JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 16, No. 2, June 2022, pp. 63-78

Semantic Vector Learning and Visualization with Semantic Cluster Using Transformers in Natural Language Understanding

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

Abstract

Natural language understanding (NLU) is a fundamental technology for implementing natural interfaces. The embedding of sentences and correspondence between text and its extracted semantic knowledge, called semantic frame, has recently shown that a semantic vector representation is key in the implementation or support of robust NLU systems. Herein, we propose an extension of cluster-aware modeling with various types of pre-trained transformers for consideration of the many-to-1 relationships of text-to-semantic frames and semantic clusters. To attain this, we define the semantic cluster, and design the relationships between cluster members to learn semantically meaningful vector representations. In addition, we introduce novel ensemble methods to improve the semantic vector applications around NLU, i.e., similarity-based intent classification and a semantic search. Furthermore, novel semantic vector and corpus visualization techniques are presented. Using the proposed framework, we demonstrate that the proposed model can learn meaningful semantic vector representations in ATIS, SNIPS, SimM, and Weather datasets

Category: Natural Language Processing

Keywords: Semantic vector; Semantic vector learning; Natural language understanding; Transformer; Cluster-aware; Visualization

I. INTRODUCTION

Natural language understanding (NLU) is a primary technique for implementing natural user interfaces, such as chatbots, mobile secretaries, and smart speakers. NLU aims to extract meaning from natural language and infer user intention.

In addition to developing NLU systems, many assistant tools are required for building and maintaining such systems. For example, semantic search utility facilitates the acceleration of data processing during the collection and annotation stages. A semantic distance calculation can be a core technique to visualize datasets for data augmentation. The re-ranking of multiple NLU systems is crucial for maintaining a stable performance over various types of input.

Jung et al. [1] and Jung [2, 3] demonstrated the importance of semantic vector representation for implementing key utilities around NLU systems. They introduced a neural architecture for embedding the correspondence between a text and a semantic frame. In embedding the semantic correspondence learning framework, the correspondence (the distance between projected vectors from text and semantic frames) was used as an objective score, whereas reconstruction (intent and slot-tag) was used as a generation cost.

In a study by Jung [2], long short-term memory (LSTM)-based readers were used to encode a sentence and

Open Access http://dx.doi.org/10.5626/JCSE.2022.16.2.63 http://jcse.kiise.org

Received 11 April 2022; **Accepted** 25 May 2022
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

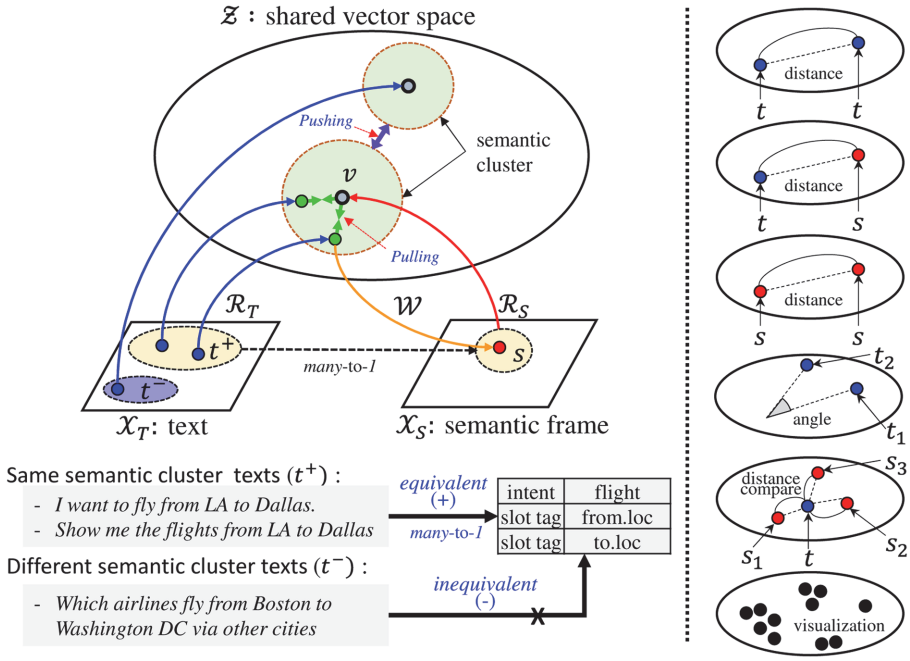

Fig. 1. Semantic cluster-aware semantic vector learning framework and its applications. Here, $\mathcal{R}_T$ and $\mathcal{R}_S$ are reader functions that encode raw and structured texts, respectively, to a semantic vector. In addition, $\mathcal{W}$ is a writing function that decodes a semantic vector to a symbolic semantic frame. Texts and a semantic frame in the same semantic cluster have a *many*-to-1 relationship. Text vectors from the same semantic clusters should be placed near the centroid vector, whereas semantic clusters should be placed apart from each other.

semantic frame. However, there is significant room for performance improvement by using recently pre-trained encoders. To accelerate representation learning, Jung [3] and Jung and Lim [4] employed pre-trained transformers, including bidirectional encoder representations from transformers (BERT) [5], to encode both sentences and semantic frames. In particular, to directly feed the structural form of the semantic frame to the transformer, they introduced an input processing method using a semantic frame and reduced the semantic frame sentences.

In NLU literature, there are clear semantic groups in the design and handling of datasets. As illustrated in Fig. 1, the sentences and corresponding semantic frames in an NLU task exhibit an *n*-to-1 relationship. In other words, many semantically equivalent texts can be members of a single semantic frame. Further analysis of the semantic frame structure reveals that the slot-values to slot-tags have an *n*-to-1 relationship. For example, single-slot tags, such as "from.city" can have many different slot values, such as "LA," "Dallas," and "Denver." To actively model these complex relationships of both text-to-semantic frames and semantic frames-to-semantic frames, Jung and Lim [4] introduced cluster-aware correspondence learning methods.

In this study, we extended the work of Jung and Lim [4] in several directions. First, we carefully review the concept of a semantic-cluster, cohesion, and separation proposed by Jung and Lim [4], and refine the concepts more concretely. In addition, we implemented many different types of transformer-based text and semantic frame readers and experimentally proved the effectiveness of the readers and cluster-awareness. An additional English NLU dataset was also tested. Further, as another key contribution, we introduce novel ensemble methods to improve the semantic vector applications around NLU, i.e., similarity-based intent classification and a semantic search. Furthermore, novel semantic vector and corpus visualization techniques are presented.

The remainder of this paper is organized as follows: Section II describes the semantic frame, semantic cluster design scheme, and semantic frame input processing methods for transformers. Section III describes the detailed structure of the proposed cluster-aware semantic vector learning framework. Section IV introduces semantic vector applications in NLU using ensemble and visualization methods. Section V presents the experimental results. Section VI discusses previous related studies. Finally, Section VII provides some concluding remarks.

II. SEMANTIC FRAME SENTENCES AND SEMANTIC CLUSTERS

Semantic frames are one of the most widely used structured representations of semantics in NLU [6-8]. They consist of intent and slots. An intent represents the sentence's role in the target domain. Slots represent domain-specific named entities or keywords. Fig. 2

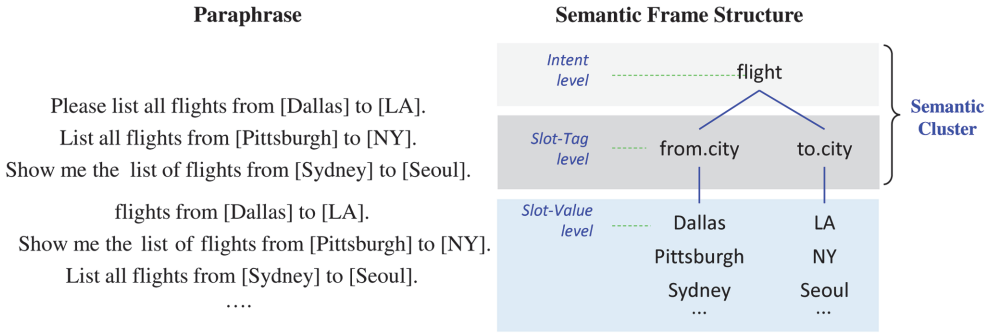

Fig. 2. Semantic frame structure and corresponding paraphrases. A semantic cluster is a cluster consisting of texts or semantic frames that share the same intent-tag and slot-tags.

illustrates the relationships between paraphrase texts and corresponding semantic frame information. We can view the semantic frame structures with several levels, i.e., intent, slot-tag, and slot-value. An intent-tag contains sentence-wise information, whereas a slot-tag and a slot-value contain entity-wise information. A single slot-tag can have many different slot values. For example, in Fig. 2, the slot-tag "from.city" has all city names including Dallas and Sydney.

To define a semantic cluster, several terminologies near the semantic frames must be stated first. From the structured knowledge, we can define a semantic frame sentence.

