JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 11, pp. 27-40, November 2020
https://doi.org/10.9708/jksci.2020.25.11.027

The Ability of L2 LSTM Language Models to Learn the Filler-Gap Dependency

Euhee Kim*

*Professor, Dept. of Computer Science & Engineering, Shinhan University, Gyeonggi-do, Korea

[Abstract]

In this paper, we investigate the correlation between the amount of English sentences that Korean English learners (L2ers) are exposed to and their sentence processing patterns by examining what Long Short-Term Memory (LSTM) language models (LMs) can learn about implicit syntactic relationship: that is, the filler–gap dependency. The filler–gap dependency refers to a relationship between a (wh-)filler, which is a wh-phrase like ‘what’ or ‘who’ overtly in clause-peripheral position, and its gap in clause-internal position, which is an invisible, empty syntactic position to be filled by the (wh-)filler for proper interpretation. Here to implement L2ers’ English learning, we build LSTM LMs that in turn learn a subset of the known restrictions on the filler-gap dependency from English sentences in the L2 corpus that L2ers can potentially encounter in their English learning. Examining LSTM LMs’ behaviors on controlled sentences designed with the filler-gap dependency, we show the characteristics of L2ers' sentence processing using the information-theoretic metric of surprisal that quantifies violations of the filler-gap dependency or wh-licensing interaction effects. Furthermore, comparing L2ers’ LMs with native speakers’ LM in light of processing the filler-gap dependency, we not only note that in their sentence processing both L2ers’ LM and native speakers’ LM can track abstract syntactic structures involved in the filler-gap dependency, but also show using linear mixed-effects regression models that there exist significant differences between them in processing such a dependency.

▸ **Key words**: LSTM language model, English sentence processing, filler-gap dependency, surprisal, linear mixed-effects regression model, wh-licensing interaction effects

• First Author: Euhee Kim, Corresponding Author: Euhee Kim
*Euhee Kim (euhkim@shinhan.ac.kr), Dept. of Computer Science & Engineering, Shinhan University
• Received: 2020. 10. 06, Revised: 2020. 11. 11, Accepted: 2020. 11. 14.

Copyright © 2020 The Korea Society of Computer and Information
http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

[요 약]

본 논문은 장단기기억신경망(LSTM)이 영어를 배우면서 학습한 암묵적 통사 관계인 필러-갭 의존 관계를 조사하여 영어 문장 학습량과 한국인 영어 학습자(L2ers)의 문장 처리 패턴 간의 상관관계를 규명한다. 이를 위해, 먼저 장단기기억신경망 언어모델(LSTM LM)을 구축하였다. 이 모델은 L2ers가 영어 학습 과정에서 잠재적으로 배울 수 있는 L2 코퍼스의 영어 문장들로 심층학습을 하였다. 다음으로, 이 언어 모델을 이용하여 필러-갭 의존 관계 구조를 위반한 영어 문장을 대상으로 의문사 상호작용 효과(wh-licensing interaction effect) 즉, 정보 이론의 정보량인 놀라움(surprisal)의 정도를 계산하여 문장 처리 양상을 조사하였다. 또한 L2ers 언어모델과 상응하는 원어민 언어모델을 비교 분석함으로써, 두 언어모델이 문장 처리에서 필러-갭 의존 관계에 내재된 추상적 구문 구조를 추적할 수 있음을 보여주었을 뿐만 아니라, 또한 선형 혼합효과 회귀모델을 사용하여 본 논문의 중심 연구 주제인 의존 관계 처리에 있어서 원어민 언어모델과 L2ers 언어모델 간 통계적으로 유의미한 차이가 존재함을 규명하였다.

▸ **주제어**: 장단기기억신경망 언어모델, 영어 문장처리, 필러-갭 의존관계, 놀라움, 선형 혼합효과 회귀모델, 의문사 상호작용효과

I. Introduction

The language model (LM) based on an algorithm like Recurrent Neural Network (RNN), which is rapidly developing in the field of recent Natural Language Processing (NLP) tasks, conceptually correspond to a language acquisition device that has frequently been mentioned in the literature on the language acquisition and learning theory in the generative grammar research. That is, by applying theory and practice, particularly through the development of an LM that models the sentence processing of Korean English learners (or L2ers) and the analysis for the results of its performances, we conduct an integrated computational psycholinguistic study of sentence processing[1].

Sentence processing, in which a sentence is the smallest unit of information transfer in foreign language use, is essential for understanding foreign language learners' use of the language they learn. Psycholinguistics has formulated various principles that explain the generalizations drawn from sentence processing.

The basic question we can ask is : When Korean English learners process a specific sentence in English, how does the processing pattern of this sentence (that is, the pattern in which the principles bearing on sentence processing are applied) have a correlation with the accumulated language learning experiences they have had? For example, assuming that English sentence processing is correlated with the learner's language learning experience English sentences or data, it is a question whether it can be verified computationally.

In the field of computational psycholinguistics that seeks to answer such a question, deep neural network-based LMs that reproduce language learning experiences and language processing mechanisms of language learners can serve as an excellent tool to investigate the language processing patterns of foreign language learners[2].

In this paper, we aim to shed lights on such an important issue related to learning English as a foreign language by using neural LMs to investigate how well deep neural network LMs perform on particular types of controlled sentences that indicate a representation of a syntactic dependency. Specifically, we investigate

the correlation between the amount of English sentences that L2ers experience and their sentence processing patterns by examining what RNN LMs learn about implicit syntactic relationship: that is, the filler-gap dependency.

