An Effective Diverse Decoding Scheme for Robust Synonymous Sentence Translation

Youngki Park ypark@cnue.ac.kr
Hwidong Na hwidong.na@samsung.com
Hodong Lee hodong.lee@samsung.com
Jihyun Lee jihyuns.lee@samsung.com
Inchul Song icsong@samsung.com
Samsung Advanced Institute of Technology, Samsung Electronics
Suwon, 443-742, South Korea

Abstract

Existing neural machine translation (NMT) systems often mistranslate synonymous sentences into sentences with different meanings. This problem is particularly serious when there is lack of parallel corpora. In this paper, we propose a novel diverse decoding algorithm for accurate translation of synonymous sentences in an NMT system. We observe that the modeling power of NMT models may not be fully utilized because of insufficient exploration of search space by beam search. The proposed algorithm overcomes the problem by expanding search space coverage through diverse decoding. First, it performs greedy search using an NMT model to build an initial candidate list. Then, it expands the list by performing approximate *k*NN search over translation logs to find previously translated results of similar sentences and adding them to the list. Finally, it uses the NMT model again to rescore the candidate list and returns the candidate with the best score. The experimental results show that the proposed scheme enhances the BLEU score significantly over the state-of-the-art NMT system while being much faster.

1. Introduction

One of the main barriers to building a machine translation system for commercial use is that existing approaches often mistranslate sentences with the same meaning (synonymous sentences) into sentences with different meanings. For example, we may translate an English sentence, "How can I get to Gangnam Station?", into different Korean sentences, "강남역까지 어떻게 가죠?" / kang.nam.yek.kka.ci (to Gangnam station) ettehkey (how) kacyo (go) [1]? and "강남역까지 가는 길을 알려주세요" / kang.nam.yek.kka.ci (to Gangnam station) ka.nun (going) kil.ul (way) al.lye.cu.sey.yo (let me know). Here, these Korean sentences are synonymous sentences with the same meaning. If we back-translate them into English using Google Translate, the translations are "I'll go to Gangnam?" (an incorrect translation) and "Tell us how to get to Gangnam" (a correct translation), respectively. The second column in Table 1 shows the Korean-to-English translation results of five synonymous source sentences obtained by using Google Translate. As shown in the table, Google Translate translates some synonymous sentences in slightly different forms incorrectly.

Although recent neural machine translation (NMT) systems have improved translation accuracies greatly, they still generate mistranslations for some synonymous

[1] We annotate Korean words by Yale Romanization and indicate their meanings within parentheses. A period (.) indicates the syllable boundary.

Source Sentence	**Google Translate**[2]	**DL4MT-tutorial** (Firat and Cho, 2016)
kang.nam.yek.kka.ci e.tteh.key ka.cyo ?	(X) I'll go to Gangnam?	(O) How can I go to Gangnam Station?
kang.nam.yek.kka.ci e.tteh.key ka.cyo	(X) I'll go to Gangnam	(X) "I'll go to Gangnam Station to Gangnam Station"
kang.nam.yek.kka.ci ka.nun kil.i e.tteh.key toyp.ni.kka	(X) How do I get to length Gangnam?	(X) What is going to Gangnam Station?
kang.nam.yek.kka.ci e.tteh.key ka.na.yo	(O) How do I get to Gangnam?	(O) How's going to Gangnam Station?
kang.nam.yek.kka.ci ka.nun kil.ul al.lye.cu.sey.yo	(O) Tell us how to get to Gangnam	(O) Tell me the way to Gangnam Station

Table 1. Five example synonymous source sentences and their translations

sentences as shown in the third column of Table 1, which is the translation results of a state-of-the-art NMT system. There are two main reasons for such mistranslations. First, there is lack of parallel corpora for synonymous sentences. Thus, only a limited number of forms of synonymous sentences may be seen during training. Second, standard decoding algorithms, such as beam search, do not efficiently explore the search space. For NMT models with an increased modeling power, the problem of insufficient exploration of search space is more significant. For example, when we performed beam search using an NMT model to translate Sentence A below (the answer is Sentence B'), the search returned Sentence A', which is an incorrect translation:

- *Source Sentence A:* 한국의 아침 식단에서 김이 나는 따끈한 밥 한 공기는 대체 불가한 주식으로 오랫동안 여겨져 왔다. / han.kuk.uy (Korean) a.chim (morning) sik.tan.ey.se (menu) kim.i (steam) na.nun (emitting) tta.kkun.han (hot) pap (rice) han (one) kong.ki.nun (bowl) tay.chey (substitution) pul.ka.han (disallowing) cu.sik.u.lo (main dish) o.lays.tong.an (for a long time) ye.kye.cye (considered) wass.ta (have been) .
- *Sentence A'*: Kim on the morning calm in korea in the breakfast table in korea, kim's warm, steamed air has long been shadowed for a long week-end.
- *Sentence B'*: A hot, steamy bowl of rice was long considered an irreplace-able staple in Korean breakfast.

