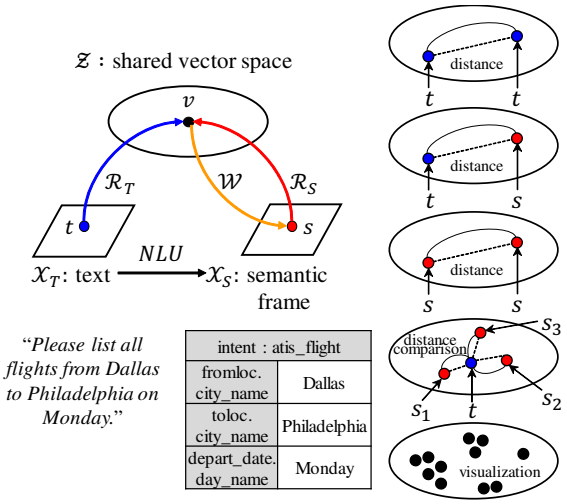
Learning to Embed Semantic Correspondence for Natural Language Understanding

Sangkeun Jung[1], Jinsik Lee[2], and Jiwon Kim[3]

[1,2,3]T-Brain, AI Research Center, SK telecom
[1]Dept. of Computer Science and Engineering, Chungnam National University
[1]hugmanskj@gmail.com
[2,3] {jinsik16.lee, jk}@sktbrain.com

Abstract

While learning embedding models has yielded fruitful results in several NLP subfields, most notably Word2Vec, embedding correspondence has relatively not been well explored especially in the context of natural language understanding (NLU), a task that typically extracts structured semantic knowledge from a text. A NLU embedding model can facilitate analyzing and understanding relationships between unstructured texts and their corresponding structured semantic knowledge, essential for both researchers and practitioners of NLU. Toward this end, we propose a framework that learns to embed semantic correspondence between text and its extracted semantic knowledge, called *semantic frame*. One key contributed technique is semantic frame reconstruction used to derive a one-to-one mapping between embedded vectors and their corresponding semantic frames. Embedding into semantically meaningful vectors and computing their distances in vector space provide a simple, but effective way to measure semantic similarities. With the proposed framework, we demonstrate three key areas where the embedding model can be effective: visualization, semantic search and re-ranking.



(a) Framework (b) Applications

Figure 1: Semantic vector learning framework and applications. We assume a pair of corresponding text and semantic frame (t, s), which has semantically the same meaning in a raw text domain (χ_T), and a semantic frame domain (χ_S) can be encoded to a vector v in a shared embedding vector space $\mathcal{Z}$. $\mathcal{R}_T$ and $\mathcal{R}_S$ are two reader functions that encode raw and structured text to a semantic vector. $\mathcal{W}$ is a writing function that decodes a semantic vector to a symbolic semantic frame.

1 Introduction

The goal of NLU is to extract meaning from a natural language and infer the user intention. NLU typically involves two tasks: identifying user intent and extracting domain-specific entities, the second of which is often referred to as slot-filling (Mesnil et al., 2013; Jeong and Lee, 2006; Kim et al., 2016). Typically, the NLU task can be viewed as an extraction of structured text from a raw text. In NLU literature, the structured form of intent and filled slots is called a *semantic frame*.

In this study, we aim to learn the meaningful distributed semantic representation, rather than focusing on building the NLU system itself. Once we obtained a reliable and reasonable semantic representation in a vector form, we can devise many useful and new applications around the NLU (Figure 1). Because all the instances of text and semantic frame are placed on a single vector space, we can obtain the natural and direct distance measure between them. Using the distance measure, the similar text or semantic frame instances can

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 131–140
Brussels, Belgium, October 31 - November 1, 2018.

be searched directly and interchangeably by the distance comparison. Moreover, re-ranking of multiple NLU results can be applied without further learning by comparing the distances between the text and the corresponding predicted semantic frame. Converting symbols to vectors makes it possible to do visualization naturally as well.

In this study, we assumed that the reasonable semantic vector representation satisfies the following properties.

- **Property - embedding correspondence**: Distributed representation of text should be the same as the distributed representation of the corresponding semantic frame.
- **Property - reconstruction**: Symbolic semantic frame should be recovered from the learned semantic vector.