When training an LM with L1 (native speaker) or L2 (like Korean English leaner) English sentences to model learners with different levels of English knowledge, the number of learning (that is, epoch) is also adjusted differently. Likewise, we also modulate the size of training data for L1 and L2 LMs. Thereafter, it is possible to inspect the correlation of the LMs' performances with the patterns of sentence processing collected for human L2ers as well as L1ers.

To this aim, we are going to build an LM based on the Long Short-Term Memory RNN (LSTM), an algorithm grounded on deep learning neural networks, which has been widely used to model LMs in the NLP field. In essence, this study adopts NLP's LM methodology, specifically to adapt and implement the computational psycholinguistic methodology. We thereby build a model that reproduces English learners' language experiences, by incorporating into it the learning methods and sentence processing methods[3-4].

The remainder of the paper is structured as follows. Section 2 reviews the relevant studies in greater details. Section 3 gives a detailed overview of the methods that are going to be employed in the LMs to be built. Section 4 proposes the LSTM-based language modelling system to estimate the ability of the L1 and L2 LMs to learn filler-gap dependency. Section 5 and 6 each outlines the experiments performed and reports the results of the experiments. Section 7 wraps with a conclusion.

II. Related works

LSTMs have recently achieved impressive results in large-scale tasks like language modeling for speech recognition and machine translation, as well as computational psycholinguistic modeling of human language processing. This suggests that LSTMs may learn to track grammatical structure even when trained on comparatively noisy natural language data[5-7].

Particularly, Linzen et al. evaluated the extent to which RNNs can approximate hierarchical structure in corpus-extracted natural language data. They tested whether RNNs can learn to predict English subject-verb agreement, a task generally thought to require hierarchical structure. They argued that the supervised language modeling method is not sufficient for RNNs to deploy the syntactic knowledge necessary to cope with constructions where subject and verb agreement occurs in a long-distance fashion, as in (1a) and (1b): the symbol '**' means that the given sentence is ungrammatical[5].

(1a) The girl that the boys like is . . .
(1b) **The girl that the boys like are . . .

Gulordava et al. went on to focus on the more interesting unsupervised setup, where RNNs are trained to perform large-scale language modelling with subject-verb agreement. They demonstrated that RNNs trained with an LM objective can solve the long-distance agreement problem. Moreover, the performance of RNNs on language modeling (measured in terms of perplexity) is a good predictor of long-distance agreement accuracy. This suggests that the ability to capture structural generalizations is an important aspect of what makes the best RNN architectures excel in language modeling. Illustratively, like Linzen et al., they focused on long-distance agreement, where an arbitrary number of words can occur between the elements that engage in the subject-verb agreement relation. They concentrated on number (plural or singular) agreement, as in (2a) and (2b)[6].

(2a) The girl (that) [you met yesterday] thinks. . .
(2b) **The girl (that) [you met yesterday through her friends] thinks. . .

Meanwhile, Wilcox et al. investigated whether RNNs can learn English long-distance filler-gap dependency and constraints on it. In sentence (3a), the (wh-)filler is 'what' in clause-edge position, and the gap is placed after the transitive verb 'devoured', the gap position being indicated with an underline. If the filler were not present, the gap would be ungrammatical, as in (3b)[7].

(3a) I know what the lion devoured _____ at sunrise.
(3b) **I know that the lion devoured _____ at sunrise.

In this paper, following the lead by Wilcox et al. but concentrating on not L1 but L2 LMs, we investigate whether LSTM LMs built on the L2 corpus can truly learn filler-gap dependency.

To this aim, we examine the filler-gap effects at the final gap position. As pointed out above, the wh-filler gap dependency in L2ers is undoubtedly an ideal test bed to identify the syntactic ability that LSTM LMs come to entertain.

To anticipate the test data to evaluate the syntactic ability of LSTM LMs, we use two sets of a pair of conditions; (i) Long vs. Short condition and (ii) VP vs. NP condition in Table 1. The difference between the former Long and Short condition lies in the presence/absence of the intermediate gap immediately before the subordinator 'that', though the two conditions differ in light of the distance between the wh-filler and its gap, as (4a)-(4b).

On the other hand, the difference between the latter VP and NP condition also lies in the presence/absence of the intermediate gap right before the subordinator 'that', but the two conditions here are almost similar in light of the distance between the wh-filler and its gap, as (4c)-(4d).

Table 1. Two sets of a pair of conditions

Condition	Definition
(4a) Long	**wh-filler . . . intermediate gap 'that' . . . final gap**
(4b) Short	**wh-filler . . . final gap**
(4c) VP	**wh-filler . . . intermediate gap 'that' . . . final gap**
(4d) NP	**wh-filler . . . . final gap**

Following Wilcox et al., we are going to quantify the improper filler-gap dependency effects using the information-theoretic measure of surprisal (to be defined below). Thus, in both (4a) vs. (4b) and (4c) vs. (4d), we measured the surprisal on a preposition right after the final gap, because the preposition right after the final gap can reflect both 'no filler effect' and 'no gap effect', which amount to the improper filler-gap dependency effects at issue.

III. Methods

1. LSTM Language Models

A language model is a statistical model that assigns probabilities to words or sentences. We typically try to predict the next word in a sentence given all the previous words, often referred to as "context". For example, given the context "For dinner I'm making_____", what's the probability that the next word is "plans"? What's the probability that the next word is "pizza"? It is the most likely that p(pizza| For dinner I'm making) > p(plans| For dinner I'm making), where p stands for the conditional probability.