However, when we computed the scores of Sentence A' and B' using the NMT model, the model assigned a higher score to Sentence B'. In other words, the mistranslation is caused not by the NMT model's insufficient modeling power, but the failure of consideration of Sentence B' during beam search. This problem may arise similarly for the synonymous sentences of Sentence A: some synonymous sentences, especially seen during training, may produce the correct answer as a candidate during beam search whereas others may not. The main idea of our approach is that if the

[2] https://translate.google.com/
Note the translation results were obtained on 7 July 2016.

translation results of synonymous sentences are all available, we could find the correct answer for any synonymous sentence by rescoring the results (using NMT model).

In this paper, we present a novel diverse decoding algorithm for robust synonymous sentence translation. It is designed to use the modeling power of an NMT model throughout the decoding process by expanding search space coverage. First, it performs greedy search using an NMT model to build an initial candidate list. Then, it expands the list with those candidates that might not be considered during beam search. This is done by finding synomyous sentences through approximate kNN search over translation logs and then adding previously translated results of those sentences to the list. Finally, it uses the NMT model again to rescore the candidate list and returns the candidate of the highest score as the final answer. The experimental results show that the proposed scheme increases the BLEU score significantly over the state-of-the-art NMT system while also being much faster. The rest of this paper is structured as follows. In Section 2, we review the state-of-the-art neural machine translation architecture as well as the beam search algorithm. In Section 3, we present a novel diverse decoding scheme that consists of greedy decoding, n-best list expansion by (approximate) kNN search, and n-best rescoring. In Section 4, we conduct extensive experiments to compare our approaches to beam search in a number of configurations. In Section 5, we discuss related work on diverse decoding. Finally, we conclude the paper and present future research directions in Section 6.

2. Neural Machine Translation

In order to build our baseline machine translators, we exploit the encoder-decoder architecture with attention mechanism presented by (Bahdanau et al., 2015). In addition, we apply the conditional gated recurrent unit in the hidden layer of the decoder (Firat and Cho, 2016). The encoder and decoder are formulated as follows:

Encoder Let X, Y be a source and target sentence, respectively. If X and Y have the T_x and T_y number of tokens (words, subwords, or characters, etc.), we can represent them as $(x_1, \dots, x_{T_x})$ and $(y_1, \dots, y_{T_y})$. For each source word, the encoder produces an embedding vector and feed the vector into a bidirectional recurrent neural network:

$$\vec{h}_t = \vec{\phi}\big(e_x(x_t), \vec{h}_{t-1}\big),$$
$$\overleftarrow{h}_t = \overleftarrow{\phi}\big(e_x(x_t), \overleftarrow{h}_{t+1}\big),$$
$$h_t = [\vec{h}_t; \overleftarrow{h}_t],$$

where $\vec{\phi}/\overleftarrow{\phi}$ is the activation function for the forward/backward hidden state of the encoder, and $e_x(x_t)$ is a word embedding vector of the t^{th} word. Two hidden states of the encoder $\vec{h}_t$ and $\overleftarrow{h}_t$ are concatenated as the hidden state h_t of the t^{th} word.

The context set C consists of the hidden states h_t of the encoder, and the initial hidden state of the decoder s_0 depends only on the context set C:

$$C = \{h_1, \dots, h_{T_x}\},$$
$$s_0 = f_{init}\big(f_{mean}(C)\big),$$

where $f_{mean}(\cdot)$ averages a set of vectors and $f_{init}(\cdot)$ is a nonlinear function.

