JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 14, No. 4, December 2020, pp. 154-162

Semantic Vector Learning Using Pretrained Transformers in Natural Language Understanding

Sangkeun Jung*
Chungnam National University, Daejeon, Korea
hugman@cnu.ac.kr

Abstract

Natural language understanding (NLU) is a core technology for implementing natural interfaces. To implement and support robust NLU, previous studies introduced a neural network approach to learn semantic vector representation by employing the correspondence between text and semantic frame texts as extracted semantic knowledge. In their work, long short-term memory (LSTM)-based text and readers were used to encode both text and semantic frames. However, there exists significant room for performance improvement using recent pretrained transformer encoders. In the present work, as a key contribution, we have extended Jung's framework to work with pretrained transformers for both text and semantic frame readers. In particular, a novel semantic frame processing method is proposed to directly feed the structural form of the semantic frame to transformers. We conducted massive experiments by combining various types of LSTM- or transformer-based text and semantic frame readers on the ATIS, SNIPS, Sim-M, Sim-R, and Weather datasets to find the best suitable configurations for learning effective semantic vector representations. Through the experiments, we concluded that the transformer-based text and semantic frame reader show a stable and rapid learning curve as well as the best performance in similarity-based intent classification and semantic search tasks.

Category: Natural Language Processing

Keywords: Semantic vector; Semantic vector learning; Natural language understanding; Transformer

I. INTRODUCTION

Natural language understanding (NLU) is a primary technique for implementing natural user interfaces, such as chatbots, mobile secretaries, and smart speakers. NLU extracts meaning from natural language and infers the user's intention.

In addition to developing NLU systems, many assistant tools are required for building and maintaining these robust systems. For example, semantic search utility facilitates the acceleration of data processing at the collection and annotation stages. Re-ranking multiple NLU systems are crucial for maintaining stable performance over various types of input.

Jung et al. [1] demonstrated the importance of semantic vector representation in implementing key utilities around NLU systems. They introduced a neural architecture for embedding the correspondence between a text and a semantic frame. In their embedding semantic correspondence learning framework, correspondence (distance between projected vectors from a text and a semantic frame) was used as an objective score, whereas reconstruction (intent and slot-tag) was used as a generation cost.

In study of Jung [2], long short-term memory (LSTM)-

Open Access http://dx.doi.org/10.5626/JCSE.2020.14.4.154 http://jcse.kiise.org

Received 18 November 2020; **Accepted** 11 December 2020
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

based readers were used to encode sentences and semantic frames. However, there exists a significant room for performance improvement using recent pretrained encoders. In this work, as a key contribution, we extended Jung's framework to work with combinations of LSTM and pretrained transformers for both text and semantic frame readers. Recently, [3-5] proposed text and semantic frame reading methods have used BERT to construct semantic vector representations; while they used bidirectional encoder representations from transformers (BERT) [6] only readers. In the present work, we propose an integrated framework that combines LSTM and transformers to find the best configurations for learning effective semantic vector representations. We conducted various experiments by combining different types of text and semantic frame readers.

The remainder of this paper is organized as follows: Section II describes the overall architecture and semantic frame input processing methods for transformers. Section III presents the experimental results. Section IV discusses the related studies. Finally, Section V discusses the conclusions.

II. SEMANTIC VECTOR LEARNING WITH A PRETRAINED TRANSFORMER

Our framework comprises a text reader, semantic frame reader, and semantic frame writer [1, 2]. The text reader embeds a sequence of tokens to a distributed vector representation. The semantic frame reader reads the structured texts and encodes each text to a vector. Finally, the semantic frame writer generates a symbolic reduced-semantic-frame from a vector representation.

In this study, to encode a text and a semantic frame, we employed BERT and its variations as well as LSTM-based readers for semantic vector learning. To take advantage of pretrained transformers for semantic frame encoding, the sentence form of the semantic frame components is devised. In this work, we devised *a semantic frame sentence and reduced semantic frame sentence*.

A. Semantic Frame Sentence

Semantic frames are one of the most widely used structured representations of semantics in NLU and comprise *intent* and *slots*. The intent represents the sentence role in the target domain, whereas the slots represent domain-specific named entities or keywords. For example, "*Please list all flights from Dal las to LA*" can be represented by the following semantic frame:

- Intent tag: flight
- Slot tag: from.city, to.city
- Slot values: Dallas, LA

Semantic frame sentence: sentence form of a semantic frame with intent (slot_tag=slot_value, slot_tag=slot_value, ...) format. Its form is similar to that of user action [7]. A semantic frame sentence of the example above is expressed in the sentence form – "fl ight(from.city=Dallas, to.city=LA".

From the semantic frame sentence, a reduced semantic frame is created by excluding the slot values from the corresponding semantic frame, which was first introduced by [1].

B. Text Readers

In this study, a transformer-encoder-based text reader was used instead of the LSTM-based text reader of [2, 4].