A Semantic frame sentence takes the form of a semantic frame with intent(tag=value, tag=value, ...) format. Its form is similar to that of a user action [9]. A semantic frame sentence in the example above is expressed in a sentence form—flight(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 Jung et al. [1].

A reduced semantic frame sentence is in a sentence form. In the above example, the reduced semantic frame sentence is flight(from.city=*, to.city=*). Note that the wildcard symbol * is used in the slot-value positions.

A cluster configuration can be understood through the relationships between a semantic frame and a reduced semantic frame. A single reduced semantic frame, e.g., intent(tag=*, tag=*, ...), has many corresponding semantically equivalent semantic frames and sentences. We can regard these semantically equivalent texts and semantic frames as a semantic cluster. In this study, we define this as follows:

A semantic cluster is a cluster consisting of texts or semantic frames that share the same intent-tag and slot-tags. The cluster centroid is a reduced semantic frame sentence (for example, flight(from.city=*, to.city=*)).

Fig. 3 depicts the semantic cluster concept. Comparing the concept with a galaxy and its solar systems is helpful in understanding a semantic cluster. There are many

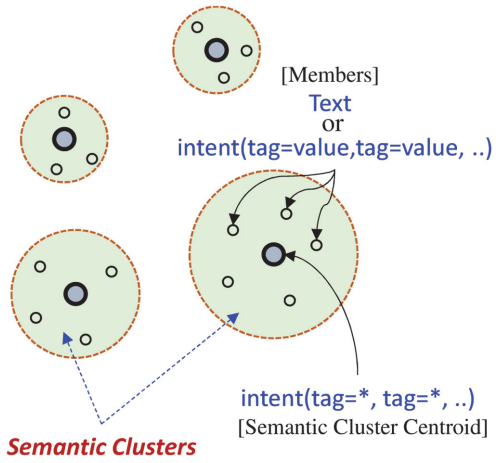

Fig. 3. Illustration of semantic cluster.

individual semantic centroids, i.e., a combination of intent and slot-tags, in a semantic space. These centroids correspond to the suns in a galaxy. A single semantic cluster can have many texts and semantic frames, much like solar systems have many planets. From the semantic clusters, we can introduce some properties to assist in semantic vector learning.

- Cohesion property: All instances in a semantic cluster must be enclosed together around the cluster centroid (see the grey inner circles in Fig. 3).
- Separation property: A semantic cluster should be separate from other clusters.

In the following section, we introduce detailed neural modeling methods that facilitate the learning of robust semantic vector representations.

III. CLUSTER-AWARE SEMANTIC VECTOR LEARNING

Our framework is composed of a text reader, semantic frame reader, and semantic frame writer, following Jung et al. [1] and Jung [2]. The text reader embeds a sequence

of tokens into a distributed vector representation. The semantic frame reader reads the structured texts and encodes each text into a vector. Finally, the semantic frame writer generates a symbolic reduced semantic frame from a vector representation.

In this study, we used transformer-based text and a semantic frame reader. Furthermore, we directly modeled the cohesion and separation between semantic clusters. We implemented BERT and RoBERTa and ALBERT transformer-based text and semantic frame readers. In this study, these transformers are denoted as a transformer encoder (TRE).

A. Text Reader

Transformer text reader: Input text is tokenized by a subword, and the tokens are fed to a transformer.

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 follows:

$$\overrightarrow{h}_t = TRE_{text}([CLS], t_1, t_2, ..., t_k, [SEP])$$
$$v_t = sigmoid(W_T \overrightarrow{h}_t),$$

where $\overrightarrow{h}_t$ is the sentence embedding vector from the transformer; t_k is a token at step k; [CLS] and [SEP] are class and separation tokens, respectively; W_T is the linear projection weight; and v_t is a semantic vector derived from the text. In this research, we set 100 dimensions as the final semantic vector dimension; however, there is no strict limitation in setting the semantic vector dimension. The dimension might be set to 2 or 3 when considering a decline in performance. The sigmoid activation function is employed for the nonlinear modeling and normalization because the sigmoid function is bounded in [0.0, 1.0].

B. Semantic Frame Reader

Jung et al. [1] and Jung [2] introduced an LSTM-based hierarchical semantic frame reader that can capture structural information using multiple LSTM and a multilayer perceptron. To accelerate and make a more robust semantic frame representation, Jung [3] and Jung and Lim [4] introduced a transformer-based semantic frame encoding method that employs the sentence form of the semantic frame components, such as a 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 a transformer. The sentence embedding vector from the [CLS] token is extracted from the final layer and linearly projected into the semantic vector dimension. The semantic frame sentence encoding process can be defined as follows:

$$\overrightarrow{h}_s = TRE_{sf}([CLS], s_1, s_2, ..., s_k, [SEP])$$
$$v_s = sigmoid(W_S \overrightarrow{h}_s),$$

where $\overrightarrow{h}_s$ is the embedding vector from a transformer, 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.

It is worth noting that the transformer-based semantic frame reader is used to project not only regular semantic frame sentences but also reduced-semantic-frame sentences (centroids) into a vector space.

C. Semantic Frame Writer

To prevent zero-vector convergence, Jung et al. [1] proposed a semantic frame writing to learn robust semantic representations. They introduced a simple linear-projection-based intent writer and recurrent neural network-based slot-tag writer to decode a reduced semantic frame. We used the same semantic frame writer architecture as Jung [2]. The overall neural network architecture is shown in Fig. 4.

D. 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 [10] and RoBERTa [11], into our framework. We denote the semantic vector learning model, called "[text reader] × [semantic frame reader]." For example, the BERT × BERT model denotes a semantic vector learning model using BERT and BERT for text and semantic frame readers, respectively. In this research, we use the same type of transformer for both text and semantic frame readers. A summary of the implemented readers is shown in Table 1.

E. Cluster Cohesion

As defined in Section II, the semantic cluster contains many corresponding semantically equivalent instances. These semantically equivalent texts or semantic frames should be packed closely around the cluster centroid. More specifically, the projected text vector v_t^+ s should be placed around the projected reduced semantic frame vector v_{rs}^+ (Fig. 5). We call this property semantic cluster cohesion.

To directly model the property, a single reduced-semantic-frame vector and randomly selected p corresponding paraphrased text vectors from a semantic cluster are fed to a neural network. Considering the distances between the p text vectors and the cluster centroid vector, the cohesion score can be calculated. More details on the modeling formula are provided in Section III-G.

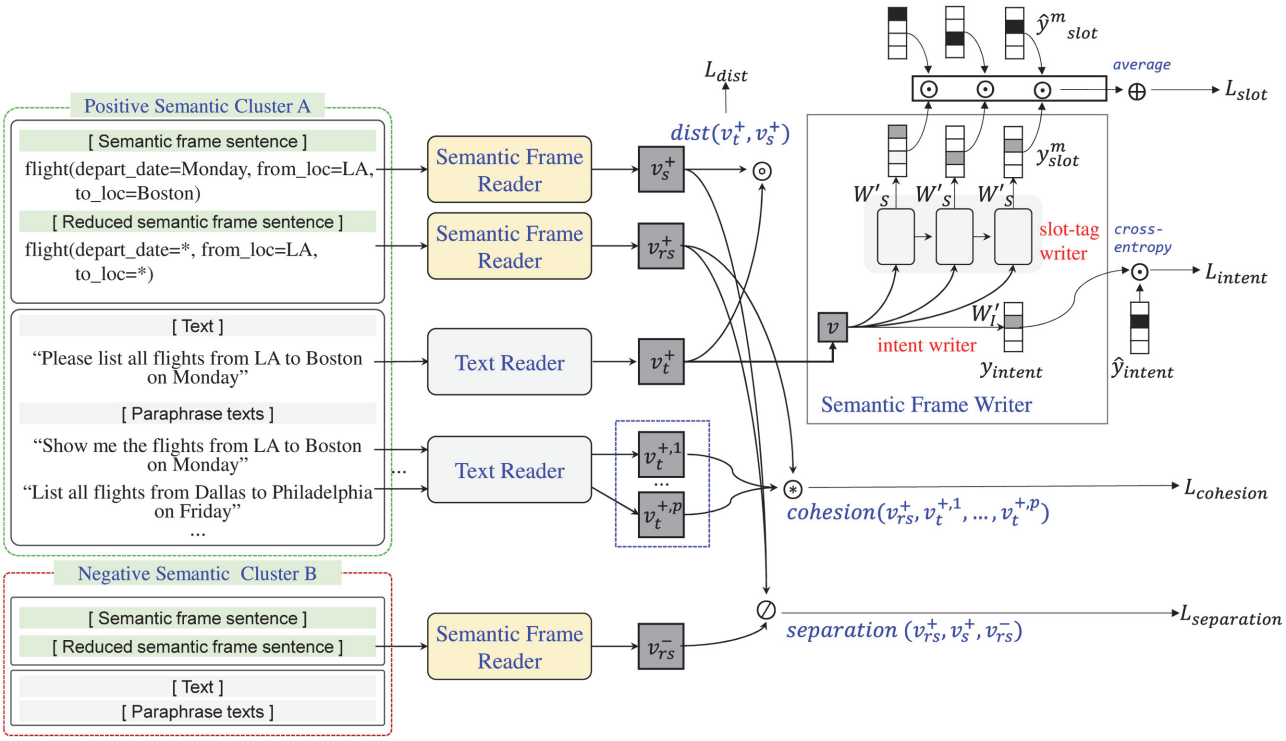

Fig. 4. Overall neural network architecture. Here, $\hat{y}_{intent}$ is a reference intent tag vector and $\hat{y}^m_{slot}$ is a reference slot tag vector at time *m*. In addition, $v_t^{+,p}$ represents the *p*th text vector from the same semantic cluster, v_{rs}^+ represents the semantic cluster centroid vector, and v_{rs}^- represents the centroid vector of another semantic cluster. Both v_{rs}^+ and v_{rs}^- share the same intent tag but do not share the slot-tag sequences.