In this paper, we used the best model with the lowest perplexity for L1 language modelling, which is the pre-existing LSTM LM trained on an L1 corpus of English sentences. The model generally known as the Gulordava model was selected for its previous success in learning the subject-verb number agreement task. It was pre-trained on about 90M word tokens of English Wikipedia by Gulordava et al.[6].

As pointed out above, we also designed two LSTM language models on the two L2 corpora (to be specified below) to model L2ers with different levels of English knowledge by adapting the same Gulordava model architecture, shown in Figure 1.

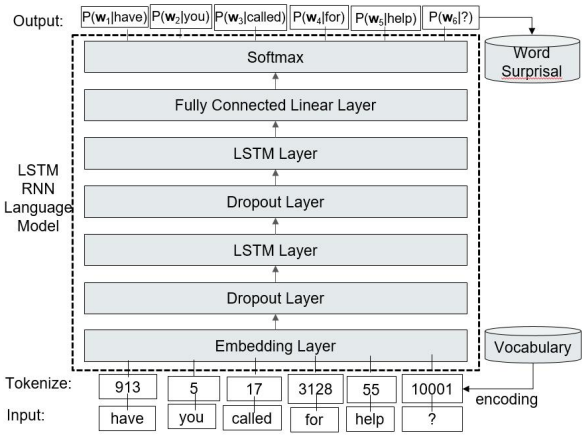

Fig. 1. LSTM network architecture for language modeling

As is now well known, the LSTM LM represents a word-to-word prediction LM composed of multiple layers: (1) *Tokenize*: splits input sentences into words tokens (integers) based on white space in sentences, and generates two symbols "<eos>" and "<unk>". This is not a layer for the LSTM network but a mandatory step of converting words into tokens; (2) *Input Layer*: tokenizes input sentences like "have you called for help?" with an extra unit that represents <eos>, that is, the end of a sentence (eos); (3) *Embedding Layer*: converts word tokens into embedding of specific size; (4) *LSTM Layer*: is defined by hidden state dimension and the number of layers; (5) *Dropout Layer*: is used to prevent over-fitting during the learning process; (6) *Fully Connected Linear Layer*: maps the output of the final LSTM layer to a desired output size; (7) *Softmax Layer*: turns all output values in a probability distribution; (8) *Output Layer*: The Softmax output from the previous time step to the next time step is considered as the final output of this network.

In Figure 1, word tokens (integers) from the input layer are fed into the embedding layer and then the two layers of LSTM cells. We define the shape of the inputs using the batch size and maximum-length of a sentence. The LSTM network module is shown to be rolled over all the time steps. Each input unit corresponds to one word, making maximum input units fix maximum-length words in the input sentence with variable length.

The proposed LSTM networks, which are built upon PyTorch, are refer to L2 model and L3 model to learn filler-gap dependencies for L2ers. On the other hand, we used the pre-trained Gulordava model for L1 model. We reported the hyper-parameters and further model details, which are used for LSTM LM implementation (to be defined in section V).

The implementation of the L2 LM (built on English textbooks for L2ers) and L3 LM (built on EBS-CSAT English Prep Books as well as English textbooks for L2ers) respectively consists of three major phases such as sentence preprocessing, LM training, and LM evaluating:

(1) *The sentences preprocessing phase*: To input English sentences, first uppercase letters were converted to lowercase letters, and only alphabet letters was extracted by removing special characters and symbols. As in Figure 2, the function *create_vocabulary* took the cleaned sentences as input and generated the Vocabulary database for each model; after splitting each sentence with symbol <eos> using a white space, the Vocabulary database for each model included unique lower case word tokens and special symbols <eos> and <unk>.

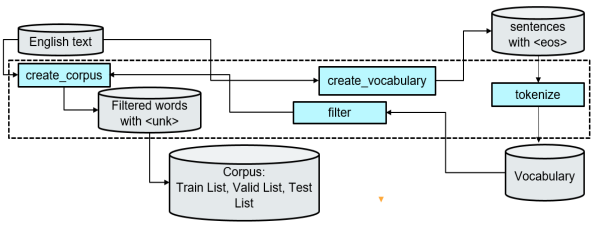

Fig. 2. The English sentence preprocessing phase

The function *create_corpus* took the cleaned sentences as input and generated the Corpus database for each model; After having filtered unknown words in sentences which do not belong to the Vocabulary database with symbol <unk> through the function *filter*, the Corpus database came to consist of Train list, Valid list, and Test list.

(2) *The LM training phase*: Figure 3 shows the deep training process of the LM based on the LSTM network architecture shown in Figure 1. The function *get_batch* took the Corpus database with Train and Valid lists as input and generated the targets and data (that is, inputs) encoded by one-hot vectors.

These inputs were fed into the Embedding layer which generated dense vectors. In the flow of the *Forward* operation, for each time-step the LSTM cell took both the embedding vectors and the hidden state weights of the previous time-step and returned the new hidden state weights as an output. The outputs passed through the fully Connected linear layer and the Softmax layer.

From the function *evaluate_loss*, the train/valid loss was calculated by comparing them with the targets using the loss function, that is, CrossEntropyLoss. In the flow of the *Backward* operation, Backpropagation was performed on the loss function. Iterative learning of *Forward* and *Backward* operations was performed in batch units by adjusting learning rates. Finally, using Adam's optimizer, the optimized models for the two models (that is, L2 and L3 LMs) were saved.