Transformer text reader: Input text is tok-enized by a subword, and the tokens are fed to the transformer encoder (TRE). A sentence embedding vector from a [CLS] token is extracted from the final layer and linearly projected to the semantic vector dimension. The text encoding process can be defined as

$$\vec{h}_t = TRE_{text}([CLS],\ t_1,\ t_2,\ \ldots,\ t_k,\ [SEP])$$

$$v_t = sigmoid(W_T \vec{h}_t),$$

where $\vec{h}_t$ is the sentence embedding vector from TRE; t_k is a token at step k; [CLS] and [SEP] are class and separation tokens, respectively; W_T is a linear projection weight; and v_t is a semantic vector derived from the text. In this research, we use 100 as the final semantic vector dimension.

C. Semantic Frame Reader

Jung et al. [1, 2] introduced an LSTM-based hierarchical semantic frame reader that can capture structural information using multiple LSTM and multilayer perceptron. In our work, we introduce a simpler semantic frame reader using TRE, which directly encodes the sentence form of semantic frame components, such as reduced semantic frame and semantic frame sentences.

Similar to a text reader, a semantic frame sentence is tokenized by a subword tokenizer, and the tokens are fed to TRE. Tag symbols such as "weather.general" or "depart_date" are tokenized in a subword manner without considering semantic segmentation. For example, "depart_date" might be tokenized as ["depart_da", "te"] if "depart_da" is a more statistically common pattern in the dataset. A sentence embedding vector from the [CLS] token is extracted from the final layer and linearly projected to the semantic vector dimension.

The semantic frame sentence encoding process can be defined as

$$\vec{h}_s = TRE_{sf}([CLS],\ s_1,\ s_2,\ \ldots,\ s_k,\ [SEP])$$

$$v_s = sigmoid(W_S \vec{h}_s),$$

where $\vec{h}_s$ is the semantic frame sentence embedding vector from TRE; s_k is a token at step k; W_S is a linear projection weight; and v_s is a semantic vector derived from the semantic frame.

D. Semantic Frame Writer

To prevent zero-vector convergence, Jung et al. [1] proposed semantic frame writing to learn robust semantic representation. They introduced a simple linear-projection-based intent writer and a recurrent neural-network-based slot-tag writer to decode a reduced semantic frame, which is created by excluding the slot values from the corresponding semantic frame (more details in [2]). We used the same semantic frame architecture of [2]. The overall neural network architecture is depicted in Fig. 1.

E. Combinations of Text and Semantic Frame Readers

Herein, we propose transformer-based text and semantic frame readers. We can easily plug any type of transformer-based readers, such as ALBERT [8] and RoBERTa [9] into our framework.

Moreover, we employed the LSTM-based text and semantic frame reader of Jung et al. [1, 2] and combined transformer-based readers. For example, a BERT-based text reader and an LSTM-based semantic frame reader can be plugged into learn about semantic vectors. Herein, we denote the semantic vector learning model name as "[text reader] × [semantic frame reader]". For example, model LSTM × BERT denotes a semantic vector learning model using LSTM and BERT for text and semantic-frame readers, respectively. A summary of the tested readers is shown in Table 1.

F. Objective Functions

The same objective functions of [2] were used in this research. Our objective is to learn semantically reasonable vector representations of text and a related semantic frame with the proposed properties. The loss functions are defined to satisfy these properties.

The loss for "embedding correspondence" Property Distance loss measures the dissimilarity between the encoded semantic vectors from the text reader and those

