Mandarin Word-Character Hybrid-Input Neural Network Language Model.

Conference Paper · January 2011

Source: DBLP

CITATIONS
10

READS
235

3 authors, including:

Tim Ng
Raytheon BBN Technologies
28 PUBLICATIONS **425** CITATIONS
SEE PROFILE

Long Nguyen
Hochschule Neubrandenburg
71 PUBLICATIONS **1,515** CITATIONS
SEE PROFILE

Mandarin word-character hybrid-input Neural Network Language Model

Moonyoung Kang, Tim Ng, Long Nguyen

Raytheon BBN Technologies,
10 Moulton Street, Cambridge, MA 02138, USA
{mkang, tng, ln}@bbn.com

Abstract

We applied neural network language model (NNLM) on Chinese by training and testing it on 2011 GALE Mandarin evaluation task. Exploiting the fact that there are no word boundaries in written Chinese, we trained various NNLMs using either word, character, or both, including a word-character hybrid-input NNLM which accepts both word and character as input. Our best result showed up to 0.6% absolute (6.3% relative) Character Error Rate (CER) reduction compared to an un-pruned 4-gram standard language model and 0.2% absolute (2.6% relative) CER reduction compared to a word-based NNLM.

Index Terms: speech recognition, neural network language model

1. Introduction

Neural Network Language Model (NNLM) has gained popularity in recent years among Large Vocabulary Continuous Speech Recognition (LVCSR) systems at the n-best rescoring stage [1]. NNLM has two advantages compared to standard n-gram LM: ability to estimate probabilities of unseen examples without back-off and tolerance to using longer context. For unseen examples, unlike standard n-gram LM which needs to back-off to lower order, NNLM projects words onto continuous space and estimates probability by using nearby examples on the projected space [2]. In addition, while standard n-gram LM suffers from exponential growth of size, serious data fragmentation, and increased miss rate by using longer context, NNLM does not suffer any data segmentation and grows linearly with respect to the increase of context size [3],[4]. Based on the above two advantages, researchers used NNLM to train a language model longer than 4-gram: pragmatic maximum with standard n-gram LM.

In this paper, we applied NNLM on Mandarin and tried to extend the above advantages in a way other than increasing the size of context: merging the effect of word and character NNLM using linear interpolation and the word-character hybrid-input NNLM.

Unlike English, Chinese words are not delimited by white space and the LVCSR system must introduce its own word segmenter to use word as basic recognition unit [5],[6]. However, this can introduce additional error to decoding system, and we tried to minimize this error in various ways: train a word-character hybrid-input NNLM or interpolate word NNLM and character NNLM.

A word-character hybrid-input NNLM is based on the fact that NNLM has no restriction on its input type. We tried to utilize this by providing more than one type of input onto NNLM. Each input type, word or character, is projected independently onto different projection layer and the hybrid-input NNLM integrate the projected information at the following hidden and output layers. Meanwhile, interpolating word NNLM and character NNLM works by linearly interpolating word probabilities from two separately trained NNLMs and the standard n-gram LM. This will show whether information NNLMs acquired from each NNLM can be integrated using linear interpolation.

The systems are evaluated on five test sets: dev09sub (3.0 hours), eval09 (2.6 hours), dev10c (5.8 hours), dev10r (12.1 hours) and dev10d (12.2 hours). dev10c, dev10r, and dev10d are the carefully transcribed, regular and difficult development sets for GALE Phase 5 (P5) evaluation. With NNLM, we could achieve 0.3~0.6% absolute (1.9~6.3% relative) CER reduction compared to a standard 4-gram LM and up to 0.2% absolute (2.6% relative) compared to a word-based NNLM. It is surprising that by applying NNLM at n-best rescoring, the last stage of decoding, we could get big improvements over our state-of-the-art baseline system, which uses state-of-the-art acoustic model techniques and a large un-pruned word-based 4-gram language model.

A few papers used similar approach to ours. [7] tried to merge the result of Mandarin word and character language model through methods such as ROVER [8], linear interpolation, and log-linear interpolation. Even though started from a similar idea, our paper differs from theirs in that we merged models after applying NNLM, a much stronger language model than standard n-gram language model, which might decrease the improvements that [7] obtained. Also, we tried a new method, a hybrid-input NNLM which is possible only from NNLM.

[9] is similar to hybrid-input NNLM in that they use inputs other than words to train NNLM: syntactic features. However, our approach is different from [9] in the following two criteria: structure of NNLM and data used in NNLM.