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

Name	Language	Pretrained	Parameter size
LSTM	Engish/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

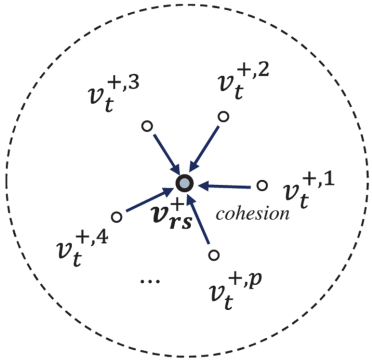

Fig. 5. Cohesion property modeling. Text vectors ($v_t^{+,i}$s) should be placed close to the semantic cluster centroid vector, which is the projected reducedsemantic-frame vector (v_{rs}^+). In addition, *p* is the number of positive texts when considering cohesion.

F. Cluster Separation

An embedding of the correspondence and cohesion is insufficient for learning robust semantic representations. Let us assume a case in which two semantic centroids are placed closely together, and the corresponding text vectors are learned by placing them around each semantic centroid (Fig. 6(a)). In this case, the embedding correspondence and cohesion properties are satisfied; however, overlapped areas exist between the two semantic clusters when the distance between the two centroids is short.

Therefore, semantic clusters must be placed apart from each other to learn distinct semantic representations. We call this property semantic cluster separation.

To model the semantic cluster separation, we consider (1) the semantic cluster centroid vector v_{rs}^+, (2) its

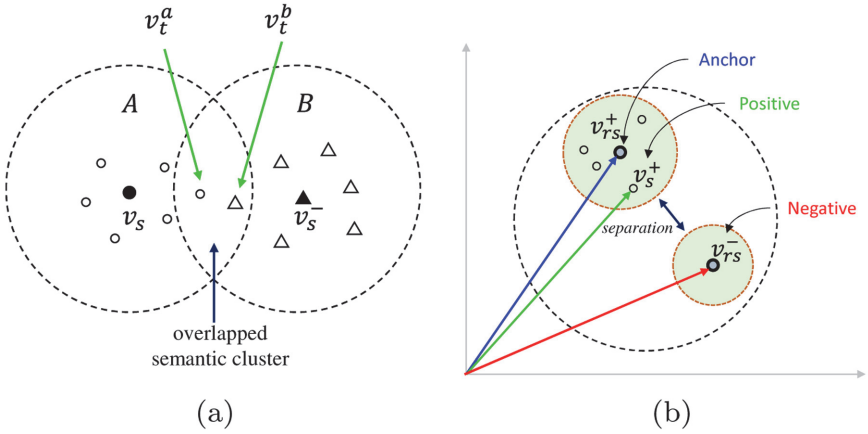

Fig. 6. Separation property modeling: (a) semantic cluster overlapping and (b) semantic cluster separation. In (a), v_t^a and v_t^b are closely placed, whereas v_t^a and v_t^b are members of semantic clusters A and B, respectively. Similar to (b), semantic clusters should be separated from each other by a certain margin.

randomly selected semantic frame vector v_s^+, and (3) the centroid vector v_{rs}^- of the negative cluster, which is a reduced-semantic-frame vector of a randomly chosen semantic cluster. Positive and negative clusters are chosen from the same intent groups. In other words, positive and negative clusters must share the same intent-tag. However, when there is only one intent-slot-tag combination in a cluster, a negative semantic cluster is selected from another randomly selected semantic cluster. To force two clusters apart, we employed a triplet margin loss [12]. This loss is used for measuring the relative similarity between samples; in addition, it can push different clusters apart from each other.

G. Objective Functions

The objective of this study is to learn semantically reasonable vector representations of text and a related semantic frame with the proposed properties. Hence, we set the properties of the desirable semantic vector, and the loss functions are defined to satisfy these properties.

We used the cohesion and separation losses as well as the embedding correspondence loss and content keeping loss of Jung [2] to learn the semantic representations.

Loss for embedding-correspondence-property: The distance loss measures the dissimilarity between the encoded semantic vectors from the text reader and those from the semantic frame reader in the vector space. The loss is defined as follows:

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

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

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

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

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

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

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

Loss for "cohesion" property: The semantic-cluster cohesion loss can be calculated as follows:

$$L_{cohesion} = \frac{1}{p}\sum_{k}^{p} dist(v_{rs}^+, v_t^{+,k}),$$

where p is the number of paraphrase texts, $v_t^{+,k}$ denotes the kth text vector from the same cluster, and v_{rs}^+ denotes the semantic cluster centroid. Paraphrase texts are randomly selected in every batch to feed the neural network. In this study, we set p to 5.

Loss for "separation" property: The separation loss is devised for a semantic cluster using the triplet loss in the cosine similarity [13]. Formally, the triplet loss captures the following property:

$$sim(a, p) - \alpha > sim(a, n),$$

where $sim(a, p)$ is the cosine similarity between the anchor a and positive example p, and $sim(a, n)$ is the cosine similarity between the anchor a and negative example n. In addition, α is a margin value. The triplet loss in the cosine similarity can be defined as follows:

$$L^{tri}(a, p, n) = [sim(a, n) - sim(a, p) + \alpha]_+,$$

where the operator $[x]_+ = \max(x, 0)$.

Using triplet loss, the semantic cluster separation loss can be defined as follows:

$$L_{separation} = L^{tri}(v_{rs}^+, v_s^+, v_{rs}^-),$$

where the margin value α is determined by examining the early loss curves empirically.

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

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

IV. APPLICATIONS

If we have a good vector form of a text or semantic frame representation, we can devise many useful applications for NLU tasks because the text or the semantic representations are computable and interpretable mathematically. Furthermore, we can devise more effective visualization methods employing learned semantic vectors as well as a symbolic semantic frame structure.

The following sub-sections illustrate two semantic vector applications, i.e., intent classification and semantic search, in NLU and introduce ensemble methods to improve the performance of each application. In addition, novel semantic vector visualization techniques will be introduced.

A. Similarity-based Intent Classification

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 [1, 2].

When a sentence is given, the sentence is projected onto v_t by the text reader. The i^j intent tags are then sorted in ascending order with the distance score $dist(\hat{v}_t^j, v_t)$ for all values of j, where $\hat{v}_t^j$ is the jth text vector in the training set. From the top-K list of intent tags, the most frequently appearing intent tag is selected as the intent tag for the given sentence.

B. Similarity-based Sentence Search

In our framework, instances having different forms (text or semantic frame) can be compared directly in a semantic vector space. Therefore, we can easily search for similar sentences with similarities between either text or semantic frames. The better the model learns the semantic representations, the greater the improvement in the sentence search performance.

C. Ensemble Methods using Multiple Semantic Vector Projections

In this study, we introduced various types of text and semantic frame readers. By employing ensemble methods in multiple text and semantic frame readers, we might achieve better results on semantic vector applications in NLU. For a similarity-based intent classification and semantic search, we introduce voting methods that are slightly modified for proper handling of the vectors and distances for the target applications.

1) Voting for Similarity-based Intent Classification

The proposed similarity-based intent classifiers output k-intent-classes according to the number k, which is for k-nearest instance finding in the training datasets. If we use N different text and semantic frame readers, we will have $k \times N$ output classes from multiple systems. To obtain the ensemble results, we conduct majority voting on the $k \times N$ outputs. Fig. 7(a) shows an illustration of the ensemble in similarity-based intent classification according to k.

2) Voting for Semantic Search

Similar sentence pattern matching is conducted by searching the closest instances of the test dataset with a given query test sentence based on the distances between them. Whereas a similarity-based intent classifier directly generates the output classes, the similar sentence pattern matcher retrieves the indexes of the closest test examples. Therefore, the voting should be conducted on these indexes. Fig. 7(b) illustrates the semantic search ensemble methods. Each text and semantic frame reader retrieves the M closest test indexes, and majority-voting is applied on $N \times M$ retrieved test indexes, and the most similar k instances are finally determined. Here, we set M to 3 based on the heuristics.