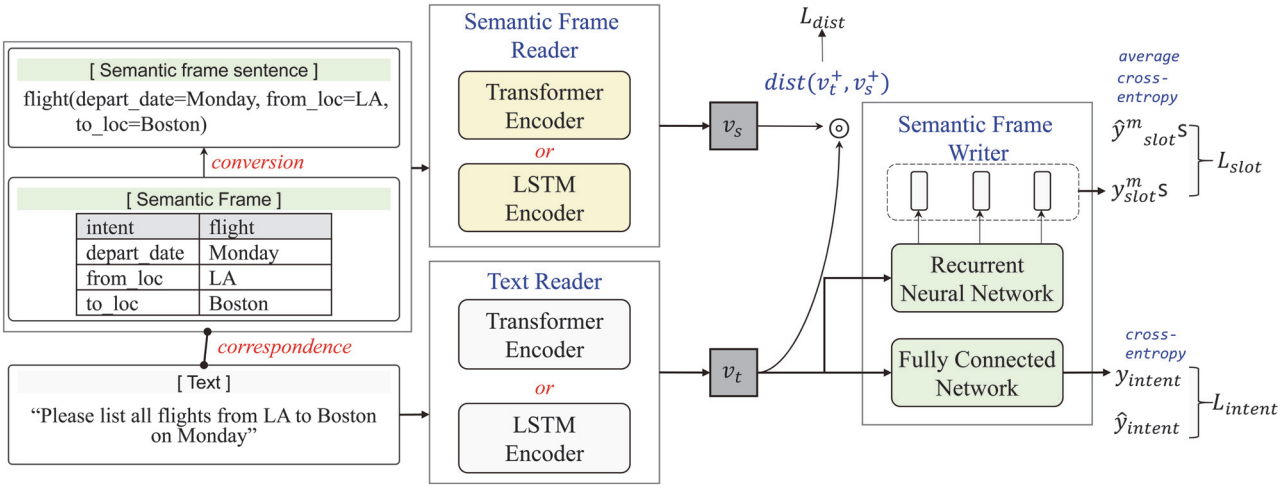

Fig. 1. Overall neural network architecture. $\hat{y}_{intent}$ is a reference intent tag vector and $\hat{y}^m_{slot}$ is a reference slot tag vector at time m. v_t represents the text vector, and v_s represents the semantic frame vector to the corresponding text.

Table 1. Descriptions of models used for text and semantic frame reader

Name	Language	Pretrained	Parameter size
LSTM	English/Korean	-	460K
BERT	English	BERT-base-uncased	109M
ALBERT	English	albert-base-v2	11M
RoBERTa	English	RoBERTa-base	124M
BERT	Korean	BERT-base-multilingual-cased	177M
KoBERT[1]	Korean	Pretrained BERT on large Korean data	92M

Table 2. Statistics of datasets

	ATIS	SNIPS	Sim-M	Sim-R	Weather
Number of trains	4,475	11,973	3,321	10,653	6,993
Number of dev.	499	685	1,032	2,442	3,009
Number of test	882	685	2,286	5,707	2,998
Avg. number of words	11.27	9.07	8.37	8.17	4.80
Per sentence number of intents	19	8	22	27	15
Number of slot tags	80	40	6	10	25

from the semantic frame reader in the vector space. The loss is defined as

$$L_{dist} = dist(v_t, v_s),$$

where the *dist* function is the cosine distance (=1.0 – cosine similarity).

The loss for "reconstruction" Property Content loss provides a measure of the amount of semantic information in a semantic frame vector. The reconstruction losses can be defined using the cross-entropy between the generated and reference tag vectors as

$$L_{intent} = CrossEntropy(\hat{y}_{intent}, y_{intent})$$

$$L_{slot} = \frac{1}{M}\sum_{m=1}^{M} CrossEntropy(\hat{y}^m_{slot}, y^m_{slot}),$$

where $\hat{y}_{intent}$ and y_{intent} are the reference and predicted intent tags, respectively, and M is the number of slots in a sentence.

$$L_{content} = L_{intent} + L_{slot}.$$

With the combination of intent and slot losses, the content loss ($Lcontent$) to reconstruct a semantic frame from a semantic vector v can be defined as follows:

Finally, the total loss value L for learning the semantic frame representation is defined by integrating all losses by summation.

$$L = L_{dist} + L_{content}.$$

III. EXPERIMENTS

We conducted experiments to determine whether the proposed pretrained transformer-based encoders contribute towards learning meaningful semantic representations and to determine the best text and semantic frame reader configurations.

For training and testing, we used four English datasets—ATIS2 [10], SNIPS [11], Simulated dialogue (Sim-R and Sim-M; https://github.com/google-research-datasets/simulated-dialogue), and one Korean dataset, Weather. The internally collected Weather dataset is relatively simple and comprises short sentences in terms of weather information. The information in the dataset is summarized in Table 2.

A. Sentence Search with Similarity

In our framework, instances having different forms (text or semantic frame) can be compared directly on a semantic vector space. Therefore, we can easily search similar sentences containing similarities between either texts or semantic frames. The sentence search performance improves with the learning of semantic representations by the model. To prove the effects of the proposed methods, we conducted sentence search experiments with the *precision at* K measure over different datasets.

Table 3 shows semantically similar sentence pattern search with both text and semantic frame as queries. From the experiments, it is clear that the best-performed models are somewhat different according to the query types. In the case of text-as-query, heavy pretrained transformer-based text readers (BERT, RoBERTa, and KoBERT) show good performance. Mostly, LSTM-based text readers showed lower performance. Meanwhile, in the case of semantic-frame-as-query, relatively light models (LSTM and ALBERT) show competitive and sometimes better performance. Note that pretrained transformer-based models showed much better performance especially in slot-tag pattern matching than LSTM based models.