Decoder A hidden state of the decoder $s_{t'}$ depends on its previous hidden state $s_{t'-1}$, its previous target word $y_{t'-1}$, and its context vector $c_{t'}$:

$$s_{t'} = \phi\big(s_{t'-1}, e_y(y_{t'-1}), c_{t'}\big),$$

where ϕ is the activation function of the decoder, and the context vector $c_{t'}$ depends on the context set C:

$$c_{t'} = f_{align}(s_{t'-1}, e_y(y_{t'-1}), C).$$

There are additional layers that calculate $p(y_{t'}|y_{<t'}, X)$ based on $s_{t'}$, $c_{t'}$ and y'_{t-1}. Here, $y_{<t'}$ is the previous target words before the time step t'. Thus the decoder can calculate the log probability of the target sentence given a source sentence as follows:

$$log\ p(Y|X) = \sum_{t'=1}^{T_y} log\ p(y_{t'}|y_{<t'}, X).$$

Beam Search. The aim of neural machine translators is to find $argmax_Y(log\ p(Y|X))$. However, the search space is so large that the exact algorithms cannot be used in most practical applications. Instead, we usually use the beam search algorithm for generating approximate results: as a first step, we set t' to 1 and find $argmax^b_{y_{t'}} p(y_{t'}|y_{<t'}, X)$, where $argmax^b$ is the b number of arguments that have the highest values. Then for each selected target word, we find the $argmax^b_{y_{t'}} p(y_{t'}|y_{<t'}, X)$ again for $t' = 2$. Since we have the b^2 number of sequences now, we prune the b number of sequences and only keep the top-b sequences that have the highest probabilities. If the end of sentence (EOS) marker is reached, we decrease b by 1 and select the sentence as a hypothesis. This process continues until all of the hypothesis meet the EOS marker ($b = 0$).

3. Robust Synonymous Sentence Translation

Although the execution time of beam search is proportional to the beam size, the algorithm is one of the most efficient algorithms when the number of hypotheses maintained is in a reasonable size (smaller than 20). However, we found that the search quality of the beam search algorithm is not good enough, especially for synonymous sentence translation tasks. Even with a large beam size, the beam search often generates translation results with different meaning for synonymous source sentences.

In this section, we propose a novel, effective diverse decoding algorithm in order to cope with this problem. Our approach consists of three main parts: first, we perform the greedy search to get the 1-best (Section 3.1). After that, we obtain the additional $(n-1)$-best list by using similar source sentences and their translation results in the log database (Section 3.2). Last, we rescore the n-best list using the original source sentence (Section 3.3).

3.1. Greedy Search

Greedy search is one variant of the beam search algorithm (beam size 1). The algorithm selects the word of a highest probability at every time step, and continues this process until the end of sentence marker is found. In other words, given a source sentence X, the greedy search selects the word $\tilde{y}_{t'}$ for each time step t':

$$\tilde{y}_{t'} = argmax_{y_{t'}}(\log p(y_{t'}|\tilde{y}_{<t'}, X).$$

After greedy search, we add the obtained result to our n-best list. Because we only obtain one translation result by greedy search, we will get additional $n-1$ results in the following subsections.

Although we can get additional candidates by using larger beam size, we prefer to use greedy search because of the two reasons: (1) greedy search is much cheaper operation than beam search. Not only is greedy search about four times faster than the beam search with beam size 2, but also if we double the beam size, the decoding time will be almost doubled, because we have to maintain and update the increased number of hypotheses along with the corresponding hidden states at every time step. (2) As pointed out by

(Cho, 2016; Chung et al., 2016; Li and Jurafsky, 2016), beam search does not provide diverse n-best lists effectively so that a large beam size (above 20) does not help to increase the quality of translations in many cases.

3.2. n-best List Expansion

n-best List Expansion by kNN Search. Let X^i, Y^i and s_0^i be the i^{th} source sentence, i^{th} target sentence, and the initial hidden state derived from the i^{th} source sentence, respectively. Recall that s_0^i is obtained as a by-product of X^i-to-Y^i translation. Thus whenever we translate X^i into Y^i, we can store the pair $\langle s_0^i, Y^i \rangle$ into our own log database.

We assume that there is a log database D that contains more than the k number of $\langle s_0^i, Y^i \rangle$ pairs. Given a source sentence X^i, we calculate s_0^i using X^i and finding the k number of s_0^j in D that have the lowest Euclidean distances between s_0^i and s_0^j:

$$R = argmin_{j \neq i}^{k} d\left(s_0^i, s_0^j \in D\right).$$

Here, $argmin^k$ returns the k most similar sentences to the source.

