NAVER Machine Translation System for WAT 2015

Hyoung-Gyu Lee[1], Jaesong Lee[1], Jun-Seok Kim[1] and Chang-Ki Lee[2]
[1]NAVER LABS, NAVER Corp., Seongnam-si, Gyeonggi-do, South Korea
[2]Kangwon National University, Chuncheon-si, Gangwon-do, South Korea
{hg.lee,jaesong.lee,jun.seok}@navercorp.com
leeck@kangwon.ac.kr

Abstract

In this paper, we describe NAVER machine translation system for English to Japanese and Korean to Japanese tasks at WAT 2015. We combine the traditional SMT and neural MT in both tasks.

1 Introduction

This paper explains the NAVER machine translation system for the 2nd Workshop on Asian Translation (WAT 2015) (Nakazawa et al., 2015). We participate in two tasks; English to Japanese (En-Ja) and Korean to Japanese (Ko-Ja).

Our system is a combined system of traditional statistical machine translation (SMT) and neural machine translation (NMT). We adopt the tree-to-string syntax-based model as En-Ja SMT baseline, while we adopt the phrase-based model as Ko-Ja. We propose improved SMT systems for each task and an NMT model based on the architecture using recurrent neural network (RNN) (Cho et al., 2014; Sutskever et al., 2014).

We give detailed explanations of each SMT system in section 2 and section 3. We describe our NMT model in section 4.

2 English to Japanese

2.1 Training data

We used 1 million sentence pairs that are contained in `train-1.txt` of ASPEC-JE corpus for training the translation rule tables and NMT models. We also used 3 million Japanese sentences that are contained in `train-1.txt`, `train-2.txt`,`train-3.txt` of ASPEC-JE corpus for training the 5-gram language model. We also used 1,790 sentence pairs of `dev.txt` for tuning the weights of each feature of SMT linear model and as validation data of neural network. We filtered out the sentences that have 100 or more tokens from training data.

2.2 Language Analyzer

We used Moses tokenizer and Berkeley constituency parser[1] for tokenizing and parsing an English sentence. We used our own Japanese tokenizer and part-of-speech tagger for tokenizing and tagging a Japanese sentence. After running the tokenizer and the tagger, we make a token from concatenation of a word and its part-of-speech.

2.3 Tree-to-string Syntax-based SMT

To determining the baseline model, we first performed comparative experiments with the phrase-based, hierarchical phrase-based and syntax-based models. As a result, we chose the tree-to-string syntax-based model.

The SMT models that consider source syntax such as tree-to-string and forest-to-string brought out better performance than the phrase-based and hierarchical phrase-based models in the WAT 2014 En-Ja task.

The tree-to-string model was proposed by Huang (2006) and Liu (2006). It utilizes the constituency tree of source language to extract translation rules and decode a target sentence. The translation rules are extracted from a source-parsed and word-aligned corpus in the training step. We use synchronous context free grammar (SCFG) rules.

In addition, we used a rule augmentation method which is known as syntax-augmented machine translation (Zollmann and Venugopal, 2006). Because the tree-to-string SMT makes some constraints on extracting rules by considering syntactic tree structures, it usually extracts fewer rules than hierarchical phrase-based SMT (HPBSMT) (Chiang, 2005). Thus it is required to augment tree-to-string translation rules. The rule augmentation method allows the training system to extract more rules by modifying parse trees. Given a parse tree, we produce additional nodes

[1]`https://github.com/slavpetrov/berkeleyparser`

Proceedings of the 2nd Workshop on Asian Translation (WAT2015), pages 69–73,
Kyoto, Japan, 16th October 2015.

by combining any pairs of neighboring nodes, not only children nodes, e.g. NP+VP. We limit the maximum span of each rule to 40 tokens in the rule extraction process. The tree-to-string decoder use a chart parsing algorithm with cube pruning proposed by Chiang (2005).