B. Similarity-based Intent Classification

Pretrained transformer-based semantic representation learning is also helpful for the NLU task. To verify it, similarity-based intent classification experiments were conducted. Similarity-based classifiers estimate the class label of a test sample based on the similarities between the test sample and a set of labeled training samples, and the pairwise similarities between the training samples [2, 12]. When a sentence is provided, it is projected to v_t by the text reader. Subsequently, the intent tag i^j is sorted in an ascending order of distance score dist($\hat{v}_t^j v_t$) for all j

Table 3. Same sentence pattern (intent and slot tags should be matched) search performance (Precision@K) over various models and datasets

Model	Precision@1						Precision@5					
	Text as query			SF as a query			Text as query			SF as a query		
	I	S	J	I	S	J	I	S	J	I	S	J
ATIS												
LSTM × LSTM	0.959	0.653	0.645	0.995	**0.802**	**0.800**	0.976	0.744	0.728	0.998	**0.802**	**0.800**
LSTM × BERT	0.950	0.662	0.650	0.990	0.709**	0.709**	0.975	0.749	0.732	0.994	0.779**	0.778**
BERT × LSTM	**0.986****	0.766**	0.763**	**0.999**	0.770**	0.770**	0.988*	0.786**	0.781**	**0.999**	0.799	0.799
BERT × BERT	0.980**	**0.769****	**0.766****	0.998	0.778**	0.778**	**0.989***	**0.790****	**0.787****	0.998	0.791**	0.791*
LSTM × ALBERT	0.955	0.651	0.642	0.984**	0.694**	0.693**	0.980	0.748	0.738	0.992	0.780**	0.778**
ALBERT × LSTM	0.978**	0.678*	0.678**	**0.999**	0.753**	0.753**	0.984	0.741	0.738	**0.999**	0.798	0.798
ALBERT × ALBERT	0.976*	0.696**	0.693**	**0.999**	0.748**	0.748**	0.984	0.759	0.754**	**0.999**	0.786**	0.786**
LSTM × RoBERTa	0.960	0.670	0.661	0.983**	0.690**	0.686**	0.977	0.748	0.737	0.988**	0.786*	0.778**
RoBERTa × LSTM	0.981**	0.761**	0.756**	**0.999**	0.799	0.799	0.988*	0.788**	0.782**	**0.999**	0.800	0.800
RoBERTa × RoBERTa	0.975*	0.768**	0.763**	**0.999**	0.761**	0.761**	0.985	**0.790****	0.783**	**0.999**	0.788**	0.788**
SNIPS												
LSTM × LSTM	**0.994**	0.474	0.474	**1.000**	0.848	0.848	**0.997**	0.692	0.692	**1.000**	0.851	0.851
LSTM × BERT	0.985*	0.501	0.501	**1.000**	0.634**	0.634**	0.994	0.674	0.674	**1.000**	0.788**	0.788**
BERT × LSTM	0.991	**0.816****	**0.816****	**1.000**	0.829**	0.829**	0.994	**0.835****	**0.835****	**1.000**	0.842*	0.842*
BERT × BERT	0.991	0.800**	0.800**	**1.000**	**0.853**	**0.853**	0.993	0.820**	0.820**	**1.000**	**0.853**	**0.853**
LSTM × ALBERT	0.987	0.469	0.469	**1.000**	0.590**	0.590**	0.994	0.648**	0.648**	**1.000**	0.769**	0.769**
ALBERT × LSTM	0.981*	0.740**	0.740**	**1.000**	0.847	0.847	0.991	0.804**	0.804**	**1.000**	0.850	0.850
ALBERT × ALBERT	0.987	0.701:	0.701**	**1.000**	**0.853**	**0.853**	0.991	0.791**	0.791**	**1.000**	**0.853**	**0.853**
LSTM × RoBERTa	0.988	0.467	0.467	0.999	0.591**	0.591**	0.994	0.672	0.672	**1.000**	0.758**	0.758**
RoBERTa × LSTM	0.985	0.793**	0.793**	**1.000**	0.137**	0.137**	0.994	0.822**	0.822**	**1.000**	0.188**	0.188**
RoBERTa × RoBERTa	0.988	0.791**	0.791**	**1.000**	**0.853**	**0.853**	0.994	0.828**	0.828**	**1.000**	**0.853**	**0.853**
Sim-M												
