Sentence-Chain Based Seq2seq Model for Corpus Expansion

Euisok Chung and Jeon Gue Park

This study focuses on a method for sequential data augmentation in order to alleviate data sparseness problems. Specifically, we present corpus expansion techniques for enhancing the coverage of a language model. Recent recurrent neural network studies show that a seq2seq model can be applied for addressing language generation issues; it has the ability to generate new sentences from given input sentences. We present a method of corpus expansion using a sentence-chain based seq2seq model. For training the seq2seq model, sentence chains are used as triples. The first two sentences in a triple are used for the encoder of the seq2seq model, while the last sentence becomes a target sequence for the decoder. Using only internal resources, evaluation results show an improvement of approximately 7.6% relative perplexity over a baseline language model of Korean text. Additionally, from a comparison with a previous study, the sentence chain approach reduces the size of the training data by 38.4% while generating 1.4-times the number of n-grams with superior performance for English text.

Keywords: Sentence chain, Lexical chain, Seq2seq model, Corpus expansion.

I. Introduction

Machine learning approaches depend on provided training data. Statistical machine translation (SMT) requires a broad-coverage bilingual corpus in order to satisfy commercial services [1]. The acquisition of language pairs is limited when there is an insufficient amount of prebuilt bilingual data. In the case of automatic speech recognition (ASR) for a specific domain, such as an automated call-center transcription service [2], a sufficient amount of domain text for a language model may not be available due to privacy policies. The training data for SMT and ASR is a type of sequential data. Therefore, this paper focuses on sequential data augmentation in order to alleviate the data sparseness problem. Specifically, we present corpus expansion techniques for enhancing the coverage of a language model.

We present a method of corpus expansion using a sentence-chain based seq2seq model. Sentence-chains are based on lexical chain studies, which are one of the main subjects in text summarization [3], [4]. A sentence-chain is a set of similar sentences within a particular body of text. We use word-embedding methods [5] to build sentence chains. Word embedding plays a role in detecting similar sentences in the initial context of a target sentence. For training the seq2seq model, sentence chains are used as triples. The first two sentences in a triple are used for the encoder of the seq2seq model, while the last sentence becomes a target sequence for the decoder. Finally, we attempt to evaluate the effectiveness of the sentence-chain based seq2seq model for corpus expansion.

The following are the major contributions of this research: First, we proposed a novel corpus expansion technique using only internal resources. Experimental results indicate significant performance improvements for

Manuscript received Aug. 14, 2016; revised Apr. 2, 2017; accepted May 22, 2017. This work was supported by the ICT R&D program of MSIP/IITP, Rep. of Korea (R0126-15-1117, Core technology development of the spontaneous speech dialogue processing for the language learning).

Euisok Chung (corresponding author, eschung@etri.re.kr) and Jeon Gue Park (jgp@etri.re.kr) are with the SW & Contents Research Laboratory, ETRI, Daejeon, Rep. of Korea.

This is an Open Access article distributed under the term of Korea Open Government License (KOGL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (http://www.kogl.or.kr/news/dataView.do?dataIdx=97).

state-of-the-art language models. Second, we present a triple seq2seq model based on sentence chain as a new learning method for language generators that can generate various similar sentences. Third, two corpus expansion methods using the triple-seq2seq model provide a solution to the unseen n-gram problem, which is a chronic problem in n-gram language models.

The remainder of this paper is structured as follows. Related works are described in Section II. In Section III, we discuss an algorithm for building sentence chains. A triple-seq2seq model is presented in Section IV, while Section V describes how to prepare input data for the triple-seq2seq model. In Section VI, an experimental evaluation is presented. Our conclusions and future work are detailed in Section VII.

II. Related Works

Data augmentation techniques have been applied to image and speech recognition in the form of input data expansion for neural network (NN) models via signal transformations [6], [7]. In the field of text, an approach for generating text by using thesaurus-based synonym replacement has been proposed [8]. For studies closely related to this paper, there have been studies on neural network machine translation (NMT) using the back-translation approach [9], [10]. This is an approach for expanding a parallel training corpus by generating corresponding language for the encoder via back-translation of the target language when the parallel corpus for the NMT is insufficient. However, we focus on expanding a monolingual corpus using a different type of NMT technique.

Previous research on corpus expansion can be divided into two main types of approaches: external resource approaches and internal resource approaches. As examples of external resource approaches, an in-domain corpus expansion using focused Web crawling was proposed in [11]. For Chinese word segmentation, a technique of exploiting the redundancy in a Web corpus was first proposed in [12]. As a lexicon-based approach, WordNet is an external resource that is almost always available. WordNet has been utilized as a query expansion technique [13] for information retrieval and feature selection for sentiment analysis [14]. As an example of an internal resource approach, a method for corpus expansion for SMT using semantic role label (SRL) substitution rules that rely on an SRL labeler was proposed in [15]. In order to consider cases where external resources do not exist, we attempt to expand a corpus using only internal resources. Additionally, approaches that require tools that rely on additional knowledge resources, such as [15], appear to have limitations in versatility and scalability. Therefore, we attempt to use only the information inherent to the input data itself.

Statistical paraphrase generations are also related research areas. An analysis and definition of paraphrase occurrences were described in [16]. From the perspective of a statistical approach, paraphrases do not have the same meaning. A small editing distance between strings can be used as the measure of a paraphrase [17]. A more sophisticated approach for paraphrase detection was proposed using a method of dynamic pooling and unfolding recursive auto-encoders (RAE) [18]. The issue here is that the RAE was based on syntactic analysis, which generally has difficulty processing ungrammatical sentences such as freeboard text and spontaneous speech transcriptions. In the first statistical paraphrase generation (SPG) scheme, presented in [19], the SPG model consisted of a paraphrase model, language model, and usability model. Here, the model exploits multiple resources to resolve the data sparseness problem. On the other hand, the acquisition of sentential paraphrases using crowdsourcing would be a practical alternative [20]. However, we attempt to find automatic methods for generating new sentences using only the provided training corpus.

Neural conversational models have achieved impressive results in deep learning studies. These studies provide an easy means of developing a natural language chat-bot, mainly using the seq2seq framework [21], [22]. In some cases, triple-type data schemes were adopted. In [21], the triples consist of context, message, and responses for generating novel responses. A question answering study, such as that in [23], converts documents into context-query-answer triples in order to answer questions. We use a similar approach to build a dataset for corpus expansion. The major issue is that the triples are based on a data structure for a particular application, making it difficult to apply them to plain text extensions. Therefore, we use the concept of sentence chains to create our triples.