Unlike our approach where different input type has different vocabulary and projection layer, they treated words and syntactic features as a single input type and created a unified vocabulary and projection layer. Projecting different types onto identical continuous space lacks generality compared to our approach and can cause unnecessarily correlation between inputs. Also, unlike hybrid-input NNLM which extracts good gain only from the given text without additional tool or information, [9] requires a good parser, which is not easy to obtain, and uses additional syntactic information which is beyond the plain text.

In this sense, our paper is unique in that we applied NNLM on Mandarin, tried to merge the effect of character and word NNLM, presented a new way of handling multiple types of input, and showed a good gain using only the information from the raw text without additional information.

The rest of the paper is structured as follows. Section 2 describes the conventional use of NNLM and changes we made to implement hybrid-input NNLM. Section 3 describes

experimental setup and result and Section 4 holds conclusion which is followed by Acknowledgements and References.

2. Neural Network Language Model

2.1. Architecture of conventional NNLM

As in Figure 1, a conventional Neural Network Language Model consists of four layers: input layer, projection layer, hidden layer, and output layer. Inputs to the network are the indices of n-1 history words using the input vocabulary of size N. Each word index is then projected onto P dimensional projection layer, representing words projected on continuous space, using NxP projection matrix which is shared by all the input words. Projection, hidden, and output layer comprises a fully connected neural network with hyperbolic tangent function applied on the hidden layer and softmax normalization applied on the output layer [2]. To use NNLM output as posterior probability, output layer size should be equal to vocabulary size N.

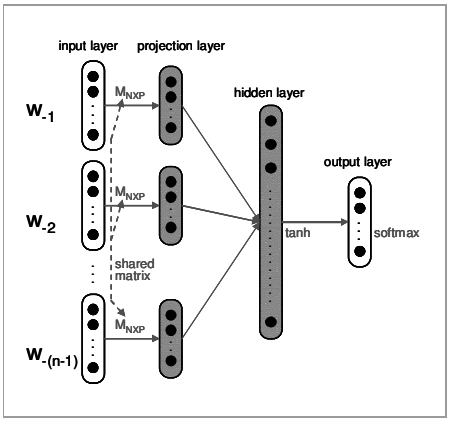

Figure 1: *Schematic diagram of a Neural Network Language Model.*

However, when N is too large, an inputlist and a shortlist replaces vocabulary in the input and output layer to reduce computational complexity [10]. Inputlist and shortlist is built by using L (<N) and M (<N) most frequent words in the vocabulary and NNLM probability is computed only when all the context words and target word belong to inputlist and shortlist. For direct usage of output value as probability, an out-of-shortlist (OOS) node is added to NNLM to represent the sum of probabilities of all the OOS words [1].

For better performance, the probability calculated through NNLM is then interpolated with standard language model by using linear interpolation with weights calculated through Expectation Maximization using a held-out set. For n-gram instances where NNLM probability cannot be calculated, standard n-gram probability is used instead.

Due to the high computational requirement of NNLM, NNLM is used only at the n-best rescoring stage of the decoding sequence. In all other places where a language model is required during decoding, such as lattice rescoring, a standard n-gram language model is used instead.

To train an NNLM, weights in projection matrix and the fully connected neural network are randomly initialized and trained using neural network back-propagation algorithm to minimize held-out data perplexity with weight decay regularization [2].

2.2. Word-character hybrid-input Mandarin NNLM

Due to the fact that the words are not delimited by white space in Chinese text, the word segmentation in training and testing data might not be optimal for language modeling. Exploiting the fact that there is no restriction on input type of NNLM, we tried using both character and word as input.

As in Figure 2, the input of hybrid-input NNLM consists of two parts: word history and character history. Word and character have their own history length: m and l. In case of word history, previous (m-1) words are used as the input, as in a typical n-gram. The words are converted into word index using word vocabulary. For character history input, previous (l-1) characters excluding white space are used for NNLM input. Characters are converted into character index integer value using a character vocabulary which is different from word vocabulary. By concatenating these two value sets, total of (m+l-2) values are used as input of m-word l-character hybrid-input NNLM.