LSTM × LSTM	0.687	0.679	0.580	0.994	**0.987**	**0.985**	0.859	0.805	0.712	0.995	**0.993**	**0.985**
LSTM × BERT	0.825**	0.771**	0.692**	0.959**	0.911**	0.897**	0.936**	0.898**	0.836**	0.983**	0.960**	0.943**
BERT × LSTM	0.958**	0.927**	0.899**	**1.000****	0.361**	0.361**	**0.980****	**0.964****	**0.946****	**1.000****	0.501**	0.501**
BERT × BERT	0.955**	0.935**	0.899**	**1.000****	0.980**	0.980**	**0.980****	**0.964****	0.943**	**1.000****	0.984**	0.984
LSTM × ALBERT	0.759**	0.703**	0.615**	0.958**	0.868**	0.857**	0.898**	0.835**	0.753**	0.983**	0.933**	0.923**
ALBERT × LSTM	0.946**	0.895**	0.866**	**1.000****	0.366**	0.366**	0.975**	0.944**	0.921**	**1.000****	0.432**	0.432**
ALBERT × ALBERT	0.931**	0.886**	0.839**	**1.000****	0.976**	0.976**	0.973**	0.949**	0.925**	**1.000****	0.981**	0.981*
LSTM × RoBERTa	0.801**	0.748**	0.654**	0.957**	0.876**	0.864**	0.910**	0.863**	0.789**	0.984**	0.944**	0.932**
RoBERTa × LSTM	0.955**	0.935**	0.901**	**1.000****	0.395**	0.395**	0.979**	0.963**	0.944**	**1.000****	0.490**	0.490:
RoBERTa × RoBERTa	**0.964****	**0.940****	**0.914****	**1.000****	0.979**	0.979**	**0.980****	0.962**	0.945**	**1.000****	0.983**	0.983
Sim-R												
LSTM × LSTM	0.914	0.851	0.798	0.999	**0.993**	**0.993**	0.966	0.931	0.905	0.999	**0.993**	**0.993**
LSTM × BERT	0.896**	0.850	0.792	0.992**	0.967**	0.962**	0.959**	0.938*	0.907	0.996**	0.984**	0.980**
BERT × LSTM	**0.950****	0.943**	0.905**	**1.000**	0.354**	0.354**	0.976**	0.978**	0.958**	**1.000**	0.446**	0.446**
BERT × BERT	**0.950****	**0.951****	**0.911****	**1.000**	**0.993**	**0.993**	0.976**	**0.981****	**0.961****	**1.000**	**0.993**	**0.993**
LSTM × ALBERT	0.900**	0.837**	0.779**	0.994**	0.967**	0.963**	0.960*	0.931	0.897**	0.997**	0.983**	0.980**
ALBERT × LSTM	0.941**	0.927**	0.884**	**1.000**	0.957**	0.957**	0.973**	0.973**	0.950**	**1.000**	0.961**	0.961**
ALBERT × ALBERT	0.944**	0.935**	0.894**	**1.000**	**0.993**	0.992	0.976**	0.972**	0.952**	**1.000**	**0.993**	0.992
LSTM × RoBERTa	0.909	0.861*	0.806	0.994**	0.959**	0.955**	0.961*	0.942**	0.910	0.997**	0.981**	0.978**
RoBERTa × LSTM	0.946**	0.948**	0.907**	**1.000**	0.308**	0.308**	0.977**	0.978**	0.958**	**1.000**	0.337**	0.336**
RoBERTa × RoBERTa	**0.950****	0.944**	0.902**	0.999	0.992	0.992	**0.978****	0.979**	0.959**	0.999	**0.993**	0.992
Weather												
LSTM × LSTM	0.995	0.710	0.708	**1.000**	**0.981**	**0.981**	0.997	0.917	0.914	**1.000**	**0.981**	**0.981**
LSTM × BERT	0.995	0.724	0.721	**1.000**	0.883**	0.883**	0.997	0.920	0.916	**1.000**	0.948**	0.948**
BERT × LSTM	**0.996**	0.970**	0.967**	**1.000**	0.055**	0.055**	**0.998**	0.975**	**0.973****	**1.000**	0.112**	0.112**
BERT × BERT	**0.996**	0.969**	0.966**	**1.000**	**0.981**	**0.981**	0.997	0.975**	0.972**	**1.000**	**0.981**	**0.981**
LSTM × KOBERT	0.994	0.685*	0.683**	**1.000**	0.861**	0.861**	0.995	0.911	0.906	**1.000**	0.942**	0.942**
KOBERT × LSTM	**0.996**	**0.971****	**0.969****	**1.000**	0.292**	0.292**	0.997	**0.976****	**0.973****	**1.000**	0.514**	0.514**
KOBERT × KOBERT	**0.996**	0.969**	0.967**	**1.000**	0.967**	0.967**	0.997	0.975**	**0.973****	**1.000**	0.980	0.979*