III. Sentence Chain

For building datasets for corpus expansion, we use triples composed of two sentences for encoder input and one sentence for decoder input for training the seq2seq model. In the generation step, the decoder generates one sentence from the two sentences that are inputted to the encoder. These triple-type datasets are similar to those used in previous studies. In a study on generating conversational responses, context-message-response

https://doi.org/10.4218/etrij.17.0116.0074

triples were mined from a twitter corpus for use as datasets [21]. The authors selected triples by considering bigram frequency, and then created a crowd-source raters filter from the triples. In [23], document-query-answer triples were built from online newspaper articles and their corresponding summaries. We propose a methodology for automatically generating triples. Inspired by works on lexical chains [3], [4] and word embedding [5], we propose an algorithm for building sentence chains.

The input for the sentence chain algorithm is a document $D = \{S_1, \ldots, S_m\}$, which is composed of sentences $S_i = \{W_1^i, \ldots, W_n^i\}$. The sentences in D are processed in reverse order because it is assumed that the search area is a preceding history. The algorithm builds a sentence chain SC if constraints are satisfied in each s_i of the D. Here, for the words in s_i, it is assumed that word embedding is applied. This allows us to use the skip-gram model [5]. The skip-gram model uses an approach that maximizes the average of log probabilities $\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leqslant j\leqslant c, j\neq 0}\log p(W_t+j|W_t)$ for a given sequence of training words $w_1, w_2, w_3, \ldots, w_T$. The c in the objective is the size of the training context. Using this method, words with a similar context are located near each other in vector space.

In the algorithm, the SC is generated from a lexical chain LC, where the LC is derived from a partial lexical chain PLC. We first build the PLC for the word W_j^i in s_i compared to the word W_m^k in s_k from a measure of cosine distance (1) [24], [25]. In Algorithm 1, this is described in lines 1 through 12.

$$d(w_j^i, w_m^k) = 1 - \frac{w_j^i \cdot w_m^k}{\|w_j^i\| \|w_m^k\|}, (i<k<i+\delta). \quad (1)$$

In line 5, δ is the maximum distance between sentences in D. Because k is a sentence index, $k < i$ indicates that s_k is one of the previous sentences of s_i in D. Additionally, m is the mth word of the kth sentence. In line 7, the function **d** returns the distance d_0 between two vectors, W_j^i and W_m^k. In line 9, (W_j^i, W_m^k, d_0) is appended to the PLC when the d_0 is lower than max_d. In Fig. 1, the underlined words are word-embedded vocabularies. The 5th word "*hungarian*" in s_9 is connected to the words of s_6 to form the PLC. "*border*" is ignored because its d value is larger than max_d.

The algorithm then finds the next candidate words for the LC using (2) in lines 13 through 26. This indicates that $PLC(W_j^i, W_m^k)$ finds the W_p^n at the closest distance. In line 14, the PLC is sorted according to d. Lines 16 and 17 show that (W_j^i, W_m^k, d) is used for LC candidates (LCC) up to nb_cand. In lines 18 through 23, each word vector of the PLC[t] is compared with W_p^n to build LCC. In line 22, the distance values between components are interpolated using (2). In line 23, (i', k', n, g) is appended to LCC. Figure 1 shows that there are four candidates in the LCC group.

Algorithm 1. Building sentence chains.

Inputs: $D = \{S_1, S_2, S_3, \ldots, S_m\}, S_i = \{W_1^i, W_2^i, \ldots, W_n^i\}$
Outputs: sentence chain SC
Initialize[1]: $SC \leftarrow \varnothing$, $\lambda_1 = 0.4$, $\lambda_2 = 0.3$, $\lambda_3 = 0.3$, $\delta = 5$, $max\_d = 0.4$, $nb\_cand = 2$

```
1   for s_i in D do
2     LC ← ∅
3     for W_j^i in s_i do
4       PLC ← ∅
5       for s_k in D, 0 < k < i, k ≥ i − δ do
6         for W_m^k in s_k do
7           d_0 ← d (W_j^i, W_m^k)                ⇒ use (1)
8           if d_0 < max_d then
9             append (W_j^i, W_m^k, d_0) to PLC
10          end if
11        end for
12      end for
13      if |PLC| > 0 then
14        PLC ← sort_by_d (PLC)
15        LCC ← ∅
16        for t in |PLC|, t < nb_cand do
17          W_j^i', W_m^k', d' ← PLC[t]
18          for s_n in D, 0 < n < k', n ≥ k' − δ do
19            for W_m^n in s_n do
20              d_1 ← d (W_j^i', W_m^n)
21              d_2 ← d (W_m^k', W_m^n)
22              g ← λ_1·d_1 + λ_2·d_2 + λ_3·d'   ⇒ use (2)
23              append (i', k', n, g) to LCC
24            end for
25          end for
26        end for
27        if |LCC| > 0 then
28          LCC ← sort_by_g (LCC)
29          for t in |LCC|, t < nb_cand do
30            append LCC[t] to LC
31          end for
32        end if
33      end if
34    end for
35    i', k', n' ← min_g LC
36    append (s_i', s_k', s_n' ) to SC
37  end for
```

[1] The constant values were determined empirically.

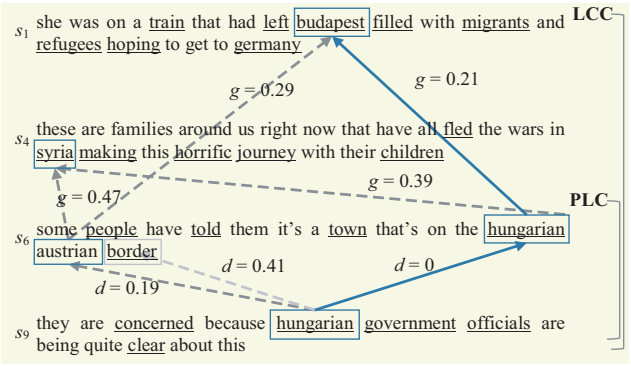

Fig. 1. Example of Building a Sentence Chain: The partial LC (PLC) and LC candidates (LCC) are built for the 5th word of s_9. The blue line is included in the LC. The final sentence chain is (s_1, s_6, s_9). d and g are from (1) and (2).

$$g(w_j^i, w_m^k, w_p^n) = \lambda_1 \cdot d(w_j^i, w_p^n) + \lambda_2 \cdot d(w_m^k, w_p^n) + \lambda_3 \cdot d(w_j^i, w_m^k), (\lambda_1 + \lambda_2 + \lambda_3 = 1). \quad (2)$$

Finally, the *LCC* is sorted using g to build the *LC* in lines 27 through 32. We select the best sentence index (i, k, n) from the *LC* to append (s_i, s_k, s_n) to the *SC*. The procedure for building sentence chains is presented in lines 35 and 36 in Algorithm 1, and the final sentence chain is described in Fig. 1.