We herein introduce a novel semantic vector learning framework called ESC (Embedding Semantic Correspondence learning), which satisfies the assumed properties.

The remainder of the paper is structured as follows: Section 2 describes the detailed structure of the framework. Section 3 introduces semantic vector applications in NLU. Section 4 describes the experimental settings and results. Section 5 discusses the related work. Finally, section 6 presents the conclusion.

2 ESC Framework

Our framework consists of ***text reader***, ***semantic frame reader***, and ***semantic frame writer***. The text reader embeds a sequence of tokens to a distributed vector representation. The semantic frame reader reads the structured texts and encodes each to a vector. v_t represents a vector semantic frame derived from the text reader, and v_s represents a vector semantic frame derived from the semantic frame reader. Finally, the semantic frame writer generates a symbolic semantic frame from a vector representation.

2.1 Text Reader

A text reader (Figure 2), implementing a neural sentence encoder, reads a sequence of input tokens and encodes each to a vector. In this study, we used long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) for encoding input sequences. The encoding process can be defined as

$$\overrightarrow{h}_s = R_{text}(E_X(x_s), \overrightarrow{h}_{s-1})$$
$$v_t = sigmoid(\overrightarrow{h}_S)$$

where $s = \{1, 2, ..., S\}$ and $\overrightarrow{h}_s$ is the forward hidden states over the input sequence at time s; R_{text} is an RNN cell; and $E_X(x_s)$ is the token embedding function, which returns a distributed vector representation of token x at time s. The final RNN output $\overrightarrow{h}_S$ is taken as v_t, which is a semantic vector derived from the text.

2.2 Semantic Frame Reader

A semantic frame consists of structured tags such as the intent and slot-tag and slot-values. In this study, the intent tag is handled as a symbol, and the slot-tags and slot-values are handled as a sequence of symbols. For example, the sentence, "Please list all flights from Dallas to Philadelphia on Monday." is handled as

- intent tag : *atis_flight*
- slot-tag sequence :
  [*fromloc.city_name, toloc.city_name, depart_date.day_name*]
- slot-value sequence :
  [*Dallas, Philadelphia, Monday*].

The intent reader is a simple embedding function $v_{intent} = E_I(i)$, which returns a distributed vector representation of the intent tag i for a sentence.

Stacked LSTM layer is used to read the sequences of slot-tags and slot-values. $E_S(o)$ is a slot-tag embedding function with o as a token. $E_V(a)$ is an embedding function with a as a token. The embedding result $E_S(o_m)$ and $E_V(a_m)$ are concatenated at time-step m, and the merged vectors are fed to the stacked layer for each time-step (Figure 2). $v_{tag,value}$ - the reading result of sequence of slot-tags and values - is taken from the final output of RNN at time M. Finally, intent, slot-tag and value encoded vectors are merged to construct a distributed semantic frame representation as

$$v_s = sigmoid(W_{sf}([v_{intent}; v_{tag,value}]) + b_{sf})$$

where $[;]$ denotes the vector concatenation operator. The dimension of v_s is same as v_t. All embedding weights are randomly initialized and learned through the training process.