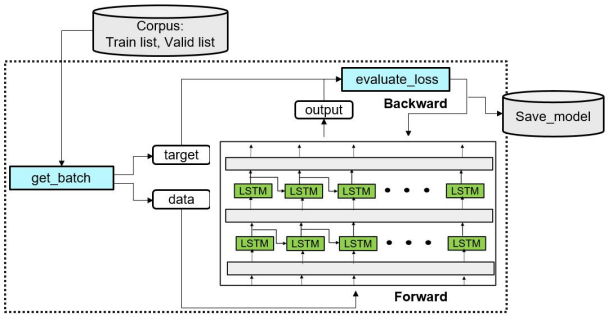

Fig. 3. The LSTM LM training phase

(3) *The LSTM LM evaluating phase*: The Figure 4 shows that the evaluating process of the LSTM language model learned the filler-gap dependency in question. The function *get_batch* took the Corpus database with test sentences as input and generated the targets and data encoded by one-hot vectors.

Then, the three optimized pre-trained L1, L2, and L3 models respectively was loaded. Each model took these inputs and generated the outputs passed through the Softmax layer. We used the Softmax function to calculate the probability distribution predicting the corresponding test words.

In addition, the loss value was calculated by comparing the Softmax result value and the target value through the CrossEntropy loss function, and the surprisal for a word at issue was calculated by the Softmax results.

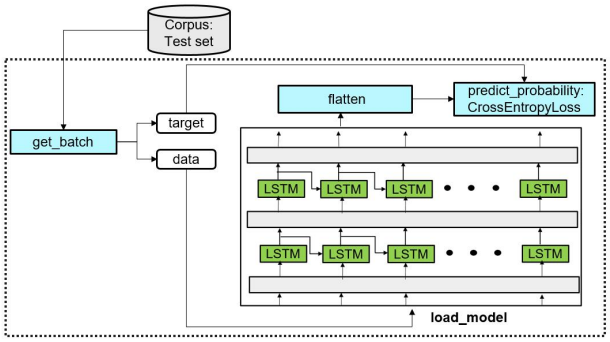

Fig. 4. The LSTM LM evaluating phase

2. Perplexity for language model

Perplexity is an evaluation metric used to judge how good a language model is. For example, if we have a perplexity of 100, it means that whenever the model is trying to guess the next word, the model is as confused as if it had to pick between 100 words. Perplexity(ppl) is defined as follows:

$$ppl(w) = 2^{-\frac{1}{n}log_2 P(w_1, w_2, ..., w_n)}, \quad (1)$$

where P is a trained language model, $w = (w_1, w_2, ..., w_n)$ is a sequence of n words,

$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} p(w_i)$ is the probability of a sentence of words given by a product. The logarithm is taken in base 2.

The following Table 2 shows the perplexity as in the above formula (1) of each of the three LSTM LMs (that is, L1, L2, and L3 models) on the valid set.

Table 2. The perplexity of the three LSTM LMs

LSTM LM	L1(Gulordava)	L2	L3
ppl	52.1	46.26	47.40

A caveat is in order. In previous work, perplexity and grammatical judgment accuracy have been reported to be partly dissociable[8-9].

3. Surprisal for information density

The measure of surprisal is widely used in information-theoretic modelling of human language. For example, consider the following English sentence: “She went to the grocery shop to buy some apples and ( ).” To fill in the appropriate word for the blank ( ), we predict that ‘pears’ is a good continuation after ‘apples’, while for instance ‘books’ is not. Language models provide the following outputs: assigning a low probability and hence a high surprisal value to the word ‘books’, but assigning a high probability and hence a low surprisal value to the word ‘pears’.

If we successively obtain a surprisal value for each word of a sentence given the preceding words, we get an information density profile of that sentence. If a word is highly unexpected in the sentence containing that word, it will lead to a high surprisal value. It is known to correlate directly with human sentence processing difficulty[10-12].

For a certain word, surprisal is the negative log-likelihood of this word given an LSTM’s hidden state before taking it:

$$S(w_i) = -\log_2 p(w_i|h_{i-1}),$$

where for w_i of a sentence’s i^{th} word, h_{i-1} is the LSTM’s hidden state before encountering w_i. We calculated surprisal values from the LSTM LMs by directly computing the negative log of the predicted conditional probability from the Softmax layer in Figure 1.

In this paper, we investigated the LSTM LMs’ learning about wh-filler and gap dependency by measuring the effects that the wh-filler has on the gap. That is, we evaluated the surprisal value for a word immediately following the gap when there is a gap. We looked for the cases where the surprisal value associated with an unusual sentence containing a gap is reduced by the presence of its wh-filler. Thus, if the LMs learn that syntactic gaps require wh-filler, then sentences with a proper wh-filler are expected to yield lower surprisal than minimally different sentences that lack a proper wh-filler.

4. Wh-Licensing Interaction for filler-gap dependencies effects

We tested whether LSTM LMs have learned filler-gap dependency by examining a 2x2 interaction between a wh-filler and a gap. The interaction between a wh-filler and a gap represents the extent to which either a wh-filler or its gap ends up not being properly associated or licensed, thereby ‘no gap’ or ‘no filler’ effects arising maximizing the surprisal. We term the effects of both ‘no gap’ and ‘no filler’ more specifically the wh-licensing interaction effects[7].

We measured such effects, by examining four minimally varying sentences, given the examples in Table 3 that contain the four possible combination of (no) fillers and (no) gaps in specific syntactic positions. Note that the three symbols: (i) ■ (i.e., the final gap position where the wh-filler is integrated for proper interpretation), (ii) □ (i.e., the wh-filler position), and (iii) ○ (i.e., the intermediate gap position where the wh-filler drops by during its overt displacement) are added for presentational purposes only and were not included in test items. All four sentence variants in (5a-d) were created in order to compute the

wh-licensing interaction effects, viz. both no filler effects in (5c) compared to (5d) and no gap effects in (5b) compared to (5a):

Table 3. The four combination of fillers and gaps.