Since each $s_0^j \in$ R has its corresponding translation Y^j, and |R| = k, we can obtain the corresponding k number of translations. Here, we set $k = n - 1$, because our aim is to obtain additional $n - 1$ candidates.

The rationale behind this approach is that an initial hidden state s_0 is a vector that embeds the corresponding source sentence, and that the encoder-decoder with attention could learn the embeddings effectively. In Section 4, we will show that this approach shows the high-level of accuracy even when there are many initial hidden states in the database.

Approximate kNN Search. Finding k-nearest neighbors for a source sentence takes a huge amount of time in the real-world scenario due to the two main reasons: first, an initial hidden state is usually a high-dimensional vector (the dimension of 512, 1024 or more) so that even single Euclidean distance calculation takes nonnegligible time. Second, calculating Euclidean distances for every possible pair consumes a huge amount of time when there are many translation logs in the database. In other words, given a source sentence X^i, it takes lots of time to calculate $d\left(s_0^i, s_0^j\right)$ for every s_0^j in the database.

We cope with the first problem by applying spherical hashing (Heo et al., 2012) which is one of the most efficient Locality Sensitive Hashing (LSH) schemes. One main characteristics of spherical hashing is that the algorithm is *data-dependent*, which means that the performance does not greatly depend on the data distributions. It converts a high-dimensional vector into a *signature* (relatively low-dimensional vector) by defining the H number of binary hash functions:

$$sig\left(s_0^j\right) = \langle f_1\left(s_0^j\right), f_2\left(s_0^j\right), \dots, f_H(s_0^j) \rangle,$$

where $f_1(\cdot), f_2(\cdot), \dots, f_H(\cdot)$ are the binary hash functions learnt by the spherical hashing algorithm.

Although spherical hashing itself can find k-nearest neighbors effectively, we further improve the algorithm by applying Signature Selection LSH (S2LSH) (Park et al., 2015). The process of this algorithm is as follows: the first step is to generate a *signature pool* consisting of many diverse signatures. Each signature in the signature pool can be generated using M random and distinct integers $r_1, r_2, \dots, r_M$ each ranged from 1 and H:

$$sig_l\left(s_0^j\right) = \langle f_{r_1}\left(s_0^j\right), f_{r_2}\left(s_0^j\right), \dots, f_{r_M}(s_0^j) \rangle$$

When a query vector s_0^q is given, the algorithm determines which signatures are the most effective for finding k-nearest neighbors of s_0^q in real time. The effectiveness of

signature l for query s_0^q is defined by the percentage of the k-nearest neighbors of q among the vectors that have the same signature l. As a next step, we find the candidates of k-nearest neighbors of s_0^q and calculate the *exact* Euclidean distances between them. By using this pruning process, the unnecessary Euclidean distance computations are significantly reduced while the accuracy is only slightly decreased.

3.3. n-best Rescoring

The next step is to rescore the n-best list obtained by the previous subsections. After the rescoring n-best lists, the target sentence with the best score will be returned to the user. There are many ways to rescore the n-best: (1) the simplest method is to reuse the decoder that was used in greedy search for calculating the score. (2) If we learn additional decoders with different configurations, then the ensemble model can also be used for rescoring. (3) Another popular rescoring method is to use language model, which has been known as particularly efficient for statistical machine translation.

Let X^i, N be a source sentence and its n-best list, respectively. Assume we reuse the decoder that was used in the greedy search process. In order to rescore the n-best list, we need to know $log\, P(Y^j|X^i)$ for all $Y^j \in N$. If Y^j was obtained by greedy search, we do not have to recalculate $log\, P(Y^j|X^i)$. If Y^j is obtained by kNN search, we need to calculate this probability because we only know the value of $log\, P(Y^j|X^l)$. Here, $X^l \neq X^i$.

Note the recalculation process is much faster than beam search (as will be shown in the experimental results of Section 4) since $log\, P(Y^j|X^i)$ for all Y^j can be simultaneously calculated through the network. In addition, all of the hidden states of the decoder can be fed into softmax layer in a batch since we know the target words already. It is interesting because, like human beings, the scoring task is much easier than decoding for neural machine translators.

4. Experiments

4.1. Experimental Setup

For evaluation, we use the Korean-to-English language pair and exploit a state-of-the-art neural machine translation system introduced in Section 2 as a baseline. The system is based on *Theano* (Theano Development Team, 2016) and we modify the codes to use *Platoon*[3] for multi-gpu training. We use 4 GeForce GTX Titan GPUs.