In order to enhance the performance of this algorithm, we use a beam search during the SC building process. Here, *nb_cand* indicates the size of the beam. This aids in avoiding excessive generation of *LC*s. The main difference between our approach and previous studies is the use of word embedding. In [3], the researchers constructed lexical chains using various knowledge sources, such as WordNet and natural language analyzers. An evaluation of our sentence chain method is provided in the evaluation section. We expect that the triples generated from our sentence chains will reduce the size of the training data for the seq2seq model because the lexical chain approach stems from summarization issues.

IV. Triple-Seq2seq Model

A recurrent neural network (RNN) can handle long-distance dependencies in sequential data streams. An enhanced RNN was proposed in [26], which is an LSTM that improves upon the RNN by solving numerous problems that are not solvable by previous RNN methodologies, such as the vanishing gradient issue. The LSTM is described in (3). We use the formulation proposed in [27]. It describes that there is an input gate i_t, forget gate f_t, output gate o_t, and memory cell m_t at a time t when x_t is an input vector. The core idea of the LSTM is to regulate information flow into and out of the memory cell m_t by using non-linear gating units [27]. In (3), l_t is the block input, which is calculated using the current input x_t and the previous block output h_{t-1}. *tanh* is a hyperbolic tangent activation function. The input gate i_t has the role of regulating the block input l_t. The forget gate f_t decides if the previous memory state m_{t-1} should be forgotten when building the current memory state m_t. The operator $\odot$ denotes point-wise multiplication of two vectors. The output gate o_t controls the block output h_t. The gating units use the logistic sigmoid as an activation function. Here, W_* is the weight matrix estimated during the training phase using the back-propagation through time (BPTT) algorithm. When an input stream ends at time t, h_t is considered an embedding value of the input sequence.

$$l_t = tanh([W_{xl}, W_{hl}] \cdot [x_t, h_{t-1}]), i_t = \sigma([W_{xi}, W_{hi}] \cdot [x_t, h_{t-1}])$$
$$f_t = \sigma([W_{xf}, W_{hf}]), o_t = \sigma([W_{xo}, W_{ho}] \cdot [x_t, h_{t-1}])$$
$$m_t = f_t \odot m_{t-1} + i_t \odot l_t, h_t = o_t \odot tanh(m_t). \quad (3)$$

In this study, we use a standard LSTM-based seq2seq model to expand the input corpus. The seq2seq model is a general end-to-end approach used in sequence learning [28]. It has been applied to SMT studies to translate an input language into a target language [28], [29]. The seq2seq model is also used for sentence embedding [30], [31]. The triple-seq2seq model (TSM) is described in (4), (5), and (6). The difference between triple-seq2seq and a general seq2seq model is that TSM encodes $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ into an embedding value h_{AB} that is inputted to the decoder, which generates $C = \{c_1, \ldots, c_n\}$. The encoders in that process A and B in (4) share the parameters of the LSTM, whereas the decoder for generating C in (6) is a separate LSTM. The encoders in (4) create embedding values: h_A for A and h_B for B. (5) shows that the TSM reduces a concatenation of h_A and h_B to h_{AB} in order to match the dimensions of the recurrent state of the decoder. Here, W_v is estimated during the training phase.

$$h_A = LSTM_{\text{enc}}(A), h_B = LSTM_{\text{enc}}(B), \quad (4)$$

$$h_{AB} = W_v \cdot [h_A, h_B], \quad (5)$$

$$C = LSTM_{\text{dec}}(h_{AB}). \quad (6)$$

The TSM can be described as a conditional probability [32]. It predicts the probability of C being conditioned by A and B, as described in (7). $p(c_t \mid h_A, h_B, c_1 \cdots c_{t-1})$ computes the probability of the next word c_t at time t when conditioned by the history $c_1 \cdots c_{t-1}$, h_A, and h_B.

Here, h_A and h_B are the embedding values of A and B. In other words, the probability of c_t is calculated using a softmax function, as shown in (8), in the LSTM decoder where $g(h_{t-1}, c_t)$ is the activation function between the block output h_{t-1} and c_t.

$$p(c_1 \cdots c_n | a_1 \cdots a_n, b_1 \cdots b_n) = \prod_{t=1}^{n} p(c_t | h_A, h_B, c_1 \cdots c_{t-1}), \tag{7}$$

$$= \prod_{t=1}^{n} \frac{exp(g(h_{t-1}, c_t))}{\Sigma_v \exp(g(h_{t-1}, c_v))}. \tag{8}$$

Figure 2 presents an example of the TSM learning process. Here, the sentence chain (s_1, s_6, s_9) found in Fig. 1 is described with $A = s_1$ and $B = s_6$ connected to LSTM encoder, and $C = s_9$ connected to LSTM decoder. This sentence chain becomes the training data for the TSM. "<EOS>" is a mark indicating an end-of-sentence, which is typical in seq2seq models. This mark acts as both start and stop signals during the decoding phase.

V. Corpus Expansion

1. Cross-Document Based Sentence Pairs

The sentence chain algorithm builds a set of triples used for training the TSM prior to corpus expansion. The TSM uses two sentences from a triple as input to generate one output sentence. Here, the major concern is how to select the two input sentences. We use an internal resource approach for corpus expansion. A restriction is encountered where the expansion stage depends on the resources of the training stage. We assume that the training resources are compiled from multiple documents.

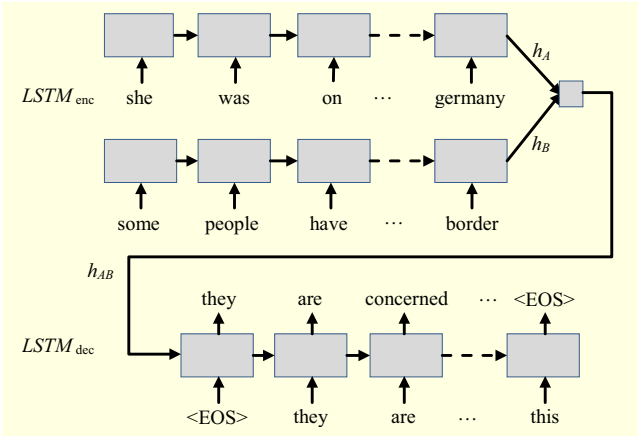

Fig. 2. Example of the Triple-seq2seq model.