Sentence	Filler	Gap
(5a) The manager knew that the secretary claimed ○ that the new salesman had pleased the boss in the meeting.	No	No
(5b) **The manager knew who the secretary claimed ○ that the new salesman had pleased the boss in the meeting.	Yes	No
(5c) **The manager knew that the secretary claimed ○ that the new salesman had pleased ■ in the meeting.	No	Yes
(5d) The manager knew which boss the secretary claimed ○ that the new salesman had pleased ■ in the meeting.	Yes	Yes

If a wh-filler sets up an expectation for the presence of a gap, then in sentences containing a wh-licensor but no gap as in (5b) we expect higher surprisal in syntactic positions. That is, $S(b) - S(a)$ should be a large positive number. The presence of a gap in the absence of a wh-licensor as in (5c) should also result in higher surprisal than when the wh-licensor is present. That is, $S(d) - S(c)$ should be a large negative number. We can assess how well the model has learned both expectations by adding up the two types of effects:

$$[S(b) - S(a)] - [S(d) - S(c)],$$

which result in improper filler-gap dependency or wh-licensing interaction effects.

To reiterate, if the LSTM LMs learn the filler-gap dependency, we expect wh-licensing interaction effects to amount to a large positive number in surprisal. In other words, a positive surprisal value of the wh-licensing interaction effects means that the model is able to learn a filler-gap dependency between a wh-filler and its gap. Conversely, a negative surprisal value of the wh-licensing interaction effects points to the fact that the model is not able to[7].

5. Mixed-effects linear regression model for wh-licensing interaction effects

If the LSTM LM learns a filler-gap dependency between a wh-word and its gap, we expect wh-licensing interaction effects to be a positive number in surprisal. For example, to ascertain whether 'no gap' or 'no filler' effects in Table 3, as a fixed effect, lead to wh-licensing interaction effects, we controled for variation between the different sentences. To do that, we derive the statistical significance of the wh-licensing interaction effects from a mixed-effects linear regression model predicting surprisal given conditions on the wh-filler gap dependency[13-16].

Modelling the conditions on the filler-gap dependency and three LMs respectively as fixed effects and treating the sentences as random effects, we formulated our mixed-effects linear regression model as follows:

$$surprisal = \alpha * Conditions * Model + \beta * Sentences + \epsilon, \quad (2)$$

where ϵ is an error term, α is a coefficient for the fixed effects and interaction effects term, and β is a coefficient for random effects term. But we omitted random slopes as we did not have repeated observations within sentences and conditions. We fitted our mixed effect model using the statsmodels packages in Python.

IV. L2 LSTM language modelling system with the ability to represent filler-gap dependency

In this section, we finally propose the LSTM-based language modelling system to estimate the ability of the L2 LM to learn filler-gap dependency by using the wh-licensing interaction effects.

As shown in Figure 5, after sentences pre-processing, the test sets (lists of sentences with filler-gap dependency) were taken to the

Gulordava model (L1 model) and the other two models (that is, L2 model and L3 model) as inputs. The wh-licensing interaction values were calculated by employing surprisal values extracted from each model.

In the above formula (2) for the linear mixed-effects regression model, the dependent variable was taken as surprisal (that is, wh-licensing interaction) values, and independent variables for fixed effects were the different conditions representing filler-gap dependency. Sentence index, model type, and sentence level were used as other variables of the linear mixed regression model. This was a good model because it modeled the variation that was present for each gap and each filler.

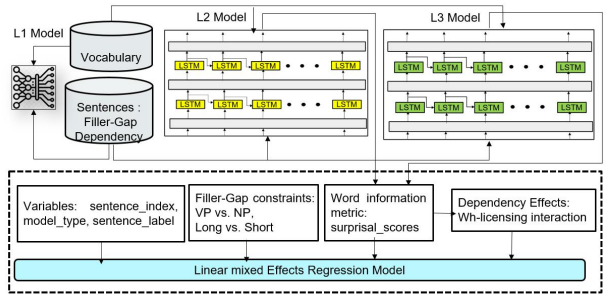

Fig. 5. The Workflow for the proposed system

V. Experiment

In this section, we introduce the experimental environment setup for the two LSTM language models for L2 and the pre-trained Gulordava language model for L1.

1. Experimental Environment

We present the hyper-parameters that were used to train the LSTM LM network on the L2 and L3 corpora respectively. For the L1 model, we used the pre-trained LSTM LM (i.e., the Gulordava model), which achieved the lowest perplexity on their valid dataset.

The layer_number is the number of the LSTM layers; the hidden_size is the dimension of the hidden layer; the learning_rate is a turning parameter in the optimization algorithm that determines the step size at each iteration, while approaching the minium of the loss function; the epoch denotes the number of times that training is performed; and the batch_size denotes the number of training words that feed into the input layer in each epoch. For the three models in question we report the following hyper-parameters in Table 4.

Table 4. Hyper-parameters for the three models

Parameter	L1	L2	L3
Network	LSTM	LSTM	LSTM
vocab_size	100M	40026	41747
Layer_number	2	2	2
Hidden_size	650	200	200
Learning_rate	0.2	0.2	0.2
Batch_size	128	20	20
Epoch	40	40	40
Embedding_size	650	200	200
Dropout_rate	0.2	0.2	0.2

The experimental environment comprised Windows 10, i5-6400, NVIDIA Geforce GTX 1050 2GB, and DDR4 8GB. The proposed LSTM RNN network was implemented in Python using PyTorch Framework.