D. Visualization

As one of the main advantages of semantic vector learning, visualization can be applied directly using projected vectors from text and semantic frames. In Jung et al. [1], they showed the entire dataset visualization based on t-sne techniques. However, their method visualizes the vector space indirectly because t-sne manipulates high-dimensional vectors, and projects them again into a low-dimensional vector space. Although the projected low-dimensional vectors can indirectly show the effectiveness of the semantic vector learning methods, the projected

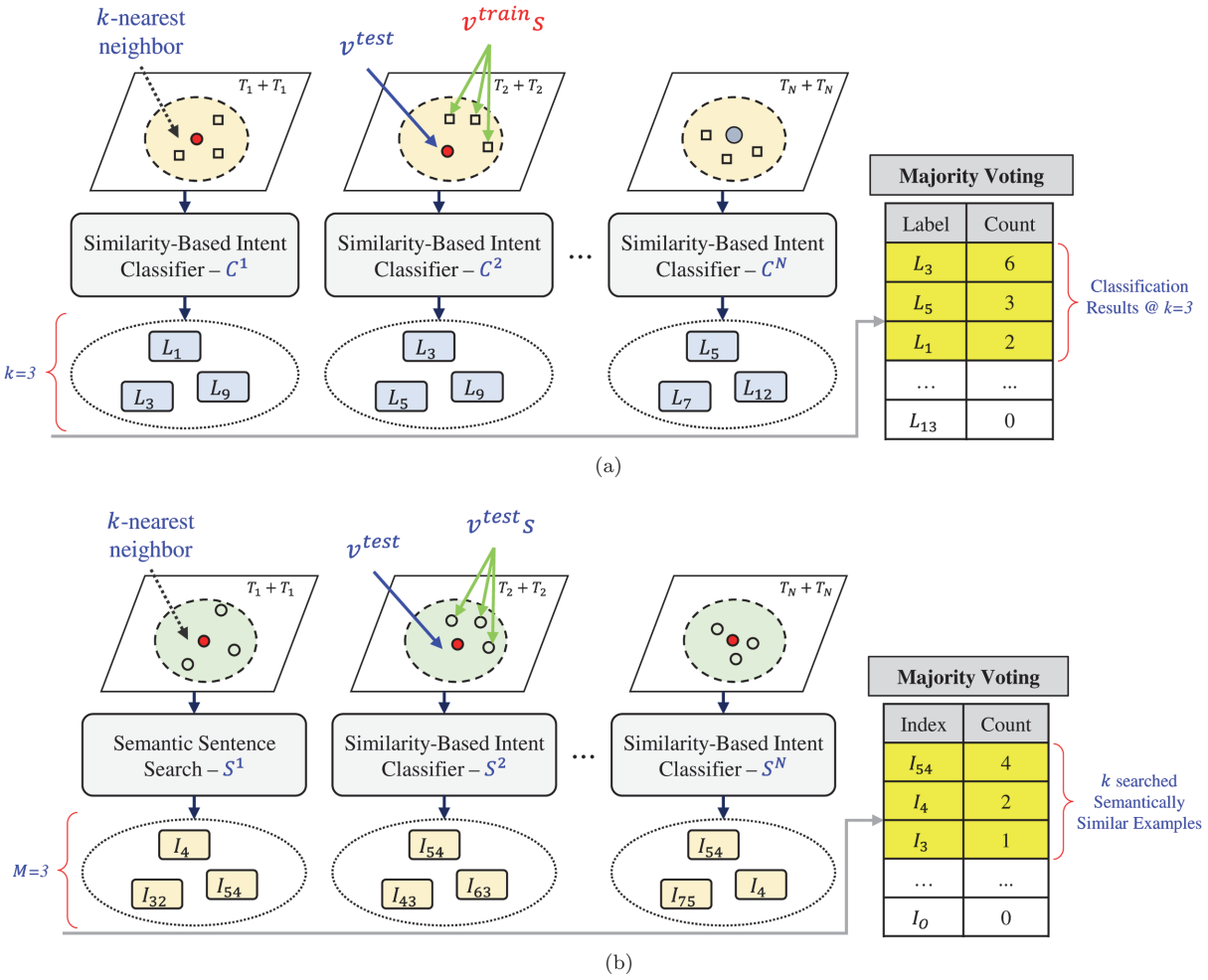

Fig. 7. Ensemble methods for (a) similarity-based intent classification and (b) semantic search in NLU, where T_k denotes a transformer type in the *k*th system.

vectors may differ from our expectations. In this study, we introduce two visualization methods empowered by the proposed semantic vector learning. The first is polar-coordinate semantic vector visualization, and the other is direct low-dimensional vector learning for visualization.

1) Polar-coordinate Semantic Vector Visualization

The proposed semantic vector learning method contains several unique features for correspondence learning in text and semantic frame representation. First, it uses the *cosine* distance, which is based on the *cosine* similarity for capturing the distance loss during the learning process. Second, the learning process actively considers the structure of a semantic frame, i.e., the intent, slot-tag, and slot-values. In this study, we introduce a novel polar-coordinate vector visualization method employing the *angle* information from the cosine distance and semantic frame structure. A polar coordinate system (https://en.wikipedia.org/wiki/Polar_coordinate_system) is a two-dimensional coordinate system in which each point on a plane is determined by the distance from a reference point and an angle from a reference direction. It represents a point labeled (r, θ) and is plotted on a polar grid. The polar grid is represented as a series of concentric circles radiating out from the pole or the origin of the coordinate plane. The reference point is called the *pole*, and the ray from the pole in the reference direction is the polar axis. The distance from the pole is called the *radial* coordinate (r), and the angle is called the *angular* coordinate (θ), or the polar angle.

A typical polar-coordination setting is applied through a conversion from the Cartesian coordinates using the Pythagorean theorem and trigonometry. However, in this study, we set the coordinates differently instead of using typical conversion methods. We set the radial and angular coordinates directly considering our visualization purpose.

Angular-Coordination: In this research, the polar-coordinated visualization is designed to more effectively show instances that are most similar to a query instance.

To achieve this, all angles between a query vector and all other testing vectors are calculated, and the angles are used for the *angular* value, θs, in terms of degree. In other words, a query vector is positioned at 0°, and all other instances should be placed apart at a certain angle. Because the vectors used in the proposed framework were trained to satisfy the *cosine* distance, similar instances will have a small degree and dissimilar instances will have a large degree.

Radial-Coordination: Typically, the radial value r is set as the distance between the origin and query points. However, in this study, we use the radial coordination for effective *intent* visualization. As shown in Fig. 8, the radial axis is divided into several layers, and each layer corresponds to the intent labels in the dataset. For example, as shown in Fig. 8, if the query instance has intent I_k, then the point is depicted in the kth radial layer by zero degrees. If the comparing instance has intent tag $I_{k'}$ and an angle of 45° from a query instance, then the compared point is depicted in the k'th layer 45° from the query point counter-clockwise

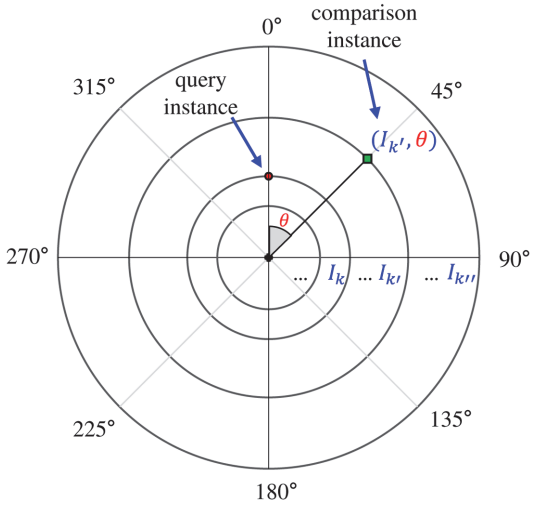

Fig. 8. Semantic vector visualization method based on polar coordination system. Here, I_k denotes an intent tag in the target domain, and θ is the angle between the query and compared instance semantic vector. The red circle is a query example and the green rectangle is the comparison example.

2) Visualization by Low-dimensional Vector Learning

Humans can effectively capture visual information when the figures are represented in 2-dimensions (2D) or 3-dimensions (3D). To acquire these 2D and 3D vectors, many visualization methods are introduced, including a principal component analysis, *t*-sne [14], and UMAP [15], among others. However, these methods directly or indirectly change the vector representation to transform higher dimensional data into lower dimensional data. In this research, we can change the target dimensions of the vector space into 2D or 3D. Because the proposed framework reads the text and semantic frame, and projects them to the final semantic vector space, which can be 2D or 3D without any structural limitations or ad-hoc dimension reduction. We can plot the target instances with the final projected semantic vectors.