Our En-Ja SMT system was developed by using the open source SMT engines; Moses and Giza++. Its other specifications are as follows:

- Grow-diag-final-and word alignment heuristic
- Good-Turing discounting for smoothing probabilities
- Minimum Error Rate Training (MERT) for tuning feature weights
- Cube-pruning-pop-limit = 3000

2.4 Handling Out-of-Vocabulary

In order to handle out-of-vocabulary (OOV) words, we use two techniques; hyphen word split and spell error correction. The former is to split a word with hyphen (-) to two separate tokens before running the language analyzer. The latter is to automatically detect and correct spell errors in an input sentence. We give a detailed description of spell error correction in section 2.4.1.

2.4.1 English Spell Correction

It is not easy to translate a word including errata, because the erroneous word has only a slim chance of occurrence in the training data. We discovered a lot of spell errors among OOV words that appear in English scientific text. We introduce English spell correction for reducing OOV words in input sentences. We developed our spell corrector by using Aspell[2].

For detecting a spell error, we skip words that have only capitals, numbers or symbols, because they are likely to be abbreviations or mathematic expressions. Then we regard words detected by Aspell as spell error words.

For correcting spell error, we use only top-3 suggestion words from Aspell. We find that a large gap between an original word and its suggestion word makes wrong correction. To avoid excessive correction, we introduce a gap thresholding technique, that ignores the suggestion word that has 3 or longer edit distance and selects one that has the shortest edit distance between an original word and its suggestion word.

[2] http://aspell.net/

3 Korean to Japanese

3.1 Training data

We used 1 million sentence pairs that are contained in JPO corpus for training phrase tables and NMT models. We also used Japanese part of the corpus for training the 5-gram language model. We also used 2,000 sentence pairs of `dev.txt` for tuning the weights of each feature of SMT linear model and as validation data of neural network. We did not filter out any sentences.

3.2 Language Analyzer

We used MeCab-ko[3] for tokenizing a Korean sentence. We used Juman[4] for tokenizing a Japanese sentence. We did not perform part-of-speech tagging for both languages.

3.3 Phrase-based SMT

As in the En-Ja task, we first performed comparative experiments with the phrase-based and hierarchical phrase-based models, and then adopt the phrase-based model as our baseline model.

For the Ko-Ja task, we develop two phrase-based systems; word-level and character-level. We use word-level tokenization for the word-based system. We found that setting the distortion limit to zero yields better translation in aspect of both BLEU and human evaluation. We use the 5-gram language model. We use character-level tokenization for character-based system. We use the 10-gram language model and set the maximum phrase length to 10 in the phrase pair extraction process.

We found that the character-level system does not suffer from tokenization error and out-of-vocabulary issue. The JPO corpus contains many technical terms and loanwords like chemical compound names, which are more inaccurately tokenized and allow a lot of out-of-vocabulary tokens to be generated. Since Korean and Japanese share similar transliteration rules for loanwords, the character-level system can learn translation of unseen technical words. It generally produces better translations than a table-based transliteration.

[3] https://bitbucket.org/eunjeon/mecab-ko/

[4] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

Moreover, we tested jamo-level tokenization[5] for Korean text, however, the preliminary test did not produce effective results.

We also investigated a parentheses imbalance problem. We solved the problem by filtering out parentheses-imbalanced translations from the n-best results. We found that the post-processing step can improve the BLEU score with low order language models, but cannot do with high order language models. We do not use the step for final submission.

To boosting the performance, we combine the word-level phrase-based model (Word PB) and the character-level phrase-based model (Char PB). If there are one or more OOV words in an input sentence, our translator choose the Char PB model, otherwise, the Word PB model.