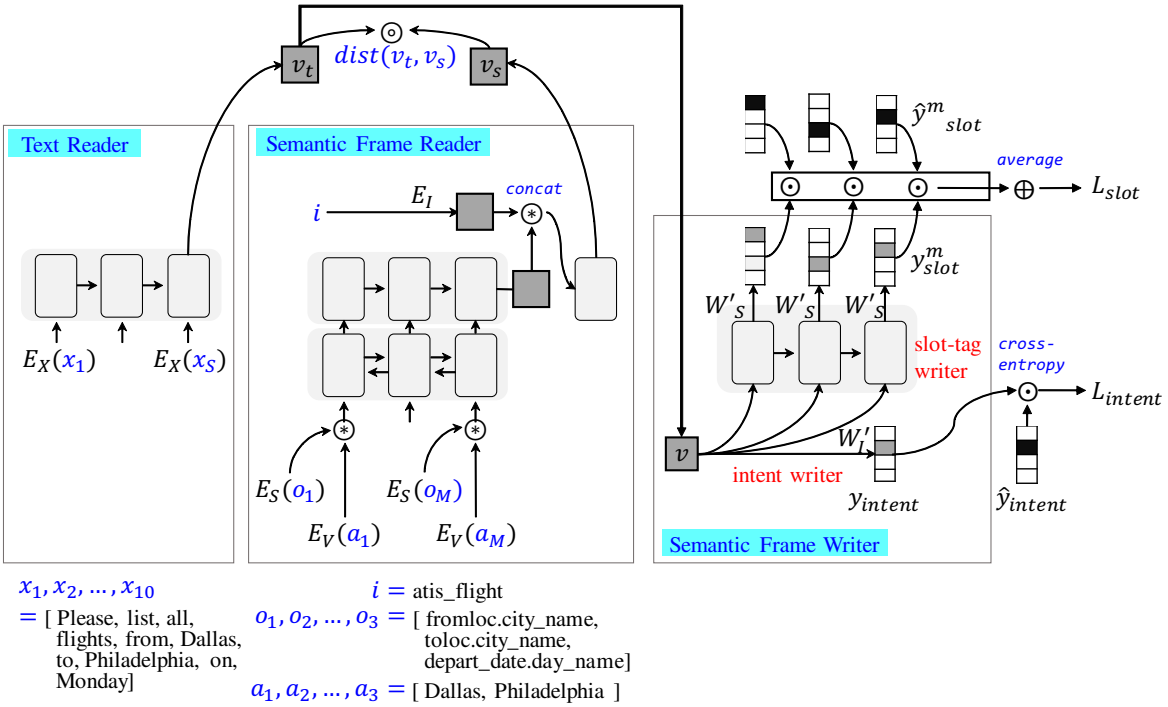


Figure 2: Text reader, semantic reader and semantic frame writer neural architecture. E_X is an embedding function for the input text token x. E_I, E_S, and E_V are the embedding functions for the intent tag, slot-tag and slot-value, respectively. $\circledast$ is a vector concatenation operation; $\odot$ is a cross-entropy; $\oplus$ is an average calculation; $\circledcirc$ represents the distance calculation. $\hat{y}_{intent}$ is a reference intent tag vector and $\hat{y}^m_{slot}$ is a reference slot tag vector at time m. M is the number of slots in a sentence (in the above example, $M = 3$).

2.3 Semantic Frame Writer and Loss Functions

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

Loss for Property "embedding correspondence" 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

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

where the $dist$ function can be any vector distance measure; however, in this study, we employed a Euclidean and a cosine distance (=1.0 - cosine similarity).

Loss for Property "reconstruction" Content loss provides a measure of how much semantic information the semantic frame vector contains. Without the content loss, v_t and v_s tend to quickly converge to zero vectors, implying the failure to learn the semantic representation. To measure the content keeping, symbolic semantic frame generations from semantic vector is performed, and the difference between the original semantic frame and the generated semantic frame is calculated.

Because the semantic frame's slot-value has a large vocabulary size to generate the slot values, a *reduced semantic frame* is devised to ease the generation problem. A reduced semantic frame is created by simply dropping the slot values from the corresponding semantic frame. For example, in Figure 2, slot values [*Dallas, Philadelphia, Monday*] are removed to create a reduced semantic frame. Content loss calculation is performed on this reduced semantic frame. Another advantage of employing reduced semantic frame is that the learned distributed semantic vectors have more abstract power because the learned semantic vectors are less sensitive to the lexical vocabulary.

For content loss, the intent and slot-tags' generation qualities are measured. The intent generation network can be simply defined using linear

Notation	Dim.	Description
E_X	50	Token embedding
E_S	50	Slot-tag embedding
E_V	50	Slot-value embedding
v_{intent}	50	Intent reader output
$v_{tag,value}$	200	Slot-tag and value reader output
v	200	Semantic vector

Table 1: Hyperparameters of the model.

projection as

$$y_{intent} = W_I^{'} v + b_I$$

where v is the semantic vector, and y_{intent} is the output vector.