A model identifier indicates [text reader] × [semantic frame reader]. For example, BERT × LSTM represents a model with a BERT-based text reader and LSTM-based semantic frame reader. I, S, and J represent the intent, slot-tag, and joint of intent and slot-tag matching, respectively. SF denotes the semantic frame.

*p<0.05, significantly different significantly different from the baseline - LSTM × LSTM [2].

**p<0.01, significantly different significantly different from the base line.

Table 4. Similarity-based intent classification results (accuracy): English data

k	L×L	L×B	B×L	B×B	L×A	A×L	A×A	L×R	R×L	R×R
					ATIS					
1	0.927	0.937	0.981**	**0.969****	0.926	**0.967****	**0.971****	0.938	**0.969****	**0.973****
3	0.931	0.937	0.980**	**0.972****	0.925	**0.966****	**0.971****	**0.943***	**0.969****	**0.973****
5	0.929	0.937	0.980**	**0.972****	0.921	**0.966****	**0.971****	**0.946****	**0.969****	**0.972****
10	0.924	0.935	0.978**	**0.972****	0.920	**0.966****	**0.968****	**0.939***	**0.969****	**0.972****
40	0.905	0.918	0.955**	**0.933****	0.900	**0.922***	0.916	0.915	0.920	**0.939****
					SNIPS					
1	0.981	0.981	0.987	0.990**	0.981	0.985	0.988	0.982	0.987	0.990**
3	0.981	0.982	0.987	0.990**	0.980	0.987	0.988	0.981	0.987	0.990**
5	0.981	0.981	0.987	0.990**	0.980	0.987	0.988	0.981	0.987	0.990**
10	0.981	0.981	0.987	0.990**	0.980	0.987	0.988	0.984	0.987	0.990**
40	0.981	0.981	0.987	0.990**	0.982	0.987	0.988	0.984	0.987	0.990**
					Sim-M					
1	0.673	**0.807****	0.961**	0.961**	**0.762****	**0.939****	**0.936****	**0.792****	**0.964****	**0.959****
3	0.710	**0.828****	0.966**	**0.965****	**0.784****	**0.945****	**0.941****	**0.814****	**0.965****	**0.965****
5	0.717	**0.832****	**0.964****	0.967**	**0.784****	**0.946****	**0.940****	**0.828****	**0.963****	**0.964****
10	0.716	**0.839****	0.966**	0.966**	**0.786****	**0.944****	**0.943****	**0.828****	**0.964****	**0.964****
40	0.705	**0.834****	**0.955****	**0.959****	**0.765****	**0.937****	**0.939****	**0.799****	0.960**	**0.955****
					Sim-R					
1	0.890	**0.877****	**0.935****	**0.932****	**0.870****	**0.925****	**0.917****	0.883	**0.931****	0.936**
3	0.900	**0.883****	**0.933****	**0.937****	**0.879****	**0.924****	**0.921****	0.894	**0.933****	0.938**
5	0.905	**0.886****	**0.934****	**0.937****	**0.882****	**0.924****	**0.923****	**0.894****	**0.933****	0.939**
10	0.906	**0.882****	**0.936****	**0.937****	**0.879****	**0.927****	**0.921****	**0.898****	**0.934****	0.938**
40	0.908	**0.872****	**0.932****	**0.937****	**0.877****	**0.927****	**0.923****	**0.896****	**0.936****	0.939**

From the top-K list of the intent tag, the most frequently appearing intent tag is selected as an intent tag for the provided sentence. L, B, A, R, and K represent LSTM, BERT, ALBERT, RoBERTa, and KoBERT, respectively.
*p<0.05, significantly different from the baseline - L×L [2].
**p<0.01, significantly different from the baseline.

where $\hat{v}_i^j$ is the jth text vector in the training set. From the top-K list of the intent tag, the most frequently appearing intent tag is selected as the intent tag for the provided sentence.

Tables 4 and 5 show the similarity-based intent classification results. The proposed models were found to be effective in capturing intent information into vector representations. BERT-based text readers (B×L, B×B) showed the best performance over all English datasets except Sim-R. Note that in most of the cases, transformer-based semantic frame readers showed much better intent classification performance (ex: L×L vs. L×B). This implies that the pretrained transformer contributes to better semantic vector learning even though they are pretrained on raw texts and not on structured semantic frames.

Table 5. Similarity-based intent classification results (accuracy): Korean data

k	L×L	L×B	B×L	B×B	L×K	K×L	K×K
1	0.996	0.996	0.998	0.997	0.995	0.997	0.997
3	0.996	0.995	0.997	0.997	0.995	0.997	0.997
5	0.996	0.995	0.997	0.997	0.995	0.997	0.997
10	0.995	0.996	0.997	0.997	0.995	0.997	0.997
40	0.995	0.995	0.997	0.997	0.994	0.997	0.996

From the top-K list of the intent tag, the most frequently appearing intent tag is selected as an intent tag for the provided sentence. L, B, A, R, and K represent LSTM, BERT, ALBERT, RoBERTa, and KoBERT, respectively.

In the case of the Korean dataset, all models showed a good performance of over 0.99. This is because the

Weather dataset consists of much shorter sentences and a small number of intent-tags; all models including LSTM successfully capture the intent-level semantic information.

Note that the reported LSTM × LSTM performance of the ATIS domain in Tables 3–5 is different from that of [1, 2]. While they used word-level tokenization, in this work, we used sub-word-level tokenization for a fair comparison with other transformer models that use subword-level tokenization.