Our Ko-Ja SMT system was developed by using the open source SMT engines; Moses and Giza++. Its other specifications are as follows:

- Grow-diag-final-and word alignment heuristic
- Good-Turing discounting for smoothing probabilities
- Minimum Error Rate Training (MERT) for tuning feature weights

4 Neural Machine Translation

Neural machine translation (NMT) is a new approach to machine translation that has shown promising results compared to the existing approaches such as phrase-based statistical machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). An NMT system is a single neural network that reads a source sentence and generates its translation. Using the bilingual corpus, the whole neural network is jointly trained to maximize the conditional probability of a correct translation given a source sentence. NMT has several advantages over the existing statistical machine translation systems such as the phrase-based system. First, NMT uses minimal domain knowledge. Second, the NMT system is jointly trained to maximize the translation performance, unlike the existing phrase-based system which consists of many separately trained features. Third, the NMT system removes the need to store explicit phrase tables and language models. Lastly, the decoder of an NMT system is easy to implement. Despite these advantages and promising results, NMT has a limitation in handling a larger target vocabulary, as the complexity of training and decoding increases proportionally to the number of target words.

In this paper, we propose a new approach to avoid the large target vocabulary problem by preprocessing the target word sequences, encoding them as a longer character sequence drawn from a small character vocabulary. The proposed approach removes the need to replace rare words with the unknown word symbol. Our approach is simpler than other methods recently proposed to address the same issue (Luong et al., 2015; Jean et al., 2015).

4.1 Model

In this paper, we use our in-house software of NMT that uses an attention mechanism, as recently proposed by Bahdanau et al. (2015). The encoder of NMT is a bi-directional recurrent neural network such that

$$h_t = [\overleftarrow{h}_t; \vec{h}_t] \tag{1}$$

$$\overleftarrow{h}_t = f_{GRU}(W_{s\_we}x_t, \overleftarrow{h}_{t+1}) \tag{2}$$

$$\vec{h}_t = f_{GRU}(W_{s\_we}x_t, \vec{h}_{t-1}) \tag{3}$$

where h_t is a hidden state of the encoder, x_t is a one-hot encoded vector indicating one of the words in the source vocabulary, W_{s_we} is a weight matrix for the word embedding of the source language, and f_{GRU} is a gated recurrent unit (GRU) (Cho et al., 2014).

At each time, the decoder of NMT computes the context vector c_t as a convex combination of the hidden states (h_1,...,h_T) with the alignment weights α_1,...,α_T:

$$c^t = \sum_{i=1}^{T} \alpha_{ti} h_i \tag{4}$$

$$\alpha_{ti} = \frac{\exp(e_{tj})}{\sum_{j=1}^{T} \exp(e_{tj})} \tag{5}$$

$$e_{ti} = f_{FFNN}(z_{t-1}, h_i, y_{t-1}) \tag{6}$$

where f_{FFNN} is a feedforward neural network with a single hidden layer, z_{t-1} is a previous hidden state of the decoder, and y_{t-1} is a previous generated target word (one-hot encoded vector).

[5]Jamo is Korean alphabet letters which represent consonants and vowels. A Korean character can be usually separated to three jamoes; choseong, jungseong and jongseong.

A new hidden state z_t of the decoder which uses GRU is computed based on z_{t-1}, y_{t-1}, and c_t:

$$z_t = f_{GRU}(y_{t-1}, z_{t-1}, c_t) \quad (7)$$

The probability of the next target word y_t is then computed by

$$\begin{aligned} p(y_t|y_{<t}, x) = y_t^T f_{softmax}\{W_{z'y} z'_t \\ + W_{zy} z_t + W_{cy} c_t \\ + W_{yy}(W_{t\_we} y_{t-1}) + b_y\} \end{aligned} \quad (8)$$

$$z'_t = f_{ReLU}(W_{zz'} z_t) \quad (9)$$

where $f_{softmax}$ is a softmax function, f_{ReLU} is a rectified linear unit (ReLU), W_{t_we} is a weight matrix for the word embedding of the target language, and b_y is a target word bias.

4.2 Settings

We constructed the source word vocabulary with the most common words in the source language corpora. For the target character vocabulary, we used a BI (begin/inside) representation (e.g., 結/B, 果/I), because it gave better accuracy in preliminary experiment. The sizes of the source vocabularies for English and Korean were 245K and 60K, respectively, for the En-Ja and Ko-Ja tasks. The sizes of the target character vocabularies for Japanese were 6K and 5K, respectively, for the En-Ja and Ko-Ja tasks. We chose the dimensionality of the source word embedding and the target character embedding to be 200, and chose the size of the recurrent units to be 1,000. Each model was optimized using stochastic gradient descent (SGD). We did not use dropout. Training was early-stopped to maximize the performance on the development set measured by BLEU.