Data Preparation. We use Korean-to-English parallel corpora for training. They consist of about 2.0 million parallel sentences from various types of domains such as news, lectures, emails, etc. These sentences are automatically tokenized, and for Korean language, they are word-spaced, and converted to basic consonants and vowels (Korean alphabets). Then based on these Korean/English alphabets, we construct about 30K subwords using the technique of (Sennrich et al., 2016). Because of a limited amount of GPU memory, we filter out the sentences of more than 50 words and use a minibatch of size 64.

Our test data consist of 3000 parallel sentences. Each Korean sentence has 9 synonymous sentences, and all of the synonymous sentences have the same English references. That is to say, there are 300 distinct English sentences in this test set. The test data is constructed based on the following process: first, we selected the most *popular* 300 English sentences from the user translation data from Web[4]. The selected English sentences are so simple that many people participated in translating them into Korean and that there are more than 10 Korean references for each English sentence. Since there are user ratings data that measure the quality of the Korean references, we selected only 10

[3] https://github.com/mila-udem/platoon
[4] http://usertranslation.naver.com/

Korean sentences of the highest quality for each English sentence and filtered out the other references.

Model Training. Our model follows the default setting of the DL4MT-tutorial: we use the GRUs for the recurrent neural networks and exploit the conditional GRU technique for the decoder. The number of hidden layers is 1 for both encoder and decoder. The encoder has 1000 hidden units for each forward and backward direction, the decoder has 1000 hidden units, and each subword is embedded into 500-dimensional vector space. We use the gradient clipping technique (Pascanu et al., 2012) with a threshold 1, and AdaDelta (Zeiler, 2012) as an optimizer. We reshuffle the entire corpus at the start of each epoch.

***k*NN Search.** For kNN search, we build two log databases: 3K DB and 100K DB: (1) 3K DB consists of the source sentences from our test set and their translation results. Note each source sentence has only 9 synonymous sentences in the test set; (2) 100K DB contains 97K source sentences from our training corpus and their translation results together with 3K DB.

For approximate kNN search, we use the default parameter settings proposed in (Park et al., 2015): we set H to 1000 and M to a random integer ranged from 5 and 15, and set the number of signatures in a signature pool to 500.

***n*-best Rescoring and Evaluation.** Before we rescore the n-best list, the score of each candidate is normalized by its decoding length. After generating translation list, the subwords are appropriately concatenated to the words. The translation quality is measured by *Tokenized BLEU* using the *multi-bleu* script from Moses.

4.2. Experimental Results

Figure 1 shows a comparison result of beam search (BS), our approach with kNN search over 3K sentences (SST 3K), our approach with kNN search over 100K sentences (SST-100K), and our approach with approximate kNN search over 100K sentences (SST*-100K). The BLEU scores according to the n-best list are described on the left-hand side of figure, and the total elapsed time are described on the right-hand side.

Note all of our approaches outperform beam search in terms of BLEU when there are sufficient number of candidates provided. Even in case of using SST*-100K, the BLEU score is higher than beam search. Also, all of our approaches are faster than beam search when n is large enough: (1) SST-3K is the fastest, (2) SST-100K is faster than beam search when n is larger than 16, and (3) the translation speed of SST-100K can be significantly improved by using SST*-100K. Note if we compare BS with SST-3K when n is 64, SST-3K increases the BLEU score by $+0.99$ points and is more than 28 times faster than BS.

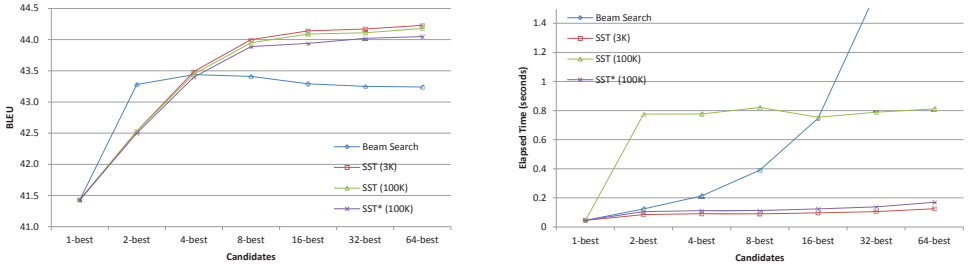

Figure 1. A comparison of beam search and our approaches in terms of BLEU and elapsed time