Unlike a traditional NNLM where all the inputs share a single projection matrix, word-character hybrid-input NNLM has a projection matrix for each input type. Projection layers of the word and character differ, so that each input type can be projected onto different continuous space. The schema of hybrid-input NNLM after projection layer is identical to the traditional NNLM: the projection, hidden, and output layer make up a fully connected neural network. Word is the only output type in the hybrid-input NNLM, because if we use character as the output type it is not clear how to define the previous word of the target character when the target character is not the first character in a word. Shortlist is used in the output layer as in conventional word-based NNLM.

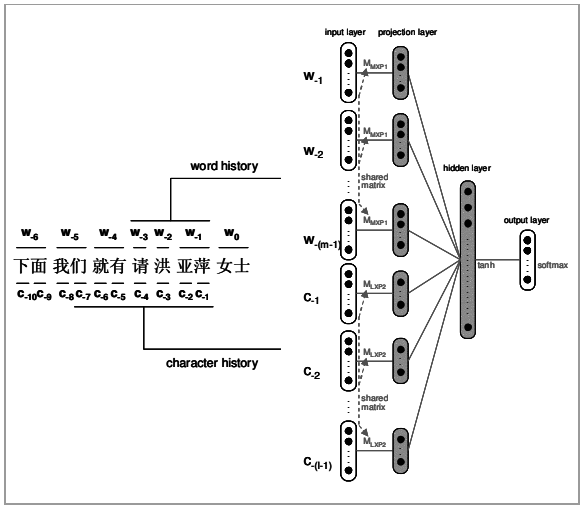

Figure 2: *Schematic diagram of hybrid-input Neural Network Language Model (4-word 8-character hybrid-input forward-NNLM)*

An alternative way of handling two types of input is to train a separate NNLM for each input type and linearly interpolate the NNLMs. However, hybrid-input NNLM is faster than linearly interpolated NNLM by two-fold in training and evaluation and this is critical since NNLM suffers from long training and evaluation time [2].

The speed of NNLM mostly depends on the size of hidden and output layer. By increasing only the size of input layer but keeping the size of hidden and output layer identical, the hybrid-input NNLM can run nbest rescoring at almost the same speed as word-based NNLM. However, linearly interpolated NNLM takes twice the time of word-based NNLM since the data has to go through the NNLM nbest rescoring process twice.

Theoretically, the word-character hybrid-input NNLM is based on the following mathematical deduction. A backward language model is defined as follows using chain rule without any loss of information:

$$P(S) = \prod_{i=1}^{n} P(w_i \mid w_{i+1}^{n}) \tag{1}$$

when S is the whole sentence, n is the number of words in S and w_i^j is all the words between i^{th} and j^{th} word in S. Let $c \in w_i^j$ be all the characters in w_i^j, the Equation (1) becomes

$$P(S) = \prod_{i=1}^{n} P(w_i \mid w_{i+1}^{n}, c \in w_{i+1}^{n}) \tag{2}$$

since $c \in w_{i+1}^{n}$ is redundant information given w_{i+1}^{n}.

By applying m^{th} order and l^{th} order Markov assumption to word and character separately, we get the following equation,

$$P(S) = \prod_{i=1}^{n} P(w_i \mid w_{i+1}^{i+m}, c_{j+1}^{j+l}) \tag{3}$$

where c_{j+1}^{j+l} are the first l characters in w_{i+1}^{i+m}. You can see that the part inside Π multiplication is the probability calculation of word-character hybrid-input backward-NNLM. For case when we back-off to 4-gram standard LM, you can imagine that Markov assumption of m=4 and l=0 is applied.

3. Experiments

3.1. NNLM settings

Our NNLM system is implemented based on the open source code released by Holger Schwenk with some modification in order to support additional functionalities such as output shortlist and hybrid input [11].

Our NNLMs project input words onto the projection layer of 400 nodes and characters onto the layer of 200 nodes. Projection layers are followed by the hidden layer of 500 nodes. 6K Mandarin character dictionary and 65K word dictionary were used as input vocabularies. The list of 20K most frequent words from the word vocabulary was used as the shortlist of output layer. Word and character frequencies were calculated using the whole GALE LM training data. The word and the character dictionary of standard n-gram LM were used as the input vocabularies and inputlists, generating no OOI word. Since our standard n-gram is a word-based backward n-gram LM, regardless of the input type, word, character, or hybrid, we trained our NNLMs to estimate the probability of word to make the interpolation with the standard n-gram feasible.