One triple for training comes from one document. Thus, one sentence chain cannot cross documents. If we extract similar sentences from different documents, the sentences can be used as input sentences for the TSM. These cross-document sentence pairs may be written by different authors, which leads us to expect the TSM to be capable of generating various expressions.

The process for building cross-document sentence chains is as follows: 1) shuffle sentences from the input text, 2) randomly sample pairs of sentences from different documents, 3) compute the similarity of each pair of sentences, 4) insert a pair of the best matching sentences into a cross-document sentence chain, and 5) iterate from 2) through 4) until a sufficient number of inputs is obtained. Step 3) computes the similarity between two sentences using (9). This is similar to the distance measure described in [33]. It calculates an average of all pairs of words in s_i and s_k, and uses (1) to find the distance between W_j^i and W_m^k. (9) allows an upper bound for word distance. This allows us to change the denominator of (9), which counts only words included in pairs valued lower than the upper-bound.

$$\text{similarity}(s_j, s_k) = \frac{\Sigma_j^{|s_i|} \Sigma_m^{|s_k|} d(w_j^i, w_m^k)}{|s_i| \times |s_k|}. \tag{9}$$

Table 1 contains an example sentence (c) generated by the TSM using the cross-document sentence pair (a), (b). Here, (b) is the related sentence extracted using the cross-document approach. The generated sentence (c) has a semantic relationship with the input sentences (a) and (b), although it is not a perfect sentence.

2. Triple-Based Sentence Pairs

For the second policy of corpus expansion, we use a feature of the seq2seq architecture. Empirically, we found that the TSM does not always generate the target sentence for a triple given an input sentence pair from the same

Table 1. Examples of cross-document based sentence pairing.

Type	Cross-document sentence pair (a), (b) and generated sentence (c).
(a)	Some people have told . . . on the Hungarian Austrian border.
(b)	Let's talk more about Europe's deepening refugee crisis.
(c)	It's now the people on the ground they're coming from.

triple. Therefore, the triples used during the training phase can be applied in the generation phase. We simply shuffle the order of components in a triple to generate additional inputs for the TSM.

In Fig. 3, we show that the triple (A, B, C) used in the training phase (a) is used again in the corpus expansion phase (b). If $AB \rightarrow C$ indicates that A and B are the inputs for the encoders of the TSM, and C is the target sentence of the decoder, then $AB \rightarrow C'$ is equivalent in the expansion phase, and inputs A and B can generate C'. $BA \rightarrow C''$ can be described similarly. Here, we expect that C, C', and C'' are different expressions, meaning $C \neq C' \neq C''$. Therefore, we can directly extract a set of inputs from the triple (A, B, C), such as $\{AB, AC, BA, BC, CA, CB\}$.

Examples of generated sentences are presented in Table 2, the second row of which contains a triple extracted as a sentence chain. The triple is the sentence chain (s_1, s_6, s_9), found in Fig. 1. The input types for the sentence pairs were derived from this triple. From AB to CB, the sentences were generated using a TSM model trained with triples. From the generation of input type AB, the TSM shows that it can generate a different sentence given an input sentence pair that is identical to the sentence pair for the encoder of the TSM during the training phase. In the final row, "_UNK" is an unknown word mark due to a fixed vocabulary size.

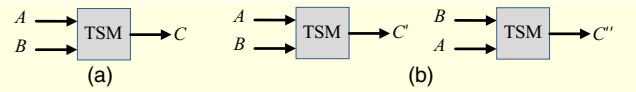

Fig. 3. Usage of triples: (a) training and (b) corpus expansion.

Table 2. Examples of triple-based sentence pairing.

Type	Triple input and generated sentences.
TRIPLE (A, B, C) A: She was on a train that had left Budapest . . . to get to Germany. B: Some people have told . . . on the Hungarian Austrian border. C: They are concerned because Hungarian government . . . clear about this.	
AB	They are running around the border here . . . trying to get people off.
AC	They're told government they can . . . government get to the people.
BA	But the Hungarian government says that they want to keep it in.
BC	I want to get this very quickly black reporting on the Serbian . . .
CA	And Germany has been very clear on the very important issue.
CB	It's a small government station and the _UNK are _UNK up.

VI. Evaluation

In order to evaluate the corpus expansion, we used two methodologies: perplexity (PPL) of the language model and the n-gram hit-ratio. Minimizing PPL, $P(w_1, w_2, \ldots, w_m)^{-1/m}$, is to maximize probability, because PPL is the inverse probability of the test set, normalized by the number of words. That is to say, a lower PPL indicates a better model [34]. If a language model built from the augmented corpus shows improved perplexity for the test set, it indicates the usefulness of our approach for corpus expansion. The n-gram hit-ratio is the ratio of the number of components per n of the n-grams hit in the language model over the amount of unseen data [35]. We expect that the hit-ratio for high-order n-grams in the test set will increase relative to the baseline when we use the TSM-based approach. This would indicate that our approach can generate unseen high-order n-grams. In the following section, we first describe the experimental settings, and then present the results.

1. Experimental Settings

A. Dataset

The dataset used for the evaluation of corpus expansion includes Korean community documents (KOR), Korean part-of-speech tag-attached documents (KOR-POS), and CNN transcriptions (ENG). The triples generated from sentence chains use a corpus composed of these documents. We selected documents written in a colloquial style because spoken language processing is our main interest. Additionally, we determined that large amounts of vocabulary may be inappropriate for TSM-based corpus expansion. We wished to avoid a long training phase during the experiments. Therefore, dialogic sentences, such as interview transcriptions and freeboard text, were prepared for the evaluations.

KOR was extracted from a women's community site[2]. Using the word-segmentation tool (WST) from [36], we segmented the words from KOR into morphemes. For KOR-POS, we attached a part-of-speech tag to each morpheme. We will refer to morphemes as words for

[2] http://www.82cook.com/entiz/enti.php?bn=15, 15 Aug 2013~5 Jan 2015

https://doi.org/10.4218/etrij.17.0116.0074

convenience. KOR had an ungrammatical style in its word spacing and sentence boundaries. We resolved the word space issue using the WST. For the sentence unit issue, sentence-end detection rules were used to divide KOR into appropriate sentence units. For ENG, CNN transcriptions were extracted[3]. We used a randomly sampled dataset from the extracted text for the KOR, KOR-POS, and ENG experiments.