V. EXPERIMENTS

We conducted experiments to determine whether the proposed pre-trained transformer-based encoders, cluster-aware framework, and ensemble and visualization techniques contribute to learning and plotting meaningful semantic vectors. To verify this, several experiments were designed to answer the following questions:

1. Is cluster-aware modeling effective?
2. Are pre-trained transformer-based readers effective?
3. Are learned semantic vectors semantically correct and helpful to NLU?
4. Are ensemble methods helpful in increasing the performance of semantic vector applications?
5. Do visualization techniques effectively plot the datasets?

For training and testing purposes, we used four datasets: ATIS [7] (https://github.com/SKTBrain/KoBERT), SNIPS [16], Simulated dialogue (Sim-R, https://github.com/google-research-datasets/simulateddialogue), and one Korean dataset, Weather. The internally collected Weather dataset is relatively simple and comprises short sentences providing weather information. The information in the dataset is summarized in Table 2.

The proposed cluster-aware separation modeling requires setting a triple margin value α. We use the same margin

Table 2. Dataset statistics

	ATIS	SNIPS	SimM	Weather
Number of trains	4,475	11,973	3,321	6,993
Number of dev.	499	685	1,032	3,009
Number of tests	882	685	2,286	2,998
Average number of words per sentence	11.27	9.07	8.37	4.80
Number of intents	19	8	22	15
Number of slot tags	80	40	6	25

value found by Jung and Lim [4] in all datasets.

To determine whether the cluster-aware modeling is indeed helpful in learning a better semantic representation, we traced the performance of similarity-based intent classification and a semantic search during the training progress over the testing datasets. Figs. 9 and 10 depict

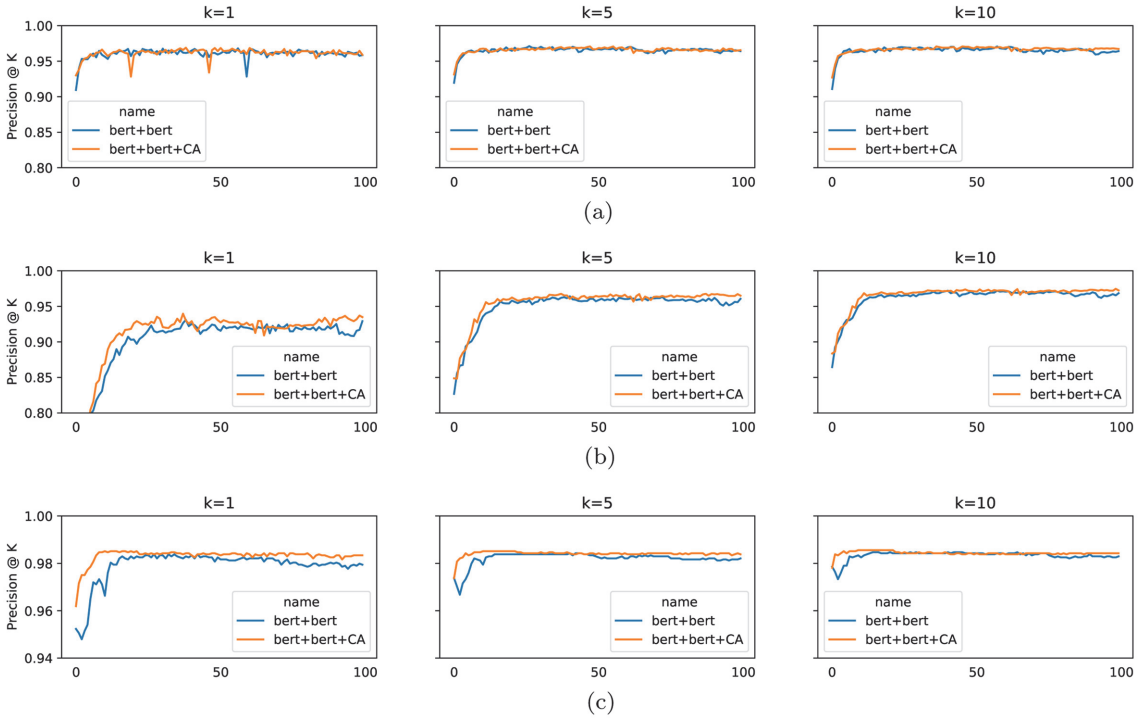

Fig. 9. Performance curve of SimM testing dataset over the training progress with BERT-based readers: (a) similarity-based intent classification, (b) semantic search with text vector as a query, and (c) semantic search with semantic frame vector as a query.

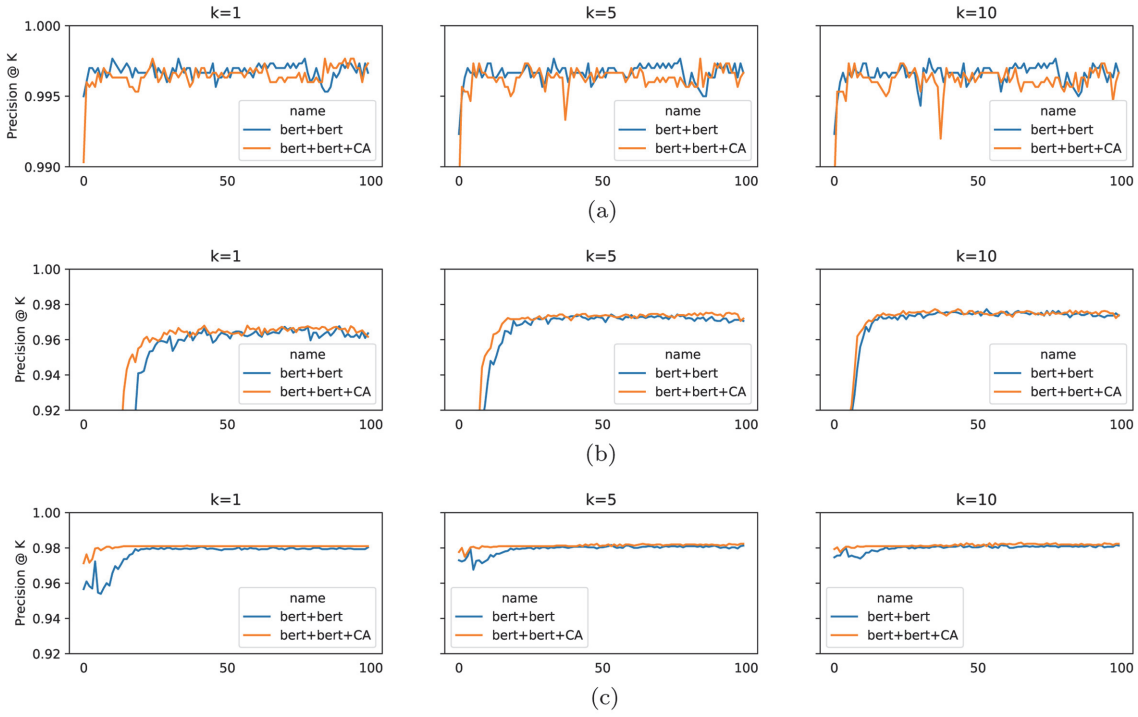

Fig. 10. Performance curve of the Weather testing dataset over the training progress using BERT-based readers: (a) similarity-based intent classification, (b) semantic search with text vector as a query, and (c) semantic search with semantic frame vector as a query.

the performance curves of BERT+BERT and BERT+BERT+CA over the training epochs in the SimM and Weather datasets used as English and Korean datasets, respectively.

As we can see in the figures, the cluster-aware models (+CA) showed a much faster learning curve and better performance results. In particular, clear performance gains can be found in a semantic search task. However, similarity-based intent classification is not as effective as a semantic search. It seems that the intent classification already showed high accuracy, and thus there is not much room for improvement by cluster-aware models. In addition, the semantic cluster was designed to cover the many slot-tags as well as intent-tag, and therefore cluster-aware models performed better in slot-tagging tasks than the baselines.

A. Sentence Search with Similarity

To prove the effects of the proposed methods, we conducted sentence search experiments with precision under K measurements over different datasets.

Table 3 shows that the proposed Transformer+Transformer and Transformer+Transformer+CA demonstrate a significantly improved search performance over the baseline model, LSTM+LSTM [2] through all datasets.

In particular, in the slot-matching task, the cluster-aware model (Transformer+Transformer+CA) outperformed LSTM+LSTM and the corresponding Transformer+Transformer. The proposed semantic frame cluster cohesion and separation modeling successfully worked in the formatting vector space more distinctly, and resulted in a better slot and intent-slot joint matching than the baseline and Transformer+Transformer. The effectiveness of cluster-aware modeling can be also founded by comparing E(BASELINES) and E(CA). In most cases, ensemble methods for cluster-aware models outperform other ensemble methods on non-cluster-aware models. Another interesting point, E(ALL), which is an ensemble method using all models, also shows a slightly better performance than E(CA). This implies that the proposed ensemble methods can effectively consider and merge multiple semantic vector projectors. Note that the performances of this research are slightly different from those of Jung [3], Jung and Lim [4], particularly in LSTM+LSTM because we split the datasets into training, testing, and validation datasets, and we stopped the training at 100 epochs for the pair comparisons. In most cases, LSTM models are insufficiently learned in 100 epochs.