Among the 2.3 billion words of GALE LM training data, 20M words of Acoustic Model (AM) training transcript is used to train NNLM. Training stops when log-likelihood of validation set (GALE dev08, bnat06, and bcat06 Mandarin data) starts to decrease. Training was done in batch mode with the batch size of 128 on 8-core x86_64 machine using Intel Math Kernel Library, [12], using the identical method as in [2]. For other NNLM coefficients, we used initial learning rate of $5*10^{-3}$, learning rate multiple of $4*10^{-7}$, and weight decay of $3*10^{-5}$ [2]. During linear interpolation with 4-gram standard LM, the NNLMs got the weight between 0.4 and 0.6.

3.2. Decoding experiment

We trained four different NNLMs: 6-word, 12-character, 6-word 12-character hybrid-input, and 8-word NNLM. First, we trained 6-gram word NNLM based on the result of [10] that no big improvement was observed after 6-gram. 12-character NNLM is trained to counter 6-word NNLM based on the following reasons: 2-character word is common in Mandarin and 12-character NNLM showed comparable result to 6-word NNLM. Combining the settings of 6-word NNLM and 12-character NNLM, we trained a 6-word 12-character hybrid-input NNLM. We also trained an 8-word NNLM for fair comparison of word based NNLM and hybrid-input NNLM, since 8-word NNLM is expected to have similar context length as 6-word 12-character hybrid-input NNLM from the analysis that our training data has 1.7 characters per word.

NNLM decoding is done by doing n-best rescoring over the baseline decoding result generated from BBN's best DARPA's GALE evaluation Mandarin system of January 2011 which used 4-gram standard language model. As mentioned in Section 2, all the NNLMs were interpolated with 4-gram standard LM to estimate the probability of the target word, and this fact holds in the following section even if not mentioned explicitly.

We did total of 6 decodings from the above NNLMs: standard 4-gram LM decoding (baseline), 6-word NNLM decoding, 12-character NNLM decoding, 8-word NNLM decoding, 6-word NNLM and 12-character NNLM decoding, and 6-word 12-character hybrid-input NNLM decoding. The 6-word NNLM and 12-character NNLM decoding is done by interpolating the probability estimates of three language models: 4-gram standard LM, 6-word NNLM and 12-character NNLM, with interpolation weights generated by doing an EM optimization with all three.

The acoustic features, generated using Regional Dependent Transform to jointly optimize the feature augmentation of the Perceptual Linear Predictive and Multi Layer Perceptron features under a discriminative training criterion, are used to train our acoustic model on 1900 hours of GALE transcribed audio data [13]. Our standard word-based 4-gram LM is trained from training data of total 2.3 billion words and 65K word dictionary by interpolating 18 un-pruned language models trained on each data source. For acoustic model and language model score combination, we used GALE P5 dev10c dataset to train the combination weights.

3.3. Result

As mentioned in Section 1, decoding was done on five GALE P5 test sets: dev09sub, eval09, dev10c, dev10r, and dev10d. Table 1 shows the CER result of all systems on each test set.

Comparing line 1 and 6, we can see that the word-character hybrid NNLM, the best model from Table 1, shows 0.3~0.6% absolute (1.9~6.3% relative) CER reduction compared to baseline system. This is a pretty big gain

considering that hybrid NNLM is trained on only 20 million words while standard 4-gram LM is trained on 2.3 billion words, hybrid NNLM is applied only at the n-best rescoring stage, and our baseline is a state-of-the-art decoding system with sophisticated acoustic model.

Table 1. *CER of systems on each devset*

System	dev09 sub	eval09s	dev10c	dev10r	dev10d
baseline	8.4	7.9	13.3	8.5	25.9
6-word NNLM	8.1	7.6	12.9	8.3	25.5
12-char NNLM	8.2	7.5	12.8	8.2	25.5
8-word NNLM	8.1	7.5	12.8	8.2	25.5
6-word NNLM + 12-char NNLM	8.1	7.4	12.7	8.2	25.4
6-word 12-char hybrid-input NNLM	8.1	7.4	12.7	8.2	25.4

From line 2, 3 ,4 and 5, 6, we can infer that word-character combined NNLMs outperform single-unit NNLMs by absolute WER reduction of 0.0~0.2%. The gain is not huge but it is consistent across different devsets. Given that NNLMs in line 3, 4, 5, and 6 uses similar length of context on average, this shows that information from words and characters can add up.