The construction of sentence chains relies on a similarity measure between words (1), which requires a word embedding process. We used the word2vec tool [5] to convert words into vectors. We only selected words that occurred at least three times within their respective datasets, yielding the following final dataset; KOR (43,859), KOR-POS (50,715), and ENG (33,917). The size of the word vector is 120, which is the word embedding size. This refers to the vector dimension and is a parameter used to convert vocabularies into vector values. The words in the vocabulary set proceed to word embedding using the left- and right-six context words from among the words in the document. We then constructed triples using Algorithm 1. The results are 249,280 triples for KOR, 231,242 triples for KOR-POS, and 499,048 triples for ENG. The size of the ENG-triples is larger than the other datasets because the ENG text is composed of many more words. These triples are used in the TSM training phase. The configuration of the dataset for evaluation is summarized in Table 3.

B. TSM Configuration

The LSTM-based TSM was developed using the Tensorflow tool described in [37]. Table 4 describes the configuration of the TSM. Configuration parameters such as learning rate, learning rate decay factor, max gradient norm, and cell size followed the neural translation model of the example in the Tensorflow seq2seq library[4]. Other constants, such as sequence size, embedding size, and vocabulary size, were experimentally determined based on a performance evaluation of the development text. We chose smaller values when possible. The reason for this is that the learning time of the TSM was approximately 2 to 3 days, despite using a GPU server with a 3.0-GHz Intel Xenon E5-2623 V3 and an NVidia GTX980Ti.

Table 3. Dataset configuration.

Data type	KOR / KOR-POS	ENG
Documents	20,307	3,122
Words	7,831,463	15,995,598
Sentences	508,871	1,154,275
Average words in a sentence	15	13
Train documents	16,245	2,497
Development documents	2,031	312
Test documents	2,031	313
Vocabulary	43,859/50,715	33,917
Triples	249,280/231,242	499,048

C. Input Sentence Pairs for Corpus Expansion

The number of input sentence pairs used for corpus expansion is presented in Table 5. In order to generate new sentences, we built two types of input sentence pairs. For the first type, *cross.doc*, we used the process of building cross-document based sentence pairs, which is described in Section V. Empirically, this was time consuming because the number of similar sentence pairs depends on the size of the sampled text. In the case of the second input type, *triple*, the building step is very simple. The order of the triple components, A, B, and C, is simply shuffled. In each input type for the *triple*, we remove duplicated sentence pairs. Additionally, this can

Table 4. Configuration of the TSM.

Configuration type: *value*	
Learning rate: 0.5	Batch size: 64
Learning rate decay factor: 0.99	Sequence size: 30
Max gradient norm: 5.0	Embedding size: 120
Cell size: 1024	Vocabulary size: 15,000

Table 5. Number of input sentence pairs for corpus expansion.

Input type		KOR	KOR-POS	ENG
cross.doc		202,410	160,938	870,540
triple	AB	213,133	198,272	434,689
	AC	242,382	225,500	485,982
	BA	213,133	198,272	434,689
	BC	241,045	224,706	482,309
	CA	242,382	225,500	485,982
	CB	241,045	224,706	482,309

[3] http://transcripts.cnn.com/TRANSCRIPTS/, 3 Sep 2015~28 Feb 2016

[4] https://www.tensorflow.org/versions/r0.10/tutorials/seq2seq

generate a larger set of input data than a cross-document approach.

2. Results

A. TSM Training

Figure 4 illustrates the learning process of the TSM for the KOR, KOR-POS, and ENG *triples* shown in Table 3, where *train_err* is the perplexity of the cross entropy of the training error value. The TSM learning configuration follows the values in Table 4. As learning progresses, the *learning_rate* is reduced by the ratio of the *learning_rate_decay_factor* when the error value increases during the final three batch operation steps. In the case of KOR, the initial *train_err* value is higher than ENG. However, as the time step increases, the *train_err* value converges to a lower level. This means that KOR's TSM is learning better than ENG's TSM. This is confirmed by the fact that the *learning_rate* reduction of KOR is slower than that of ENG.

The performance results of the trained TSM are described in Table 6. The experimental results show the correlation between corpus expansion and the TSM learning curves. As corpus expansion performance becomes better, the *train_err* reduction of the TSM becomes faster, and at the same time, the rate of decrease in the *learning_rate* becomes slower.

B. Evaluation of Corpus Expansion

The evaluations of corpus expansion are presented in Table 6, where the LM type *baseline* of (a) KOR used 16,245 training documents from KOR, described in Table 3, to build a 4-gram LM using the SRILM toolkit [38]. Additionally, *cross.doc* is an LM of 202,410 sentences generated from the same number of sentence pairs shown in Table 5. The LM type *+cross.doc* is an interpolation of *baseline* and *cross.doc* using interpolation weights estimated from the development set. Furthermore, *triple* is made up of six types of LM: AB, AC, BA, BC, CA, and CB LMs. The AB LM for the (a) KOR test was built from 213,133 sentences generated from the AB input sentence pairs shown in Table 5. In the same manner as *+cross.doc*, *+triple* is an interpolation of *baseline* and these six LMs. The last LM type, *+cross.doc+triple,* is an interpolation of all LMs: *baseline*, *cross.doc*, and *triple*.

In all experiments on corpus expansion, *+cross.doc+triple* LMs achieved the best PPL reduction rates (RR). The results for *+triple* are better than those for *+cross.doc*, while the results for *+triple* are slightly lower than those for *+cross.doc+triple*. In the (b) KOR-POS test, the best RR was 7.6%. This suggests that the part-of-speech information plays a role in generating a better LM than that in other tests. Although the results of the (c) ENG test were not better than those of the (a) KOR and (b) KOR-POS tests, they still indicate that our methods of corpus expansion can be applied to multilingual text.

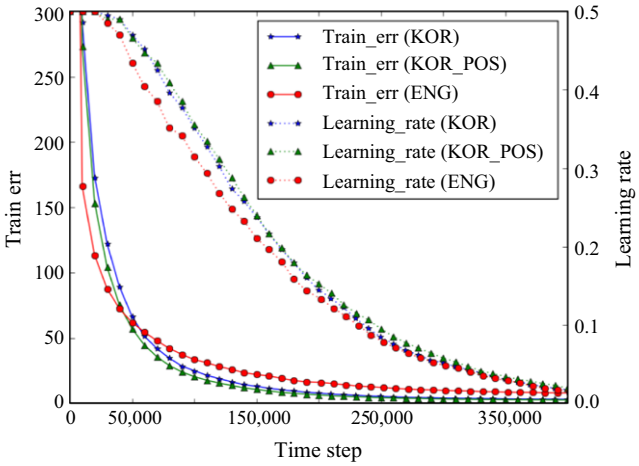

Fig. 4. Learning curve of TSM training.