5 Experimental Results

All scores of this section are reported in experiments on the official test data; `test.txt` of the ASPEC-JE corpus.

5.1 En-Ja SMT

Table 1 shows the evaluation results of our En-Ja traditional SMT system. The first row in the table indicates the baseline of the tree-to-string systax-based model. The second row shows the system that reflects the tree modification described in section 2.3. The augmentation method drastically increased both the number of rules and the BLEU score. Our OOV handling methods described in section 2.4 also caused a positive effect as shown in the last row of the table.

SYS	BLEU	#Rules
Tree-to-string SB	31.34	250M
+ Rule augmentation	32.48	1950M
+ Parameter modification	32.63	1950M
+ OOV handling	32.76	1950M

Table 1: En-Ja SMT.

SYS	BLEU	#Rules
Word PB	70.36	57M
Char PB	70.31	55M
Word PB + Char PB	70.91	57M & 55M

Table 2: Ko-Ja SMT.

The decoding time of the rule-augmented tree-to-string SMT is about 1.3 seconds per a sentence in our 12-core machine. Even though it is not a terrible problem, we are required to improve the decoding speed by pruning the rule table or using the incremental decoding method (Huang and Mi, 2010).

5.2 Ko-Ja SMT

Table 2 shows the evaluation results of our Ko-Ja traditional SMT system. We obtained the best result in the combination of two phrase-based SMT systems.

5.3 NMT

Table 3 shows effects of our NMT model. "Human" indicates the pairwise crowdsourcing evaluation scores provided by WAT 2015 organizers. In the table, "T2S/PBMT only" is the final T2S/PBMT systems shown in section 5.1 and section 5.2. "NMT only" is the system using only RNN encoder-decoder without any traditional SMT methods. The last row is the combined system that reranks T2S/PBMT n-best translations by NMT. Our T2S/PBMT system outputs 100,000-best translations in En-Ja and 10,000-best translations in Ko-Ja. The final output is 1-best translation selected by considering only NMT score.

NMT outperforms the traditional SMT in En-Ja, while it does not in Ko-Ja. This result means that NMT produces a strong effect in the language pair with long linguistic distance. Moreover, the reranking system achieved a great synergy of T2S/PBMT and NMT in both task, even

SYS	En-Ja		Ko-Ja	
	BLEU	**Human**	**BLEU**	**Human**
T2S/PBMT only	32.76	N/A	70.91	6.75
NMT only	33.14	48.50	65.72	N/A
T2S/PBMT + NMT reranking	34.60	53.25	71.38	14.75

Table 3: Effect of NMT.

if "NMT only" is not effective in Ko-Ja. From the human evaluation, we can be clear that our NMT model produces successful results.

6 Conclusion

This paper described NAVER machine translation system for En-Ja and Ko-Ja tasks at WAT 2015. We developed both the traditional SMT and NMT systems and integrated NMT into the traditional SMT in both tasks by reranking n-best translations of the traditional SMT. Our evaluation results showed that a combination of the NMT and traditional SMT systems outperformed two independent systems.

For the future work, we try to improve the space and time efficiency of both the tree-to-string SMT and the NMT model. We also plan to develop and evaluate the NMT system in other language pairs.

Acknowledgments

We would like to thank Kevin Cho, who has visited for the summer internship program in NAVER LABS, for his contribution for development of the English spell corrector.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th biennial conference of the Association for Machine Translation in the Americas (AMTA)*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (Coling-ACL)*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

Toshiaki Nakazawa, Hideya Mino, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2015. Overview of the 2nd workshop on asian translation. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, Kyoto, Japan, October.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Twenty-eighth Annual Conference on Neural Information Processing Systems (NIPS)*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation (WMT)*.