C. Learning Curves

The effect of pretrained transformers can be confirmed by examining the training loss curves from Fig. 2. The losses from the models that use transformer-based text readers decrease and reach the lowest value much faster than the models using LSTM-based text readers. Note that the lowest loss values of each model are not similar. The lowest loss values of LSTM-based text readers are

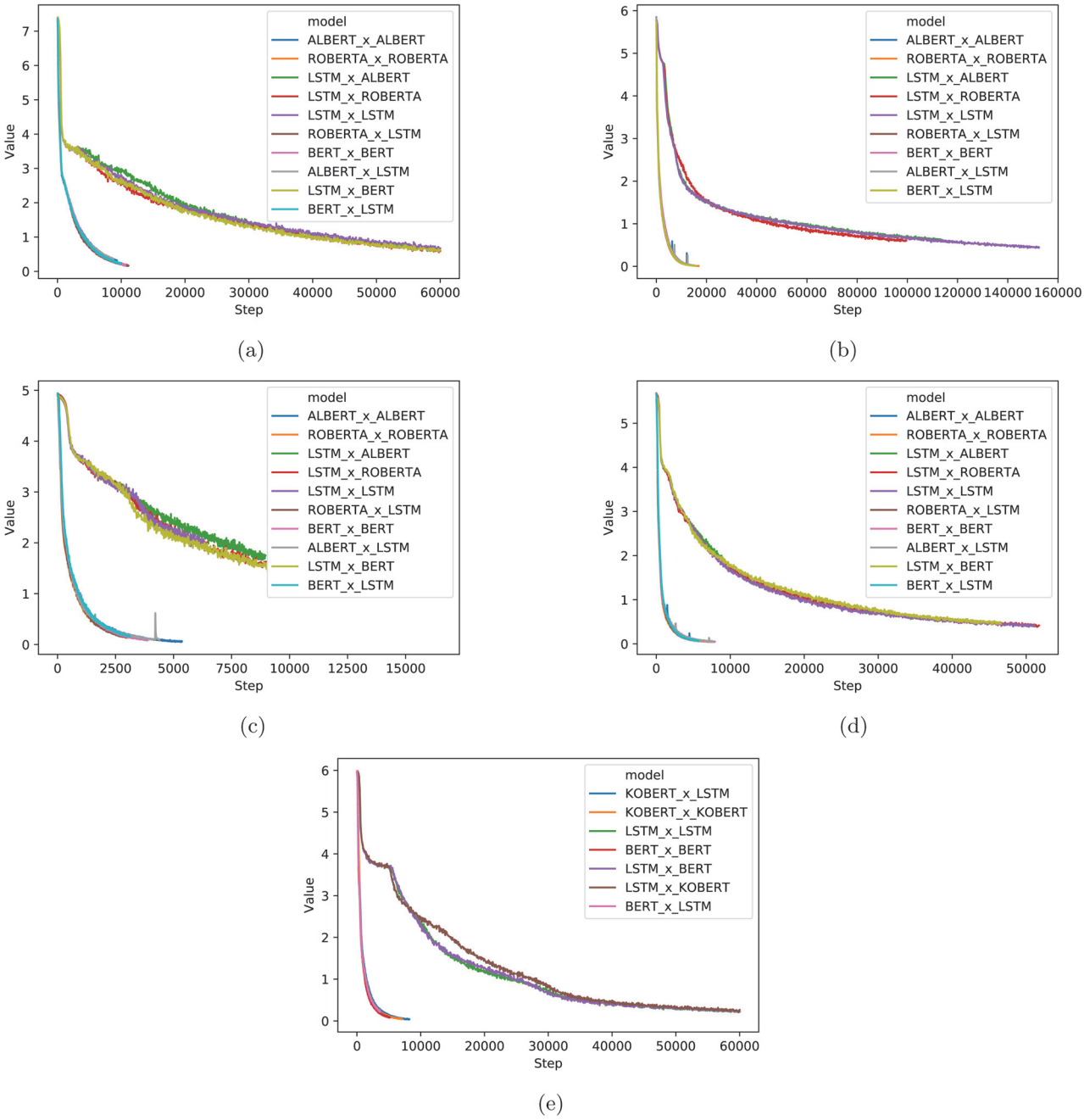

Fig. 2. Training losses over steps in each dataset: (a) ATIS, (b) SNIPS, (c) Sim-M, (d) Sim-R, and (e) Weather.

always larger than the values of the models using the transformer text reader. For example, in the case of ATIS, the lowest loss value of LSTM × BERT (0.2133) is larger than the value of BERT × BERT (0.08129).

IV. RELATED WORK

Recent studies have focused on enriching the representations for neural architectures to implement NLU. For example, Chen et al. [13] focused on leveraging substructure embedding for joint semantic frame parsing. Kim et al. [14] utilized several semantic lexicons, such as WordNet, PPDB, and the Macmillan dictionary, to enrich word embedding, and they subsequently used them in the initial representation of words for intent detection.

Furthermore, many structured text-to-vector techniques have been introduced recently. A. Preller [15] introduced a logic formula embedding method, while A. Bordes et al. [16] and K. Do et al. [17] proposed translating symbolic-structured knowledge, such as WordNet and freebase.

Recently, Jung et al. [1, 2] introduced a semantic frame embedding method by simultaneously executing the raw and structured text-to-vector methods in a single framework to learn more semantic representations. In their framework, the text and semantic frames are each projected onto a vector space, and the distance loss between the vectors is minimized to satisfy the embedding correspondence

Recently, contextual sentence modeling has been introduced in contextual language models, including ELMo [18], GPT [19], BERT [6], and XLNet [20]. In particular, BERT uses a different pretraining objective, masked language model, and the next sentence prediction task that jointly pretrains text-pair representations.