Table 6. Results of corpus expansion.

LM type	dev		test	
	PPL	RR	PPL	RR
(a) KOR				
baseline	257.6	N/A	266.0	N/A
+cross.doc	246.6	4.2%	254.6	4.2%
+triple	238.9	7.2%	247.0	7.1%
+cross.doc +triple	238.7	7.3%	246.6	7.2%
(b) KOR-POS				
baseline	244.1	N/A	249.4	N/A
+cross.doc	233.8	4.2%	239.1	4.1%
+triple	225.5	7.6%	230.5	7.5%
+cross.doc +triple	225.2	7.7%	230.2	7.6%
(c) ENG				
baseline	120.8	N/A	128.8	N/A
+cross.doc	114.0	5.6%	121.8	5.4%
+triple	113.3	6.1%	120.8	6.1%
+cross.doc +triple	112.6	6.7%	120.2	6.6%

https://doi.org/10.4218/etrij.17.0116.0074

Additionally, a Wilcoxon signed-rank test [39] was conducted to determine statistical significance. In each test, the improvements of the extended models were shown to be statistically significant, achieving a p-value of (<0.001) when compared to the baseline model.

C. N-gram Hit-Ratio

The back-off LM computes the probability of unseen 4-grams, such as $p(w_1w_2w_3w_4)$, using $p(w_2w_3w_4)$ and the back-off value of a 3-gram, such as $w_1w_2w_3$. The back-off value is computed from the residual probability of previously seen 4-grams. The residual value comes from the n-gram discount approach, which subtracts a small value from the count of each n-gram. Thus, the probability of an unseen n-gram is approximated using lower-order n-grams. Thus, if the high-order n-gram hit-ratio of LM *A* is relatively higher than that of LM *B*, it can be said that the quality of LM *A* is better.

Figure 5 presents improvements in high-order n-gram hit-ratios using our corpus expansion methods. The n-gram hit-ratios are the proportion of the number of components per *n* of the n-grams hit in the LM over the total amount of unseen test data. The first four-color bar in Fig. 5 (a) shows the ratios for the number of matching 1-grams, 2-grams, 3-grams, and 4-grams when the baseline LM calculates the probability of the test text. As the corpus expansion technique is applied, it can be seen that the ratio of high-order n-grams gradually increases. In the graphs, both (a) KOR and (b) KOR-POS show similar results for n-gram hit-ratios. The tests for *+cross.doc +triple* yield the best n-gram hit-ratios across all experiments. In the case of (C) ENG, the 4-gram area is larger than in the other sections. This is caused by ENG containing relatively small n-gram variations, which results in the low perplexity of the LM test. From these experiments, we are able to confirm the basis for the performance improvements seen in Table 6. That is to say, our corpus expansion methods can handle the unseen n-gram problems.

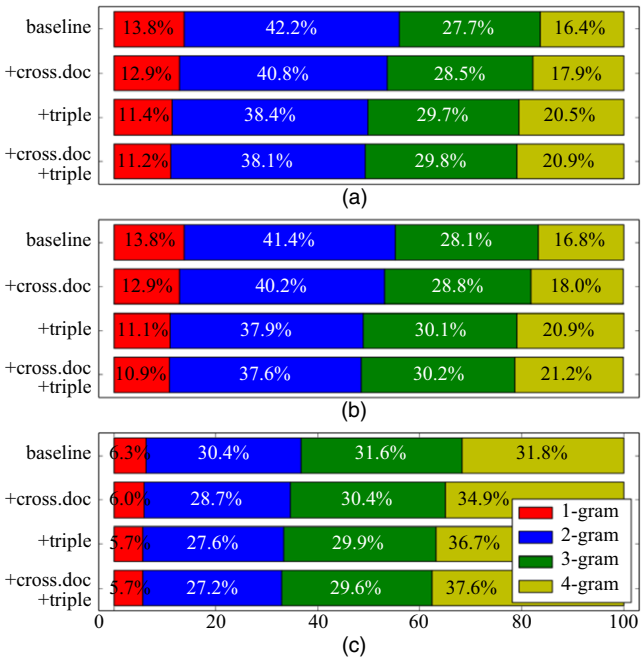

Fig. 5. The n-gram hit ratio: (a) KOR, (b) KOR-POS, and (c) ENG.

D. Sentence Embedding Comparison

The final test is a comparison with a previous approach. Although we were unable to find any studies on expanding sequential data using a neural model, the technique for sentence embedding proposed in [31] is appropriate for comparison with our sentence-chain based TSM approach. The authors used triples composed of sequential sentences for sentence embedding. They trained a seq2seq model to reconstruct the surrounding sentences for an encoded target sentence. We tested these sequential sentence based triples (SST) for corpus expansion. In this experiment, our approach uses sentence chain triples (SCT), which are evaluated on the KOR-POS and CNN datasets. These SCT tests are the *+triple* tests in shown Tables 7 and 8.

Table 7 contains the results for the KOR-POS dataset, while the results for the CNN dataset are described in Table 8. SCT reduced the number of triples used for

Table 7. Results of SST and SCT for KOR-POS.

Type			SST	SCT	Diff. of %
No. of triples			360,305	231,242	−35.8%
No. of generations			2,020,012	1,296,956	−35.7%
+triple	No.	1-g	15,001	15,001	+19.5%
		2-g	1,639,861	2,046,052	
		3-g	6,920,321	8,713,150	
		4-g	12,964,806	14,965,274	
	n-gram hit ratio	1-g	12.1%	11.1%	−8.27%
		2-g	38.0%	37.9%	−0.26%
		3-g	28.9%	30.1%	+4.15%
		4-g	21.0%	20.9%	−0.48%
test PPL RR (baseline 249.46)			6.67% (232.7)	7.59% (230.5)	+13.79%

Table 8. Results of SST and SCT for CNN.