2. Training data for LSTM LMs

The L1 corpus for the pre-trained L1 model was extracted from the Wikipedia. The L2 corpus for the L2 model is a collection of all the sentences extracted from English textbook corpora based on the English 11 middle-school and 12 highschool textbooks published in Korea in 2001 and the English 19 middle-school and 12 highschool textbooks published in Korea in 2009. The L3 corpus for the L3 model is a collection of all the sentences extracted from EBS-CSAT English Prep Books published in Korea in 2016~2018, on top of the L2 English textbook corpora mentioned above.

All the training sets were shuffled at sentence level, and we filtered out sentences with more than 5% unknown words. Each training set was split into a training subset and a validation subset at an 8:1 ratio. Each model used the most

frequent words in the training Corpus, replacing the other tokens with the <unk> symbol in the Vocabulary.

3. Test data for LSTM LMs

In this section, we report the design of the test set for the three LSTM models to measure word surprisal to estimate filler-gap dependency. We investigated whether LSTM LMs model the probabilistic dependency between fillers and gaps at all. We introduce the two cases in point which represent how well the three LMs with the LSTM networks learn the wh-licensing relationship. The relationship is to represent the filler-gap effects at the final gap position. It is specifically represented in the four sentence conditions of filler-gap dependency: that is, Long vs. Short condition and VP vs. NP condition. A positive wh-licensing interaction means that a model can track a filler-gap dependency between the wh-word and the gap sites; a negative licensing interaction means that no such dependency can be acquired by it.

Case 1 : Long vs. Short condition.

We finally examined whether LSTM LMs can learn constraints on filler-gap dependencies by comparing the wh-licensing interactions in Long condition with those in Short condition.

To measure the difference between Long and Short conditions at the final gap position, we designed the 60 sets of four test conditions (30 Long and 30 Short condition sets) for the three models L1, L2, and L3, in a 2 x 2 design for a total of 240 sentences, as in Table 5. For each set we produced four variant conditions, as described above. In the following set of four conditions, the filler (□) is 'who' or 'which', and the gap(■) is placed before the preposition 'before'. The symbol ○ is the intervening gap position that the wh-filler drops by as it searches for its final gap.

In the four Short condition sentences, the distance between filler and gap is shorter than in the Long wh-condition counterparts, so when the filler finds ■, less surprisal is expected to arise.

We extracted the Softmax values of the post-gap region items (such as the preposition 'before'). When the LSTM LMs learned the wh-licensing of the filler-gap dependency, then we predicted to discover wh-licensing interaction effects at the final gap that might vary in terms of the distance between a wh-filler and its gap.

Table. 5. Test sentences for Short and Long conditions

Level	Condition	Sentence	Filler	Gap
a	Short	The townspeople hoped that the cameraman knew that the former mayor would honor the soldiers before the fireworks.	No	No
b	Short	The townspeople hoped that the cameraman knew **who** the former mayor would honor the soldiers before the fireworks.	Yes	No
c	Short	The townspeople hoped that the cameraman knew that the former mayor would honor ■ before the fireworks.	No	Yes
d	Short	The townspeople hoped that the cameraman knew **which soldiers** the former mayor would honor ■ before the fireworks.	Yes	Yes
a	Long	The cameraman knew that the townspeople hoped ○ **that** the former mayor would honor the soldiers before the fireworks.	No	No
b	Long	The cameraman knew **who** the townspeople hoped ○ the former mayor would honor the soldiers before the fireworks.	Yes	No
c	Long	The cameraman knew that the townspeople hoped ○ that the former mayor would honor ■ before the fireworks.	No	Yes
d	Long	The cameraman knew **which soldiers** the townspeople hoped ○ that the former mayor would honor ■ before the fireworks.	Yes	Yes

Case 2 : VP vs. NP condition.

In addition, we also examined whether LSTM LMs can learn the syntactic role of an

intermediate gap in filler-gap dependency by comparing the wh-licensing interaction effects in the Noun Phrase (NP) conditions with those in Verb Phrase (VP) conditions.

When the LSTM LMs learned the presence of the intervening gap in the wh-licensing of the filler-gap dependency, then we predicted to find wh-licensing interaction effects at the final gap position.

To test the embedding clause of the model's filler-gap dependency representation, we designed the 40 test items for the three models L1, L2, and L3, in a 2 x 2 design for a total of 160 sentences, as in Table 6.

In Table 6, the NP conditions are comparatively similar in filler-gap distance to the VP conditions. But the latter conditions have an intermediate gap position that the wh-filler drops by to find the final gap; this is represented as ◯. It has been generally noted in theoretical studies of grammar that the surprisal value increases if ◯ is present; otherwise, it deceases. For 4 variant levels of each condition, we measured the wh-licensing interaction effects in the post-gap items right after the final gap position.

Table 6. Test sentences for VP and NP conditions

Level	Condition	Sentence	Filler	Gap
a	VP	The manager knew that the secretary claimed ◯ **that** the new salesman had pleased the boss in the meeting.	No	No
b	VP	The manager knew **who** the secretary claimed ◯ that the new salesman had pleased the boss in the meeting.	Yes	No
c	VP	The manager knew that the secretary claimed ◯ that the new salesman had pleased ■ in the meeting.	No	Yes
d	VP	The manager knew **which boss** the secretary claimed ◯ that the new salesman had pleased ■ in the meeting.	Yes	Yes
a	NP	The manager knew that the secretary's claim about the new salesman had pleased the boss in the meeting.	No	No
b	NP	The manager knew **who** the secretary's claim about the new salesman had pleased the boss in the meeting.	Yes	No
c	NP	The manager knew that the secretary's claim about the new salesman had pleased ■ in the meeting.	No	Yes
d	NP	The manager knew **which boss** the secretary's claim about the new salesman had pleased ■ in the meeting.	Yes	Yes

VI. Results

In this section, the results for the two experiments with the test data described in Case 1 and Case 2 are to be reported. We demonstrated whether with the L1 LM as a baseline, the LSTM LMs for L1, L2, and L3 learned the difference of the two conditions (that is, Long vs. Short, VP vs. NP) of filler-gap dependency. To anticipate the results, we found a significant wh-licensing interaction resulting in fluctuation of surprisal for the conditions att issue.