B. Similarity-based Intent Classification

The proposed semantic representation learning framework is also helpful for the NLU task itself. To verify this, similarity-based intent classification experiments were conducted. The proposed models are found to be effective in capturing intent information into vector representations (Tables 4 and 5). Note that, in most cases, transformer-based semantic frame readers show a much better intent classification performance (e.g., LxL). This implies that the pre-trained transformer contributes to better semantic vector learning despite being pre-trained on raw texts and not on structured semantic frames. In the case of the Korean dataset, all models except LxL show 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. In an intent classification task, ensemble methods outperform all other models with a clear gap. In particular, E(ALL) shows much better results than E(BASELINES) and E(CA). This is additional evidence that the proposed ensemble methods can effectively consider and merge multiple semantic vector projectors.

C. Visualization

In this study, we proposed new ways of semantic vector visualization using polar-coordinate systems and low-dimensional semantic vector learning.

1) Polar-coordinate Semantic Vector Visualization

Fig. 11 shows examples of polar-coordinate visualizations in the datasets. A query example (marked by the star-symbol) was chosen from the test dataset and plotted with the top-20 closest examples of the training dataset in terms of the cosine distance. As we can see in Fig. 11, the closest training examples are placed near the query testing example at a certain degree apart. A more interesting point is that similar examples share many slot-tags with a query example. For example, in the Weather dataset on polar charts, the query example has slot tags [“day.p”], as do the closest examples [“day.p”], [“day.p,” “ti range.p”], [“day.p,” “ti range.p”], and [“day.p,” “ti range.p”, “time”].

2) Visualization by Low-Dimensional Vector Learning

In this research, we can directly change the target dimension of the vector space to two or three dimensions for visualization. Fig. 12 shows 2D and 3D test dataset visualizations of the ATIS and Weather datasets as English and Korean representations, respectively. From the figure, we can see clear semantic cluster groups even in lower dimensional semantic vector learning.

VI. RELATED STUDIES

Recent studies have focused on enriching the representations for neural architectures when implementing NLU. For example, Chen et al. [17] focused on leveraging substructure

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

		Precision @ 1						Precision @ 5					
		text as query			semantic frame as query			text as query			semantic frame as query		
		I	S	J	I	S	J	I	S	J	I	S	J
ATIS	LSTM x LSTM	0.558	0.127	0.111	0.985	0.483	0.482	0.790	0.300	0.271	0.988	0.781	0.776
	LSTM x LSTM+CA	0.560	0.093*	0.077**	0.998**	0.357**	0.357**	0.781	0.294	0.261	0.998**	0.718**	0.718**
	BERT x BERT	0.986**	0.768**	0.766**	0.999**	0.772**	0.772**	0.993**	0.787**	0.785**	0.999**	0.791	0.791*
	BERT x BERT+CA	0.982**	0.769**	0.764**	0.999**	0.781**	0.781**	0.989**	0.789**	0.786**	0.999**	0.793	0.793**
	ALBERT x ALBERT	0.985**	0.751**	0.748**	0.999**	0.757**	0.757**	0.990**	0.780**	0.777**	0.999**	0.788	0.788
	ALBERT x ALBERT+CA	0.989**	0.762**	0.762**	0.998**	0.762**	0.762**	0.992**	0.788**	0.786**	0.999**	0.788	0.788
	ROBERTA x ROBERTA	0.990**	0.769**	0.766**	0.998**	0.763**	0.763**	0.991**	0.790**	0.788**	0.999**	0.789	0.789*
	ROBERTA x ROBERTA+CA	0.985**	0.765**	0.763**	0.999**	0.761**	0.761**	0.991**	0.785**	0.780**	0.999**	0.789	0.789*
	E(BASELINES)	0.985**	0.738**	0.737**	0.998**	0.774**	0.774**	0.997**	0.788**	0.783**	0.999**	0.797*	0.796**
	E(CA)	0.988**	0.738**	0.737**	0.999**	0.764**	0.764**	0.994**	0.789**	0.785**	0.999**	0.798**	0.798**
	E(ALL)	0.991**	0.761**	0.760**	0.999**	0.782**	0.782**	0.995**	0.793**	0.789**	0.999**	0.798**	0.798**
SNIPS	LSTM x LSTM	0.702	0.096	0.096	1.000	0.835	0.835	0.917	0.302	0.299	1.000	0.841	0.841
	LSTM x LSTM+CA	0.762**	0.145**	0.145**	1.000	0.764**	0.764**	0.931	0.345*	0.345*	1.000	0.801**	0.801**
	BERT x BERT	0.987**	0.765**	0.765**	1.000	0.853**	0.853**	0.993**	0.823**	0.823**	1.000	0.853*	0.853*
	BERT x BERT+CA	0.984**	0.791**	0.791**	1.000	0.853**	0.853**	0.994**	0.818**	0.818**	1.000	0.853*	0.853*
	ALBERT x ALBERT	0.991**	0.747**	0.747**	1.000	0.842	0.842	0.993**	0.825**	0.825**	1.000	0.848	0.848
	ALBERT x ALBERT+CA	0.993**	0.768**	0.768**	1.000	0.853**	0.853**	0.996**	0.823**	0.823**	1.000	0.853*	0.853*
	ROBERTA x ROBERTA	0.988**	0.769**	0.769**	1.000	0.851**	0.851**	0.994**	0.828**	0.828**	1.000	0.853*	0.853*
	ROBERTA x ROBERTA+CA	0.988**	0.788**	0.788**	1.000	0.853**	0.853**	0.991**	0.822**	0.822**	1.000	0.853*	0.853*
	E(BASELINES)	0.984**	0.746**	0.746**	1.000	0.851**	0.851**	0.997**	0.836**	0.836**	1.000	0.853*	0.853*
	E(CA)	0.981**	0.762**	0.762**	1.000	0.848*	0.848*	1.000**	0.839**	0.839**	1.000	0.853*	0.853*
	E(ALL)	0.987**	0.784**	0.784**	1.000	0.853**	0.853**	0.999**	0.842**	0.842**	1.000	0.853*	0.853*
SimM	LSTM x LSTM	0.499	0.421	0.367	0.996	0.985	0.985	0.766	0.675	0.581	0.997	0.986	0.985
	LSTM x LSTM+CA	0.524	0.575**	0.406**	0.999*	0.790**	0.790**	0.801**	0.739**	0.618**	0.999	0.895**	0.894**
	BERT x BERT	0.960**	0.929**	0.897**	1.000**	0.979**	0.979**	0.980**	0.961**	0.941**	1.000*	0.982**	0.982*
	BERT x BERT+CA	0.956**	0.935**	0.901**	1.000**	0.983	0.983	0.976**	0.965**	0.941**	1.000*	0.984	0.984
	ALBERT x ALBERT	0.958**	0.917**	0.887**	1.000**	0.975**	0.975**	0.978**	0.957**	0.936**	1.000*	0.983*	0.983
	ALBERT x ALBERT+CA	0.957**	0.936**	0.904**	1.000**	0.982*	0.982*	0.980**	0.968**	0.949**	1.000*	0.984	0.984
	ROBERTA x ROBERTA	0.963**	0.904**	0.879**	1.000**	0.976**	0.976**	0.978**	0.951**	0.933**	1.000*	0.982**	0.982*
	ROBERTA x ROBERTA+CA	0.959**	0.935**	0.904**	1.000**	0.982*	0.982*	0.979**	0.967**	0.945**	1.000*	0.983*	0.983
	E(BASELINES)	0.933**	0.898**	0.870**	1.000*	0.982*	0.982*	0.985**	0.967**	0.950**	1.000*	0.986**	0.985**
	E(CA)	0.946**	0.927**	0.893**	1.000**	0.981**	0.981**	0.984**	0.973**	0.953**	1.000*	0.986	0.985**
	E(ALL)	0.960**	0.941**	0.913**	1.000**	0.983	0.983	0.986**	0.976**	0.958**	1.000*	0.985	0.985**
Weather	LSTM x LSTM	0.498	0.207	0.119	1.000	0.972	0.972	0.825	0.553	0.344	1.000	0.976	0.976
	LSTM x LSTM+CA	0.507	0.212	0.129	1.000	0.972	0.972	0.824	0.567	0.369*	1.000	0.975	0.975
	BERT x BERT	0.995**	0.964**	0.961**	1.000	0.980**	0.980**	0.997**	0.971**	0.968**	1.000	0.981**	0.981**
	BERT x BERT+CA	0.996**	0.962**	0.961**	1.000	0.981**	0.981**	0.997**	0.972**	0.969**	1.000	0.982**	0.981**
	KOBERT x KOBERT	0.997**	0.969**	0.966**	1.000	0.969	0.969	0.998**	0.974**	0.972**	1.000	0.979	0.978
	KOBERT x KOBERT+CA	0.998**	0.970**	0.968**	1.000	0.967	0.967	0.998**	0.977**	0.975**	1.000	0.979	0.978
	E(BASELINES)	0.927**	0.871**	0.851**	1.000	0.979**	0.979**	0.998**	0.976**	0.974**	1.000	0.981**	0.981**
	E(CA)	0.927**	0.867**	0.850**	1.000	0.980**	0.980**	0.998**	0.977**	0.975**	1.000	0.982**	0.981**
	E(ALL)	0.983**	0.951**	0.944**	1.000	0.981**	0.981**	0.999**	0.978**	0.975**	1.000	0.982**	0.981**

I, S, and J denote intent, slot-tag, and joining of intent and slot-tag matching, respectively. A model identifier indicates the existence of a [text reader] + [semantic frame reader] + [cluster-aware model or not]. For example, BERT+BERT+CA stands for a model with a BERT-based text reader and a BERT-based semantic frame reader with cohesion and separation modeling. In addition, E(BASELINES) and E(CA) denote ensemble methods without and with clustering-aware models, respectively. E(All) stands for ensemble methods using all models.
*p<0.05, **p<0.01, significantly different from the baseline, LSTM+LSTM [2].