Type			SST	SCT	diff. of %
No. of triples			810,395	499,048	−38.42%
No. of generations			4,665,024	2,805,960	−39.85%
+triple	No.	1-g	15,001	15,001	+41.92%
		2-g	1,267,374	1,427,003	
		3-g	5,261,759	6,980,361	
		4-g	10,477,538	15,734,108	
	n-gram hit ratio	1-g	6.1%	5.7%	−6.56%
		2-g	29.0%	27.6%	−4.83%
		3-g	30.1%	29.9%	−0.66%
		4-g	34.8%	36.7%	+5.46%
test PPL RR (baseline 128.84)			5.25% (122.0)	6.18% (120.8)	+17.71%

SST by 35.8% for the KOR-POS dataset. Similarly, for the CNN dataset, SCT reduced the number of triples used by 38.42%. This suggests that the training size for SCT is smaller than for SST because SCT drops triples that are not included in sentence chains. Using these two types of training sets, we trained each TSM for the SST and SCT methods for both datasets. For the number of generated sentences, SCT for the KOR-POS dataset achieves a reduction of 35.7%, with a reduction of 39.85% for the CNN dataset. However, in the *+triple* LMs, the n-gram size of SCT is bigger than that of SST for both datasets. The *+triple* for the KOR-POS dataset increases n-gram size by 19.5%. Furthermore, the CNN dataset increases *n*-gram size by 41.91%. These results indicate that SCT uses a small training set while generating a rich set of *n*-grams.

The n-gram hit-ratio and PPL results support that theory that SCT is superior to SST for corpus expansion. In the KOR-POS test, the n-gram hit-ratio increased by 4.15% when using 3-grams, while it decreased slightly when using 4-grams. As shown for the CNN test in Table 8, the hit-ratio increased by 5.46% when using 4-grams. In the PPL reduction rate tests, the improvement in the PPL reduction rate of the SCT was 13.19% for the KOR-POS test set and 17.71% for the CNN test set. We have confirmed that this improvement is statistically significant using the p-value (<0.001) obtained using the Wilcoxon signed rank test [39]. In summary, our SCT approach reduced training data size by 38.42% and generated 41.92% more n-grams while improving the high-order n-gram hit-ratio. Additionally, it decreased the reduction rate of the PPL by 17.71% for the CNN test set.

3. Discussion

We experimentally demonstrated that the proposed corpus expansion technique improves the performance of the traditional n-gram LM. We also demonstrated the reason for the performance improvement of the corpus expansion technique through the n-gram hit-ratio experiment. N-grams that are not present in the training text are generated by the TSM, allowing the probability value of the n-grams in the test data to be calculated directly. Additionally, we confirmed that the triples generated by the sentence chain approach are superior to triples composed of consecutive sentences. This means that relational sentence sets extracted by the sentence chain approach are more suitable for text generation via TSM.

We were unable to find a similar existing study, but we reviewed previous studies that improved upon PPL by using the interpolation methodology of the n-gram model. The best PPL improvement was seen in [40]. Using a generalized linear interpolation technique based on a specific feature rather than the existing linear interpolation approach, the performance improvement reached 6.5%. Our study showed a PPL improvement of 6.6% in the English domain and 7.6% in the Korean domain. Despite the differences in the evaluation sets and technical fields, we can confidently state that our results are close to the state-of-the-art for PPL improvement.

VII. Conclusions and Future Work

We proposed a sentence-chain based triple seq2seq model (TSM) for corpus expansion. The proposed model uses an approach based on internal resources without requiring external knowledge. Experimental results demonstrate that corpus expansion improves the perplexity of the language models by approximately 7.6% in the KOR-POS test. An analysis of the n-gram hit-ratio indicated that TSM-based corpus expansion has the ability to generate unseen high-order n-grams. Additionally, triple-generated inputs are an effective technique for language generation. From a comparison with a previous study, our sentence-chain approach reduced the size of the training data by 38.4% and generated 1.41-times the number of n-grams while maintaining better performance in the CNN test.

We used an LM interpolation technique because our sequential data integrations depend on language models. For future work, we wish to test additional methods for the integration of existing resources and generated resources. Additionally, we found that TSM-based corpus

https://doi.org/10.4218/etrij.17.0116.0074

expansion can be applied to natural language research, such as semantic role labeling, part-of-speech tagging, and syntactic structure analysis. The reason for this is that the KOR-POS dataset yielded the best results in our experiments. Finally, we will make an attempt to generate semantically similar sentences for the input rather than simple text extension using the results of this study. In order to achieve this, it is necessary to study semantic similarity evaluation methods and big-data based similar sentence extraction technology.

References

[1] Y. Ma and A. Way, "Bilingually Motivated Domain-Adapted Word Segmentation for Statistical Machine Translation," *Int. Conf. EACL*, Athens, Greece, Mar. 30–Apr. 3, 2009, pp. 549–557.

[2] V.L. Colson, B. Mandalia, and R.D. Swan, *Automated Call Center Transcription Services*, US Patent 7,184,539, filed Apr. 29, 2003, issued Feb. 27, 2007.

[3] R. Barzilay and M. Elhadad, "Using Lexical Chains for Text Summarization," *Int. Workshop Intell. Scalable Text Summarization*, Madrid, Spain, July 11, 1997, pp. 10–17.

[4] M. Marathe and G. Hirst, "Lexical Chains Using Distributional Measures of Concept Distance," *Int. Conf. CICLing*, Iaşi, Romania, Mar. 21–27, 2010, pp. 291–302.

[5] T. Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality," *Int. Conf. NIPS*, Lake Tahoe, NV, USA, Dec. 5–10, 2013, pp. 3111–3119.

[6] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," *Int. Conf. NIPS*, Lake Tahoe, NV, USA, Dec. 3–8, 2012, pp. 1097–1105.

[7] X. Cui, V. Goel, and B. Kingsbury, "Data Augmentation for Deep Neural Network Acoustic Modeling," *IEEE/ACM Trans. Audio, Speech Language Proc.*, vol. 23, no. 9, 2015, pp. 1469–1477.

[8] X. Zhang, J. Zhao, and Y. LeCun, "Character-Level Convolutional Networks for Text Classification," *Int. Conf. NIPS*, Montréal, Canada, Dec. 7–12, 2015, pp. 649–657.

[9] R. Sennrich, B. Haddow, A. Birch, "Improving Neural Machine Translation Models with Monolingual Data," arXiv preprint arXiv:1511.06709, 2015.

[10] J. Bradbury and R. Socher, "MetaMind Neural Machine Translation System for WMT 2016," *Int. Conf. WMT16*, Berlin, Germany, Aug. 11–12, 2016, pp. 264–267.

[11] S. Remus and C. Biemann, "Domain-Specific Corpus Expansion with Focused Webcrawling," *Int. Conf. LREC*, Portoroz, Slovenia, May 23–28, 2016.

[12] X. Qiu, C.C. Huang, and X. Huang, "Automatic Corpus Expansion for Chinese Word Segmentation by Exploiting the Redundancy of Web Information," *Int. Conf. COLING*, Dublin, Ireland, Aug. 23–29, 2014, pp. 1154–1164.