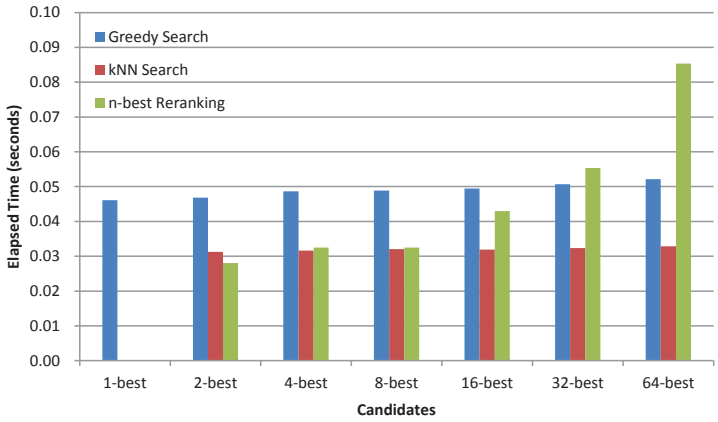

Figure 2. The elapsed time of greedy search, approximate $\boldsymbol{k}$NN search, and $\boldsymbol{n}$-best rescoring for SST*-100K

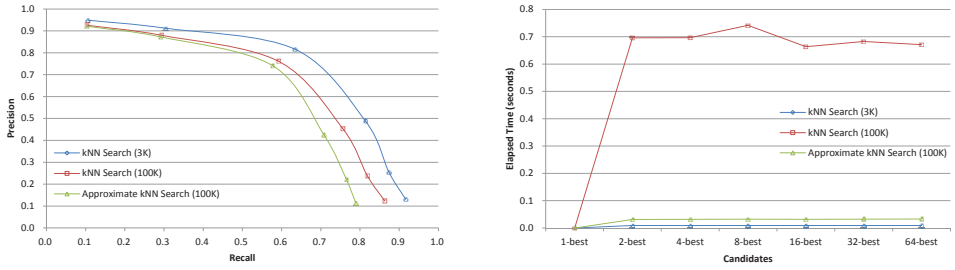

Figure 3. An analysis of recall, precision, and elapsed time for different $\boldsymbol{k}$NN methods

4.3. Performance Analysis

Elapsed Time. The total elapsed time of our approaches consists of three major components: greedy search time, (approximate) kNN search time, and n-best rescoring time. Figure 2 shows the elapsed time of each component for SST*-100K: (1) obviously, the greedy search time does not depend on the number of candidates; (2) the kNN search time also does not greatly depend on n, because the S2LSH algorithm carefully controls the elapsed time; (3) as the number of candidate increases, so does the n-best rescoring time.

Note that the n-best rescoring time increases slowly as the number of candidates increases: for example, while the rescoring time is 0.0325 seconds when n is 8, it is 0.0430 seconds when n is 16. That is to say, even if we have to rescore twice as much as before, the elapsed time would not be doubled. It is because we do the rescoring in a batch. Even when n is 64, the rescoring time is still lower than 0.1 seconds.

kNN Search. Because the kNN search methods play a crucial role in our approches, we need to analyze their behaviors. The left-hand side of Figure 3 shows the recall-precision curve of different kNN operations: if we kNN search over 3K sentences, the recall is more than 80% when precision is 50%, which means that our encoder-decoder architecture embeds the sentences quite well. If we approximate/exact kNN search over 100K sentences, the recall and precision are slightly decreased, but still shows the high level of accuracy.

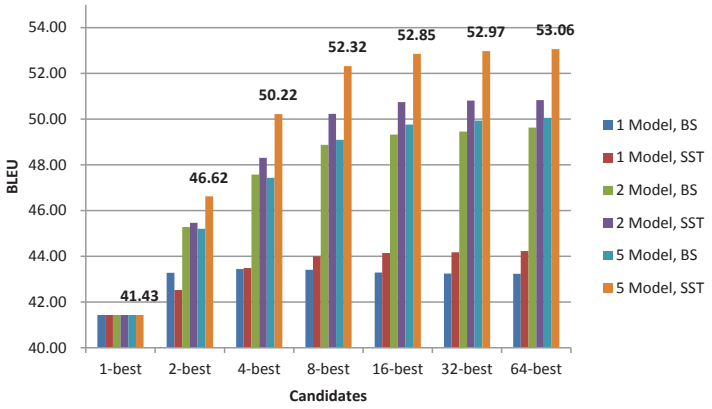

Figure 4. Effect of advanced rescoring models on BS and SST-3K

The right-hand side of Figure 3 shows the practicality of our approach. In summary, even for the large database which contains very diverse sentences with very small portion of synonymous ones, we can find the synonymous sentences in a very efficient way.