The results from the experiments for Case 1 can be found in Figure 6. On the left three panels, between the three experiments shown left, we found a significant decrease in wh-licensing interaction effects for the Long condition, relative to the Short condition in both the L1 model and the L3 model. This indicates a significant reduction in surprisal (i.e., wh-licensing interaction) in the Long condition ($p < 0.005$) in Table 7. By contrast, in the L2 LM we find no significant decrease in surprisal (wh-licensing interaction) in the Long condition, relative to the Short condition.

The results from the experiments for Case 2 can be found on the right three panels in Figure 6, between the three experiments shown right. For L1 model we found a significant decrease in surprisal for the VP condition, compared to the NP condition. This indicates a significant reduction of surprisal

(wh-licensing interaction) for the VP condition ($p < 0.001$) in relation to the Long condition in Table 7. Conversely, in both L2 and L3 LMs we have found no significant decrease in surprisal for the VP condition, compared to the NP condition.

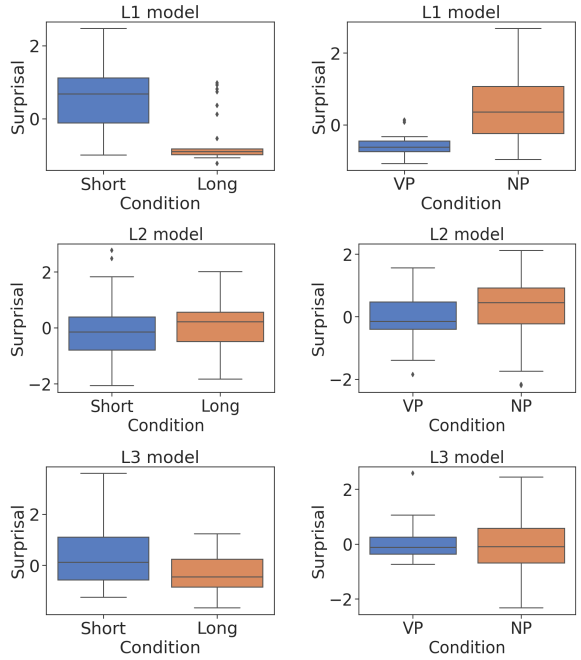

Fig. 6. Wh-licensing interaction effects for two cases

Table. 7 Correlations between conditions for each model

Case 1 for L1 model				
	Coefficient	STD	z-value	p>|z|
Intercept	-0.612	0.122	-5.013	0.000
C(condition)[T.Short]	1.224	0.211	5.804	0.000
Case 2 for L1 model				
	Coefficient	STD	z-value	p>|z|
Intercept	0.572	0.202	2.824	0.005
C(condition)[T.VP]	-1.143	0.217	-5.269	0.000
Case 1 for L2 model				
	Coefficient	STD	z-value	p>|z|
Intercept	0.044	0.160	0.272	0.786
C(condition)[T.Short]	-0.087	0.195	-0.447	0.655
Case 2 for L2 model				
	Coefficient	STD	z-value	p>|z|
Intercept	0.118	0.262	0.450	0.653
C(condition)[T.VP]	-0.236	0.294	-0.802	0.423
Case 1 for L3 model				
	Coefficient	STD	z-value	p>|z|
Intercept	-0.341	0.123	-2.772	0.006
C(condition)[T.Short]	0.682	0.209	3.263	0.001
Case 2 for L3 model				
	Coefficient	STD	z-value	p>|z|
Intercept	-0.046	0.229	-0.200	0.841
C(condition)[T.VP]	0.092	0.235	0.390	0.696

In addition, we went on to examine whether there exists a significant difference for the L1 vs. L2 models or the L1 vs. L3 models in the VP condition. The results from the experiments can be found in Figure 7. On the left panel, in the VP condition we found a significant reduction in surprisal of the L1 model, relative to the L2 model ($p < 0.005$). Besides, in the VP condition we also found a significant decrease in surprisal of the Ll model, compared to the L3 LM ($p < 0.001$), as shown in Table 8.

We then examined whether there exists a significant difference for the L1 vs. L2 models or the L1 vs. L3 models in the Long condition. The results from the experiments can be found on the right panel of Figure 7. In the Long condition we found a significant reduction in surprisal of the L1 model, compared to the L2 model ($p < 0.001$). In tandem, in the Long condition we found a significant decrease in surprisal of the L1 model in relation to the L3 model ($p < 0.001$), as shown in Table 8.