[13] A.F. Smeaton, F. Kelledy, and R. O'Donnell, "TREC-4 Experiments at Dublin City University: Thresholding Posting Lists, Query Expansion With WordNet and POS Tagging of Spanish," *Int. Conf. TREC-4*, Gaithersburg, USA, Nov. 1–3, 1995, pp. 373–389.

[14] F.H. Khan, U. Qamar, and S. Bashir, "SWIMS: Semi-Supervised Subjective Feature Weighting and Intelligent Model Selection for Sentiment Analysis," *Knowl. Based Syst.*, vol. 100, May 2016, pp. 97–111.

[15] Q. Gao and S. Vogel, "Corpus Expansion for Statistical Machine Translation with Semantic Role Label Substitution Rules," *Int. Conf. ACL-HLT*, Portland, OR, USA, June 19–24, 2011, pp. 294–298.

[16] R. Bhagat and E. Hovy, "What is a Paraphrase?," *Comput. Linguistics*, vol. 39, no. 3, 2013, pp. 463–472.

[17] B. Dolan, C. Quirk, and C. Brockett, "Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources," *Int. Conf. COLING*, Geneva, Switzerland, Aug. 23–27, 2004, pp. 350–356.

[18] R. Socher et al., "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection," *Int. Conf. NIPS*, Granada, Spain, Dec. 12–17, 2011, pp. 801–809.

[19] S. Zhao et al., "Application-Driven Statistical Paraphrase Generation," *Int. Conf. ACL-IJCNLP*, Suntec, Singapore, Aug. 2–7, 2009, pp. 834–842.

[20] M. Negri et al., "Chinese Whispers: Cooperative Paraphrase Acquisition," *Int. Conf. LREC*, Istanbul, Turkey, May 21–27, 2012, pp. 2659–2665.

[21] A. Sordoni et al., "A Neural Network Approach to Context-Sensitive Generation of Conversational Responses," *Int. Conf. NAACL-HLT*, Denver, CO, USA, May 31–June 5, 2015, pp. 196–205.

[22] O. Vinyals and V.L. Quoc, "A Neural Conversational Model," *Int. Workshop Deep Learning*, Lille, France, July 10–11, 2015.

[23] K.M. Hermann et al., "Teaching Machines to Read and Comprehend," *Int. Conf. NIPS*, Montréal, Canada, Dec. 7–12, 2015, pp. 1693–1701.

[24] D. Zhang and G. Lu, "Evaluation of similarity measurement for image retrieval," *Int. Workshop NNSP*, Toulouse, France, Sept. 17–19, 2003, pp. 928–931.

[25] J.R. Smith, "*Integrated Spatial and Feature Image System: Retrieval, Analysis and Compression*," Ph.D. Dissertation, School arts Sci., Columbia Univ., 1997.

[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, 1997, pp. 1735–1780.

[27] K. Greff et al., "LSTM: A Search Space Odyssey," arXiv preprint arXiv:1503.04069, 2015.

[28] I. Sutskever, O. Vinyals, Q.V. Le, "Sequence to sequence learning with neural networks," *Int. Conf. NIPS*, Montréal, Canada, Dec. 8–13, 2014, pp. 3104–3112.

[29] K. Cho et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *Int. Conf. EMNLP*, Doha, Qatar, Oct. 25–29, 2014, pp. 1724–1734.

[30] A.M. Dai and Q.V. Le, "Semi-Supervised Sequence Learning," *Int. Conf. NIPS*, Montréal, Canada, Dec. 7–12, 2015, pp. 3079–3087.

[31] R. Kiros et al., "Skip-Thought Vectors," *Int. Conf. NIPS*, Montréal Canada, Dec. 7–12, 2015, pp. 3294–3302.

[32] J. Li et al., "A Diversity-Promoting Objective Function for Neural Conversation Models," *Int. Conf. NAACL-HLT*, San Diego, CA, USA, June 12–17, 2016, pp. 110–119.

[33] J. Zhao, M. Lan, and J.F. Tian, "ECNU: Using Traditional Similarity Measurements and Word Embedding for Semantic Textual Similarity Estimation," *Int. Workshop on Semantic Evaluation*, Denver, Colorado, June 4–5, 2015, pp. 117–122.

[34] F. Jelinek, R.L. Mercer, L.R. Bahl, and J.K. Baker, "Perplexity—A Measure of Difficulty of Speech Recognition Tasks," *94th Meet. Acoustical Society of America*, Miami Beach, FL, Dec. 15, 1977.

[35] B. Harb et al., "Back-off Language Model Compression," *Int. Conf. INTERSPEECH*, Brighton, United Kingdom, Sept. 6–10, 2009, pp. 353–355.

[36] E. Chung et al., "Domain-Adapted Word Segmentation for an Out-of-Domain Language Modeling," *Int. Workshop on Spoken Dialog Systems*, Granada, Spain, Sept. 1–3, 2011, pp. 63–73.

[37] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[38] A. Stolcke, "SRILM – an Extensible Language Modeling Toolkit," *Int. Conf. Spoken Language Processing*, Denver, Colorado, Sep. 16–20, 2002, pp. 901–904

[39] D. Rey, and M. Neuhäuser, "Wilcoxon-signed-rank test," *International Encyclopedia of Statistical Science*, Springer Berlin Heidelberg, 2011, pp. 1658–1659.

[40] B.J. Hsu, "Generalized linear interpolation of language models," *Int. Workshop ASRU*, Kyoto, Japan, Dec. 9–13, 2007, pp. 136–140.

Euisok Chung received his BS degree in computer science from Soongsil University, Seoul, Rep. of Korea, in 1997, and his MS degree in computer science from Yonsei University, Seoul, Rep. of Korea, in 1999. Since 1999, he has worked with ETRI, Daejeon, Rep. of Korea. His current research interests include natural language processing, machine learning, and spoken dialogue systems.

Jeon Gue Park received his PhD in information and communication engineering from Paichai University, Daejeon, Rep. of Korea in 2010. He has worked with ETRI, Daejeon, Rep. of Korea since 1991, Lernout and Hauspie (L&H) since 2000, and Donga Seetech Inc. since 2002. He rejoined ETRI in 2004. He is currently in charge of the Spoken Language Processing Research Section, ETRI. His current research interests include artificial intelligence, computer-assisted language learning, spoken dialogue systems, and cognitive systems.

https://doi.org/10.4218/etrij.17.0116.0074