V. CONCLUSION

Herein, we have proposed an extended framework for learning semantic vector representation following [1–4]. The framework enables combining both LSTM and transformer-based readers to learn representations. Through experiments with the ATIS, SNIPS, Sim-M, Sim-R, and Weather datasets, we found that the proposed methods could successfully learn semantic vector representations. In most cases, the proposed models that use the transformer-based text and semantic frame reader show better performance in semantic search and intent classification as well as faster training speed. However, in the case of semantic frame readers, LSTM-based models showed competitive results over transformer-based semantic frame readers. It appears that the semantic frame format is already simplified and well-structured; thus, lightweight LSTM is sufficient to learn the semantic representation.

Based on the results of the proposed research, various research directions can be considered in the future. Semantic operation or algebra on a vector space is a promising research topic. Furthermore, we will study the shared semantic vector space between multiple datasets or the general domain datasets.

ACKNOWLEDGEMENT

This work was supported by a research fund from Chungnam National University.

REFERENCES

1. S. Jung, J. Lee, and J. Kim, “Learning to embed semantic correspondence for natural language understanding,” in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, Brussels, Belgium, 2018, pp. 131-140.
2. S. Jung, “Semantic vector learning for natural language understanding,” *Computer Speech & Language*, vol. 56, pp. 130-145, 2019.
3. S. Jung, H. Suh, H. J. Kim, and T. W. Hwang, “Semantic similarity-based intent analysis using pretrained transformer for natural language understanding,” *Journal of KIISE*, vol. 47, no. 8, pp. 748-760, 2020.
4. H. Kim and S. Jung, “Semantic vector learning using pretrained BERT,” in *Proceeding of the Korea Software Congress*, Pyeongchang, Korea, 2019, pp. 1427-1429.
5. T. W. Hwang, S. Jung, H. Suh, and H. J. Kim, “Low dimensional semantic frame transformation method and regression-based semantic vector learning,” in *Proceeding of the Korea Computer Congress*, Busan, Korea, 2020, pp. 341-343.
6. J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, MN, 2019, pp. 4171-4186.
7. M. Henderson, B. Thomson, and J. Williams, “Dialog state tracking challenge 2&3,” 2013; https://github.com/matthen/dstc.
8. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite BERT for self-supervised learning of language representations,” in *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA, 2019.
9. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: a robustly optimized Bert pretraining approach,” 2019; https://arxiv.org/abs/1907.11692.
10. P. J. Price, “Evaluation of spoken language systems: the ATIS domain," in *Proceedings of the workshop on Speech and Natural Language*, Hidden Valley, PA, 1990.
11. A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, et al., “Snips voice platform: an embedded spoken

language understanding system for private-by-design voice interfaces," 2018; https://arxiv.org/abs/1805.10190.
12. Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based classification: concepts and algorithms," *Journal of Machine Learning Research*, vol. 10, no. 3, pp. 747-776, 2009.
13. Y. N. Chen, D. Hakanni-Tur, G. Tur, A. Celikyilmaz, J. Guo, and L. Deng, "Syntax or semantics? Knowledge-guided joint semantic frame parsing," in *Proceedings of 2016 IEEE Spoken Language Technology Workshop (SLT)*, San Diego, CA, 2016, pp. 348-355.
14. J. K. Kim, G. Tur, A. Celikyilmaz, B. Cao, and Y. Y. Wang, "Intent detection using semantically enriched word embeddings," in *Proceedings of 2016 IEEE Spoken Language Technology Workshop (SLT)*, San Diego, CA, 2016, pp. 414-419.
15. A. Preller, "From logical to distributional models," 2014; https://arxiv.org/abs/1412.8527.
16. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in Neural Information Processing Systems*, vol. 26, pp. 2787-2795, 2003.
17. K. Do, T. Tran, and S. Venkatesh, "Knowledge graph embedding with multiple relation projections," *Proceedings of 2018 24th International Conference on Pattern Recognition (ICPR)*, Beijing, China, 2018, pp. 327-337.
18. M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, LA, 2018.
19. A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018; http://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.
20. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: generalized autoregressive pretraining for language understanding," 2019; https://arxiv.org/abs/1906.08237.

Sangkeun Jung

Sangkeun Jung received his B.S., M.S., and Ph.D. degrees in computer engineering from POSTECH, Pohang, Korea, from 2004 to 2010. From 2010 to 2012, he was a researcher with the Samsung Electronics, Suwon, Korea. From 2012 to 2014, he was a researcher with the ETRI, Daejeon, Korea. From 2014 to 2018, he was a researcher at SK Telecom, Seoul, Korea. Since 2018, he has been a professor of computer science and engineering at Chungnam National University, Daejeon, Korea. His research interests include natural language processing, machine learning, and deep learning.