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

		LxL	LxL+CA	BxB	BxB+CA	AxA	AxA+CA	RxR	RxR+CA	E(BASE-LINES)	E(CA)	E(All)
ATIS	1	0.561	0.577	0.976**	0.977**	0.981**	0.982**	0.976**	0.980**	0.976**	0.982**	0.982**
	3	0.625	0.658*	0.977**	0.977**	0.981**	0.982**	0.976**	0.981**	0.978**	0.983**	0.983**
	5	0.677	0.690	0.976**	0.977**	0.980**	0.982**	0.975**	0.980**	0.981**	0.983**	0.983**
	10	0.702	0.721**	0.977**	0.978**	0.978**	0.982**	0.977**	0.980**	0.976**	0.980**	0.980**
	40	0.713	0.713**	0.939**	0.941**	0.948**	0.933**	0.947**	0.939**	0.954**	0.949**	0.958**
SNIPS	1	0.596	0.634	0.985**	0.987**	0.987**	0.988**	0.985**	0.985**	0.985**	0.987**	0.987**
	3	0.647	0.663	0.985**	0.987**	0.987**	0.988**	0.985**	0.985**	0.987**	0.987**	0.988**
	5	0.664	0.688	0.985**	0.987**	0.987**	0.988**	0.985**	0.985**	0.987**	0.987**	0.988**
	10	0.677	0.709	0.985**	0.987**	0.987**	0.988**	0.985**	0.985**	0.987**	0.987**	0.988**
	40	0.669	0.691	0.985**	0.987**	0.990**	0.988**	0.985**	0.985**	0.987**	0.987**	0.988**
SimM	1	0.460	0.495**	0.958**	0.959**	0.958**	0.954**	0.959**	0.960**	0.967**	0.965**	0.968**
	3	0.513	0.516	0.965**	0.965**	0.961**	0.957**	0.963**	0.966**	0.969**	0.969**	0.972**
	5	0.517	0.540	0.964**	0.965**	0.964**	0.961**	0.964**	0.965**	0.969**	0.968**	0.970**
	10	0.522	0.527	0.964**	0.967**	0.965**	0.960**	0.965**	0.967**	0.971**	0.969**	0.972**
	40	0.488	0.551**	0.960**	0.959**	0.958**	0.951**	0.957**	0.958**	0.962**	0.960**	0.962**

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. In addition, L, B, A, R, and K represent LSTM, BERT, ALBERT, RoBERTa, and KoBERT, respectively. E(BASELINES) and E(CA) denote ensemble methods with and without clustering-aware models. E(All) stands for ensemble methods using all models.
*p<0.05, **p<0.01, significantly different from the baseline, LSTM+LSTM [2].

Table 5. Similarity-based intent classification results (accuracy): Korean dataset of Weather

		LxL	LxL+CA	BxB	BxB+CA	KxK	KxK+CA	E(BASELINES)	E(CA)	E(All)
Weather	1	0.505	0.519	0.997**	0.997**	0.998**	0.998**	0.997**	0.998**	0.998**
	3	0.535	0.540	0.997**	0.997**	0.997**	0.998**	0.998**	0.998**	0.998**
	5	0.554	0.567	0.997**	0.997**	0.997**	0.998**	0.998**	0.998**	0.998**
	10	0.575	0.568	0.997**	0.997**	0.997**	0.998**	0.997**	0.998**	0.998**
	40	0.561	0.571	0.996**	0.996**	0.996**	0.998**	0.998**	0.998**	0.998**

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. In addition, L, B, A, R, and K represent LSTM, BERT, ALBERT, RoBERTa, and KoBERT, respectively. E(BASELINES) and E(CA) denote ensemble methods with and without clustering-aware models. E(All) stands for ensemble methods using all models.
*p<0.05, **p<0.01, significantly different from the baseline, LSTM+LSTM [2].

embedding for joint semantic frame parsing. In addition, Kim et al. [6] utilized several semantic lexicons, such as WordNet, PPDB, and the Macmillan dictionary, to enrich a word embedding and subsequently used them in the initial representation of words for intent detection. Furthermore, many structured text-to-vector techniques have recently been introduced. For example, Preller [18] introduced a logic formula embedding method, whereas Bordes et al. [19] and Do et al. [20] the translation of symbolically structured knowledge such as WordNet and freebase is proposed.

In addition, Jung et al. [1] and Jung [2] introduced a novel semantic frame embedding method by simultaneously executing the raw text-to-vector 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, Jung [3] extended LSTM-based text and semantic frame readers to include transformer-based readers, and Jung and Lim [4] introduced a cluster-aware modeling technique in semantic vector learning employing a triplet-margin loss.

Recently, contextual sentence modeling has also been introduced in contextual language models including ELMo [21], GPT [22], BERT [5], and XLNet [23]. In

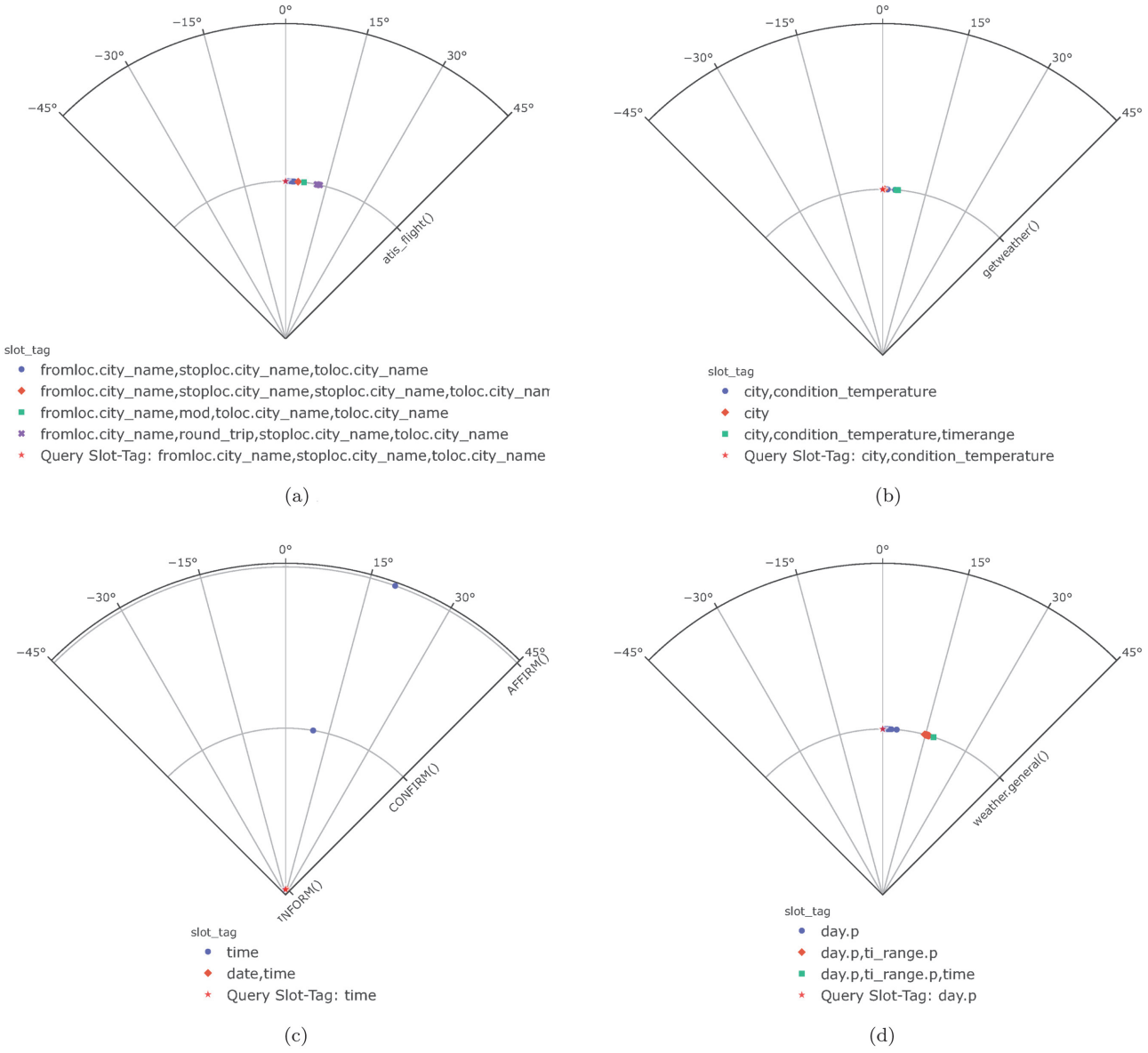

Fig. 11. Polar coordinate visualization examples in (a) ATIS, (b) SNIPS, (c) SimM, and (d) Weather datasets. A query example (star-symbol-marked) was chosen from the test dataset and plotted with top-20 closest examples of the training dataset in terms of cosine distance.

particular, BERT uses a different pre-training objective, i.e., a masked language model, and a next-sentence prediction task that jointly pre-trains the text-pair representations.