At line 5 and 6, hybrid-input NNLM showed identical result to the linearly interpolated decoding of 6-word NNLM and the 12-character NNLM meaning that hybrid-input NNLM has similar power as the linear interpolation of the two NNLMs. The interpolation weights at Table 2 and 3 backs the decoding results since interpolation weight is related to the performance of the NNLM and the interpolation weight of hybrid-input NNLM is similar to the sum of 6-word NNLM and 12-character NNLM interpolation weights.

Table 2. *Interpolation weight of 4-gram LM and hybrid-input NNLM*

System	Interpolation weight
4-gram standard LM	0.538
hybrid-input NNLM	0.461

Table 3. *Interpolation weight of 4-gram LM, 6-word NNLM, and 12-character NNLM*

System	Interpolation weight
4-gram standard LM	0.523
6-word NNLM	0.345
12-character NNLM	0.132

4. Conclusions

We could obtain up to 0.3~0.6% absolute (1.9~6.3% relative) CER reduction compared to standard 4-gram LM by applying an NNLM using both word and character information onto state-of-the-art Mandarin LVCSR system. By combining information from words and characters through linear interpolation or constructing a hybrid-input NNLM, we could get more improvement consistently over all devsets compared to a single-input NNLM. It is a pity that hybrid input NNLM showed no additional power over a linear interpolation model of two separate NNLMs, however the hybrid-input NNLM still holds computational superiority compared to linearly interpolated models as mentioned in section 2.

Based on the result, in the future we are trying to input more information into hybrid-input NNLM, especially information that cannot be implemented through linear interpolation. The first thing in plan is to input standard N-gram probability into the NNLM which we expect to minimize the internal randomness that an NNLM has from the random weight initialization at the beginning of the NNLM train [14].

5. Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0022. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or its Contracting Agent, the U.S. Department of the Interior.

6. References

[1] Park, J., Liu, X., Gales, M.J.F., and Woodland, P.C., "Improved neural network based language modelling and adaptation", in International Conference on Spoken Language Processing, Interspeech 2010, 26-30 Sep 2010, Makuhari, Japan.

[2] Schwenk, H., "Continuous space language models", in Computer Speech and Language Volume 21, Issue 3, 492-518, Academic Press Ltd., July 2007.

[3] Bengio, Y., Ducharme, R., and Vincent, P., "A neural probabilistic language model", in Advances in Neural Information Processing Systems, 2001.

[4] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C., " A neural probabilistic language model", Journal of Machine Learning Research, vol.3, 2003.

[5] Sproat, R., Shih, C., Gale, W., and Chang, N., "A Stocastic Finite-State Word-Segmentation Algorithm for Chinese", Computational Lingustics, 22(3):218-228,1996.

[6] Wu, D. and Fung, P., "Improving Chinese tokenization with linguistic filters on statistical lexical acquisition", Proceedings of the Fourth Conference on Applied Natural Language Processing, 180-181, October, 1994, Stuttgart.

[7] Hieronymus, J.L., Liu, X., Gales, M.J.F., Woodland, P.C., "Exploiting Chinese Character Models to Improve Speech Recognition Performance", in International Conference on Spoken Language Processing, Interspeech 2009, 6-10 Sep 2009, Brighton, UK.

[8] Fiscus., J.G. "A Post-Processing System to Yield Reduced Word Error Rates: Recogniser Output Voting Error Reduction(ROVER)", in Proceedings of IEEE Automatic Speech Recognition & Understanding '97

[9] Kuo, H.-K.J., Mangu, L., Emami, A., Zitouni, I., Y.-S. Lee, "Syntactic Features for Arabic Speech Recognition", in Automatic Speech Recognition & Understanding, 2009, 327-332, 2009.

[10] Emami, A., Mangu, L., "Empirical Study of Neural Network Language Models for Arabic Speech Recognition", in Automatic Speech Recognition & Understanding 2007, 147-152, Dec. 2007.

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

[12] Intel Math Kernal Library. Online: http://software.intel.com/en-us/articles/intel-mkl/, accessed on 23 Mar 2011

[13] Ng, T., Zhang, B., and Nguyen, L., "Jointly Optimized Discriminative Features for Speech Recognition", in International Conference on Spoken Language Processing, Interspeech 2010, 2618-2621, 26-30 Sep 2010, Makuhari, Japan.

[14] Son, L.H., Allauzen, A., Wisniewski, G., and Yvon, F., "Training continuous space language models: some practical issues" in Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, 778-788, 2010