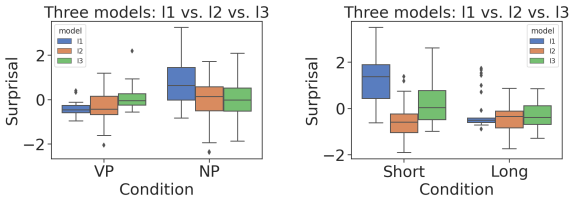

Fig. 7. Wh-licensing interaction effects for three models

Table 8. Correlations between conditions vs. three models

VP vs. NP				
	Coefficient	STD	z-value	p>|z|
Intercept	0.883	0.254	3.479	0.001
C(condition)[T.VP]:C(model)[T.l2]	1.059	0.366	2.895	0.004
C(condition)[T.VP]:C(model)[T.l3]	1.359	0.366	3.716	0.000
Long vs. Short				
	Coefficient	STD	z-value	p>|z|
Intercept	-0.169	0.117	-1.439	1.150
C(condition)[T.Short]:C(model)[T.l2]	-1.515	0.264	-5.733	0.000
C(condition)[T.Short]:C(model)[T.l3]	-0.948	0.264	-3.587	0.000

VII. Conclusions

In this paper, we investigated the correlation between the amount of English sentences that L2ers experience and their sentence processing patterns by examining what LSTM LMs implementing L2ers' English learning can learn about implicit syntactic relationship: that is, the filler-gap dependency.

Building the three L1, L2, and L3 LMs, we concentrated on the filler-gap dependency tasks to investigate the grammatical ability of neural language models trained on English sentences that different groups of learners learn English with. We showed that the L1 model can effectively track syntactic structure involved in the filler-gap dependency. Using linear mixed-effects models we went on to demonstrate that there are significant differences between the three models in light of the restrictions on filler-gap dependency. Particularly, we found that for the L3 model there was a significant difference between Long and Short wh-condition of filler-gap dependency. By contrast, for the L2 model there was no significant difference between the two conditions of Case 1 and Case 2.

In essence, investigating whether LSTM LMs are sensitive to different contexts involving filler-gap dependency, we showed that the language models display sensitivity to some but not all of the contexts at issue. Given the study reported in this paper, two issues are pending for future study. Since LMs are data-driven, we need to examine in greater details how the size of training data affects the ability of an LM to process language. Second, since the introduction of LSTM, other more effective algorithms like BERT and three versions of GPT have been introduced and their performances have been evaluated seriously. In this milieu, it is worth investigating how much the LMs built on more recent algorithms can contribute to understanding L2 language processing as well as L1 language processing.

ACKNOWLEDGEMENT

This work was supported by 2020 Shinhan Univ. Research Grant.

REFERENCES

[1] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," Machine learning, Vol. 7(2-3), pp. 195-225, Sep 1991.

[2] Y. Goldberg, "Neural network methods for natural language processing," Synthesis lectures on Human language Technologies, Vol. 10(1), pp. 1-309, Apr 2017.

[3] S. Hochreiter and S. Jurgen, "Long short-term memory," Neural Computation, Vol. 9(8), pp. 1735-1780, Nov 1997.

[4] E. Kim, "Sentence Comprehension with an LSTM Language Model," Journal of Digital Contents Society, Vol, 19(12), pp. 2393-2401, Dec 2018.

[5] T. Linzen, E. Dupoux, and Y. Goldberg, "Assessing the ability of LSTMs to learn syntax-sensitive dependencies," Transactions of the Association for Computational Linguistics, Vol. 4, pp. 521-535, Dec 2016.

[6] K. Gulordava, P. Bojanowski, E. Grave, T. Linzen, and M. Baroni, "Colorless green recurrent networks dream hierarchically," NAACL-HLT, pp. 1195-1205, Jun 2018.

[7] E. Wilcox, R. Levy, T. Morita, and R. Futell, "What do RNN Language Models Learn about Filler-Gap Dependencies?," ACL Anthology, Proceedings of the 2018 EMNLP Workshop Blackbox NLP: Analyzing and Interpreting Neural Networks for NLP, pp. 211-221, Aug 2019.

[8] A. Kuncoro, C. Dyer, J. Hale, D. Yogatama, S. Clark, and P. Blunsom, "LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better," Computational Linguistics, Vol. 1, pp. 1426-1436, Aug 2018.

[9] K. Tran, A. Bisazza, and C. Monz, "The importance of being recurrent for modeling hierarchical structure," In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Aug 2018.

[10] J. Hale, "Uncertainty about the rest of the sentence," Cognitive Science, Vol. 30(4), pp. 609-642, Jul 2006.

[11] J. Hale, "A probabilistic Earley parser as a psycholinguistic model," Proceedings of the Second meeting of the North American Chapter of the Association for Computational Linguistic and Language Technologies, pp. 1-8, Jun 2001.

[12] R. Levy, "Expectation-based syntactic comprehension," Cognition, Vol. 106(3), pp. 1126-1177, Mar 2008.

[13] R.H. Baayen, D.J. Davidson, and D.M. Bates, "Mixed-effects

modeling with crossed random effects for subjects and items," Journal of memory and language, Vol. 59(4), pp. 390-412, Mar 2008.

[14] E. Kim, M. Park, and W. Chung, "On Korean English L2ers' processing of wh-filler-gap dependencies: An ERP study," Language and Information , Vol. 21(3), pp. 1-24, Nov 2017.

[15] E. Kim, "A Deep Learning-based Article- and Paragraph-level Classification," The Journal of the Korea Society of Computer and Information, pp. 31-41, Nov 2018.

[16] E. Kim, "The Unsupervised Learning-based Language Modeling of Word Comprehension in Korean," The Journal of the Korea Society of Computer and Information, pp. 41-49, Nov 2019.

Authors

Euhee Kim received the M.S. degrees in Computer Engineering from Dongguk University, Korea, in 2002 and Ph.D. degrees in Mathematics from The University of Connecticut, U.S.A in 1995.

Dr. Kim is currently an associate Professor in the Department of Computer Science & Engineering at Shinhan University. She is interested in AI, NLP and Big Data computing.