VII. CONCLUSION

Herein, we employed pre-trained transformers including BERT, RoBERTa, and ALBERT for text and semantic frames and extended a neural architecture for learning cluster-aware semantic representations using transformers. To learn a valid and meaningful cluster-wise distributed semantic representation, two properties, i.e., semantic cluster cohesion and separation were considered. A semantic cluster is defined as a set of semantic frames and paraphrased texts that share the same intent and slot-tags. For cluster cohesion modeling, the distance between the semantic centroid vector and text vectors in the same cluster is minimized. For cluster separation modeling, triplet margin loss is employed to push the semantic clusters apart for a clearer distinction.

In addition, we introduce elegant ensemble methods to improve the performance of a similarity-based intent classification and semantic search. The majority voting techniques are used for classification and search tasks, and showed a better performance in most cases. As an

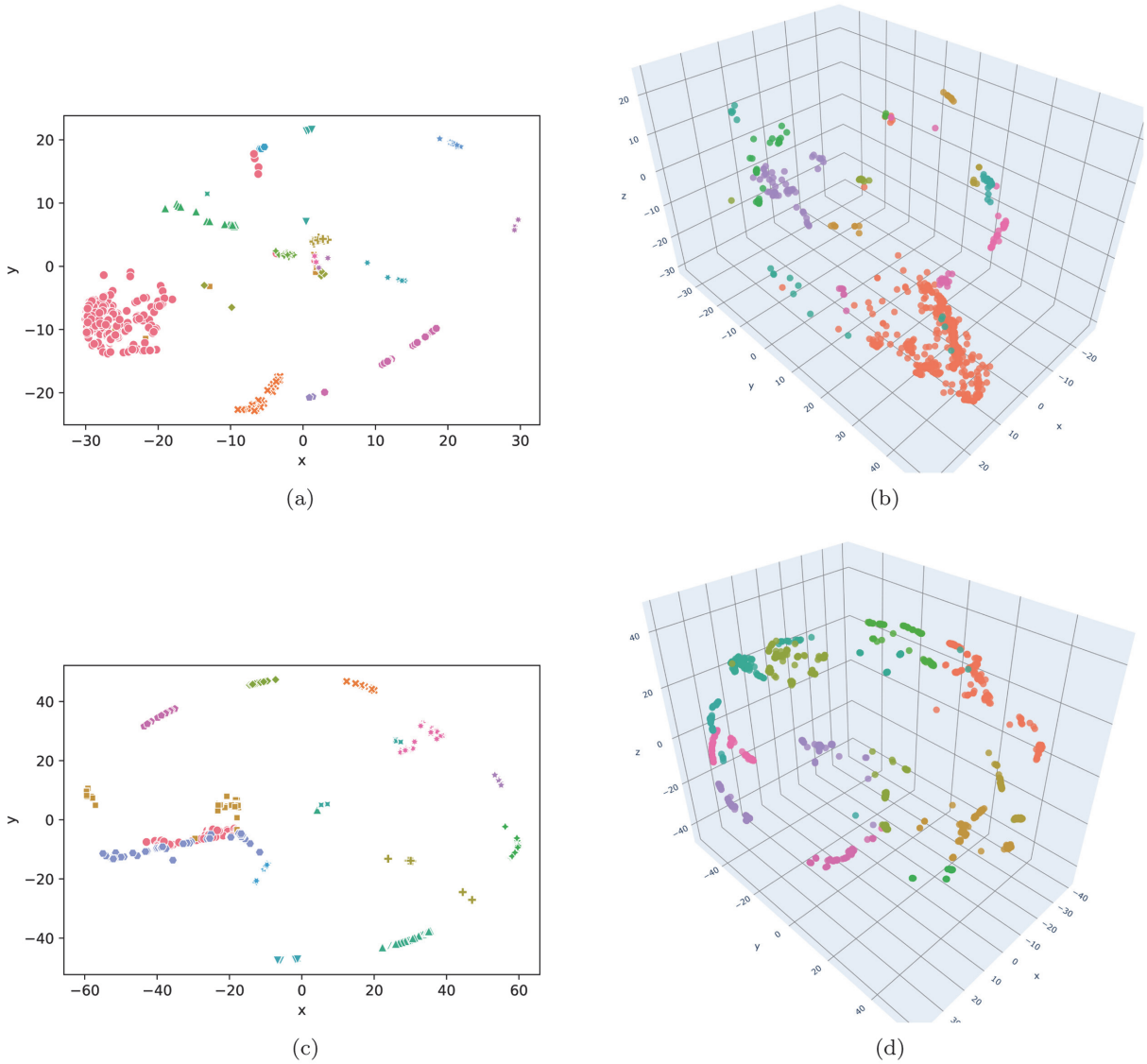

Fig. 12. Test dataset visualizations in ATIS and Weather datasets using low-dimensional vector learning: (a) ATIS in 2D, (b) ATIS in 3D, (c) Weather in 2D, and (d) Weather in 3D. The different colors for each point represent intent-tags.

NLU dataset visualization, polar-coordinate and low-dimensional semantic vector learning are employed to show an effective example plotting with semantic vectors.

Through experiments with the ATIS, SNIPS, SimM, and Weather datasets, we confirmed that the proposed methods can successfully learn semantic vector representations and consider the relationship between a text-to-semantic frame and semantic clusters. Based on the results of the proposed research, various research directions can be considered in the future. Semantic operations or algebra within a vector space are promising research topics. Furthermore, we intend to study the shared semantic vector space between multiple datasets or a general domain dataset

ACKNOWLEDGMENT

This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-01441, Artificial Intelligence Convergence Research Center, 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, "Semantic vector learning using pretrained transformers in natural language understanding," *Journal of Computing Science and Engineering*, vol. 14, no. 4, pp. 154-162, 2020.
4. S. Jung and S. Lim, "Cluster-aware semantic vector learning using BERT in natural language understanding," in *Proceedings of 2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju, South Korea, 2021, pp. 91-98.
5. 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 (NAACL-HLT)*, Minneapolis, MN, 2018, pp. 4171-4186.
6. 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.
7. P. Price, P. (1990). "Evaluation of spoken language systems: the ATIS domain," in *Speech and Natural Language: Proceedings of a Workshop*, Hidden Valley, PA, 1990.
8. D. Hakkani-Tur, G. Tur, A. Celikyilmaz, Y. N. Chen, J. Gao, L. Deng, and Y. Y. Wang, "Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, San Francisco, CA, 2016, pp. 715-719.
9. M. Henderson, B. Thomson, and J. Williams, "Dialog state tracking challenge 2 & 3," 2013 [Online]. Available: https://github.com/matthen/dstc/blob/master/handbook.pdf.
10. 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 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.
11. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, et al., "RoBERTa: arobustly optimized BERT pretraining approach," 2019 [Online]. Available: https://arxiv.org/abs/1907.11692.
12. V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, York, UK, 2016.
13. C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," 2017 [Online]. Available: https://arxiv.org/abs/1705.02304.
14. L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579-2605, 2018.
15. L. McInnes, J. Healy, and J. Melville, "UMAP: uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, article no. 861, 2018. https://doi.org/10.21105/joss.00861
16. 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 [Online]. Available: https://arxiv.org/abs/1805.10190.
17. 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.
18. A. Preller, "From logical to distributional models," 2014 [Online]. Available: https://arxiv.org/abs/1412.8527.
19. 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, 2013.
20. K. Do, T. Tran, and S. Venkatesh, "Knowledge graph embedding with multiple relation projections," 2018 [Online]. Available: https://arxiv.org/abs/1801.08641.
21. M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, New Orleans, LA, 2018, pp. 2227-2237.
22. A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018 [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.
23. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: generalized autoregressive pretraining for language understanding," 2019 [Online]. Available: https://arxiv.org/abs/1906.08237.

Sangkeun Jung

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