Rescoring Methods. Through the manual investigation of our n-best lists obtained, we found that the quality of candidates is much higher than we expected. Thus we wondered whether the difference of BLEU scores between SST and BS would be higher than before when we use a high-quality model (such as ensemble models) for rescoring.

Figure 4 shows the effect of advanced rescoring models on BS and SST-3K. Here, "1 Model BS" indicates the beam search algorithm with single-model rescoring, "2 Model BS" indicates the beam search algorithm with ensemble-model (2-model) rescoring, and so on. Assume $n = 8$. When we use a single model for rescoring, the difference of BLEU scores between BS and SST-3K is 0.59. However, when we use an ensemble model constructed by 2 different models for rescoring, our approach increases 1.36 and 6.82 BLEU points over 2 Model BS and 1 Model BS, respectively. When we use a more advanced rescoring model (5 Model SST), 3.23 and 8.91 BLEU points are increased over 5 Model BS and 1 Model BS, respectively. This is a surprising result because the difference could be higher if we train the ensemble models more carefully. Although the ensemble models can also be used for decoding instead of rescoring, we do not consider the ensemble model decoding, because this requires significant amount of execution time in practice.

Examples of Improvements. Table 2 shows examples of improvements when using our approach. The first column indicates the 14 synonymous source sentences, and the second column shows the result of beam search. The sentences of the third column are obtained by applying our approach to the source sentences, assuming that given a source sentence, the other 13 synonymous sentences are already translated and stored in the log database. The results show that while beam search generates six incorrect translations, our approach generates only three incorrect translations.

5. Related Work

(Li and Jurafsky, 2016) proposes a diverse decoding scheme, called *Diversity*. The main idea is that at each decoding time step t', if the words A and B are both preceded by $y_{<t'}$ and $p(A|y_{<t'}, X) > p(B|y_{<t'}, X)$, then it decreases the value of $p(B|y_{<t'}, X)$. For example, suppose that there are four candidates, "he is", "he has", "it is" and "it has", their log-probabilities are -2.5, -2.8, -3.0 and -3.1, respectively, and the beam size is

Source Sentence	Beam Search	Robust SST
kang.nam.yek.kka.ci e.tteh.key ka.cyo ?	**(X) How do you get to Gangnam Station?**	**(O) How can I go to Gangnam Station?**
kang.nam.yek.kka.ci e.tteh.key ka.cyo	(X) "I'll go to Gangnam Station to Gangnam Station."	(X) "Let's go through Gangnam Station."
kang.nam.yek.kka.ci ka.nun kil.ul al.lye cu.sey.yo .	(O) Tell me the way to Gangnam Station.	(O) Give me directions to Gangnam Station.
kang.nam.yek.kka.ci ka.nun kil.ul al.lye.cwe .	(O) Tell me the way to Gangnam Station	(O) Tell me the way to Gangnam Station.
kang.nam.yek.kka.ci ka.nun kil.i e.tteh.key toyp.ni.kka ?	**(X) What is going to Gangnam Station?**	**(O) What is the road going to Gangnam Station?**
kang.nam.yek.kka.ci e.tteh.key ka.na.yo ?	(O) How's going to Gangnam Station?	(O) How can I go to Gangnam Station?
kang.nam.yek.u.lo e.tteh.key ka.na.yo ?	(O) How's going to Gangnam Station?	(O) How can I get to the Gangnam Station?
kang.nam.yek.u.lo ka.nun kil.ul al.lye cu.sey.yo .	(O) Tell me the way to Gangnam Station.	(O) Please give me directions to Gangnam station.
kang.nam.yek.u.lo ka.nun kil.ul al.lye.cwe .	(O) Please let me know the way to Gangnam Station.	(O) Tell me the way to Gangnam Station.
kang.nam.yek.u.lo ka.nun kil.ul al.lye.cwe	(O) Please let me know the way to Gangnam Station.	(O) Please let me know the path to Gangnam Station.
kang.nam.yek.u.lo ka.nun kil.ul al.lye cu.sey.yo	**(X) Please answer the way to Gangnam Station."**	**(O) Give me a way to Gangnam Station.**
kang.nam.yek.u.lo ka.nun kil.ul al.lye cu.si.keys.sup.ni.kka ?	(O) Can you give me a way to Gangnam Station?	(O) Can you give me a way to Gangnam Station?
kang.nam.yek.u.lo e.tteh.key kal.kka.yo ?	(X) How do you go with Gangnam Station?	(X) How do you go with Gangnam Station?
kang.nam.yek.kka.ci e.tteh.key kal.kka.yo	(X) "How do we go from Gangnam Station to Gangnam Station?"	(X) How about going to Gangnam Station

