. Ostendorf, L. Zettlemoyer, P. Salletmayr, J. Hirschberg, and S. Stoyanchev, "Can you give me another word for hyperbaric?: Improving speech translation using targeted clarification questions," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vancouver, June 2013.

[13] D. Chiang, "Hierarchical phrase-based translation," in *Computational Linguistics*, 2007, vol. 33(2).

[14] D. Chiang, K. Knight, and W. Wang, "11,001 new features for statistical machine translation," in *Proc. NAACL–HLT 2009*, 2009, pp. 218–226.

[15] M. Hopkins and J. May, "Tuning as ranking," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK., July 2011, pp. 1352–1362, Association for Computational Linguistics.

[16] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, July 2002, pp. 311–318, Association for Computational Linguistics.

[17] B. Zhao and Y. C. Tam, "Bilingual recurrent neural networks for improved statistical machine translation," in *SLT*, 2014.

[18] M. Auli, M. Galley, C. Quirk, and G. Zweig, "Joint language and translation modeling with recurrent neural networks," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, October 2013, pp. 1044–1054, Association for Computational Linguistics.

[19] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.