Table 2. An example of synonymous source sentences and their translations

2. Then, beam search would keep the first and second candidates, namely "he is" and "he has". On the other hand, because $p("is"|"he", X)$ is larger than $p("has"|"he", X)$ and $p("is"|"it", X)$ is larger than $p("has"|"it", X)$, Diversity decreases the log probabilities of the second and fourth candidates by, say, 0.5, which become -3.3 and -3.6, respectively. Then the first and third candidates, namely "he is" and "it is", would be kept, which results in more diversification.

(Cho, 2016) points out that the main disadvantage of beam search and Diversity is that they may incur high communication overhead when implemented on multiple-GPUs for decoding. Cho proposes a new decoding algorithm, called *noisy parallel approximate decoding (NPAD)*, in which N parallel greedy searches are launched. In each greedy search, weight noise is randomly injected to the transition function of a recurrent neural network to consider a set of semantically similar configurations in the input space. Then the obtained N-best list is rescored and the one with the highest score is returned.

Although these state-of-the-art approaches have been proposed to improve beam search, none of them significantly has been able to increase the BLEU scores: (1) while Diversity increases the BLEU score of English-to-French translation by 1.0∼1.2 points with a large beam size (e.g., 200), with a small beam size (e.g., 10), it improves the BLEU score by only 0.03~0.07 points, as shown in (Cho, 2016). Since a large beam size may lead to a prohibitively long decoding time, Diversity is ill-suited for use in a real-world scenario. (2) Similarly, even with 50 simultaneous greedy searches, NPAD does not outperform the beam search with a beam size of 5.

6. Conclusion

In this paper, we propose a novel diverse decoding algorithm for accurate translation of synonymous sentences in an NMT system. We observe that the modeling power of NMT models may not be fully utilized because of insufficient coverage of search space by beam search. The proposed algorithm expands search space coverage by using previous translations of synonymous sentences through diverse decoding. The experimental results show that the proposed scheme enhances the BLEU score significantly over the state-of-the-art NMT system while being much faster.

One limitation of our approach is that, given a source sentence, there must be synonymous sentences in our log database. As future work, we plan to extend the applicability of our work by automatically generating and translating synonymous sentences in advance.

References

Bahdanau, D., Cho, K. and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv: 1409.0473.*

Cho, K. (2016). Noisy Parallel Approximate Decoding for Conditional Recurrent Language Model. *arXiv: preprint arXiv: 1605.03835.*

Chung, J., Cho, K. and Bengio, Y. (2016). A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. *arXiv preprint arXiv: 1603.06147.*

Firat, O and Cho, K. (2016). DL4MT-Tutorial: Conditional Gated Recurrent Unit with Attention Mechanism. *https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf.*

Heo, J. Lee, Y., He, J. and C., Y, S. (2012). Spherical Hashing. *In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16-21, Rhode Island, USA.

Li, J., Jurafsky, D. (2016). Mutual Information and Diverse Decoding Improve Neural Machine Translation. *arXiv preprint arXiv: 1601.00372.*

Park, Y., Hwang, H., Lee, S. (2015). A Fast k-Nearest Neighbor Search Using Query-Specific Signature Selection. *In Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 1883-1886, New York, USA.

Pascanu, R., Mikolov, T. and Bengio, Y. On the Difficulty of Training Recurrent Neural Networks. *arXiv preprint arXiv: 1211.5063*.

Sennrich, R., Haddow, B., Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv: 1508.07909*.

Theano Development Team (2016). Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints arXiv: 1605.02688.*

Zeiler, M. (2012). ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv: 1212.5701*.