The slot-tag generation networks are defined as

$$\overrightarrow{q}_m = R_G(v, \overrightarrow{q}_{m-1})$$
$$y_{slot}^m = W_S^{'} \overrightarrow{q}_m + b_S$$

where R_G is an RNN cell. The semantic vector v is copied and repeatedly fed into each RNN input. The outputs from the RNN are projected onto the slot tag space with $W_S^{'}$.

Figure 2 shows the intent and slot tag generation networks and the corresponding loss calculation methods. The generational losses can be defined with the cross entropy between the generated tag vector and the reference tag vector 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 M is the number of slots in a sentence.

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

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

Finally, the total loss value (L) for learning the semantic frame representation is defined with the ***distance loss*** and ***content loss*** as

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

The hyperparameters of the proposed model are summarized in Table 1.

3 Applications

3.1 Multi-form Distance Measurement

Using the learned text- and semantic-frame reader, we can measure not only the instances from the same form (text or semantic frame form) but also from different forms. Let's denote a text as t and a semantic frame as s, and the text and semantic frame reader as $\mathcal{R}_T$ and $\mathcal{R}_S$, respectively. The distance measurements between them can be performed as follows:

- $dist(v_t^i, v_t^j)$:
$$t_i \rightarrow \mathcal{R}_T(t_i) = v_t^i$$
$$t_j \rightarrow \mathcal{R}_T(t_j) = v_t^j$$
- $dist(v_t^i, v_s^j)$:
$$t_i \rightarrow \mathcal{R}_T(t_i) = v_t^i$$
$$s_j \rightarrow \mathcal{R}_S(t_j) = v_s^j$$
- $dist(v_s^i, v_s^j)$:
$$s_i \rightarrow \mathcal{R}_S(s_i) = v_s^i$$
$$s_j \rightarrow \mathcal{R}_S(s_j) = v_s^j$$

3.2 Visualization

With vector semantic representation, we can visualize the instances (sentences) in an easier and more natural way. Once the symbolic text or semantic frame are converted to vector, vector visualization methods such as *t*-sne (Maaten and Hinton, 2008) can be used directly to check the relationship between instances or the distribution of the entire corpus.

3.3 Re-ranking Without Further Learning

Re-ranking the NLU results from multiple NLU modules is difficult but important if a robust NLU system is to be built. Typically, a choice is made by comparing the scores produced by each system. However, this technique is not always feasible because the scores are often in different scales, or are occasionally not provided at all (e.g., in the purely rule-based NLU systems). The vector form of the semantic frame provides a very clear and natural solution for the re-ranking problem.

Figure 3 shows the flow of the re-ranking algorithm with the proposed vector semantic representation. In this study, we reordered the NLU results from multiple NLU systems according to their corresponding distances of v_t to v_s. It is noteworthy that the proposed re-ranking algorithm does not require further learning for ranking such as ensemble learning or learning-to-rank techniques. Further, the proposed methods are applicable to any type of NLU system. Even purely rule-based systems can be satisfactorily compared to purely statistical systems.

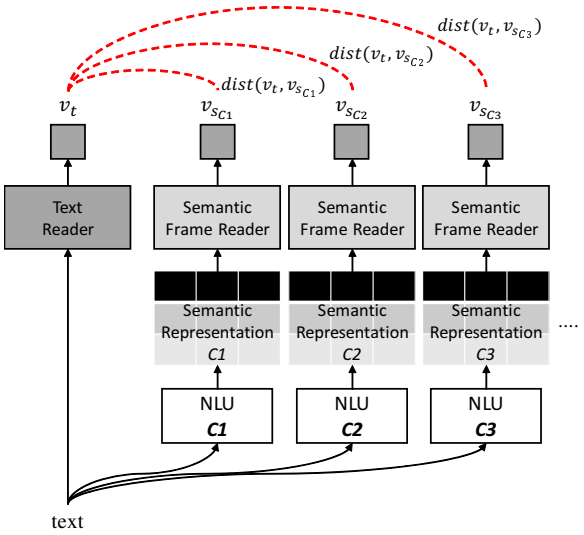
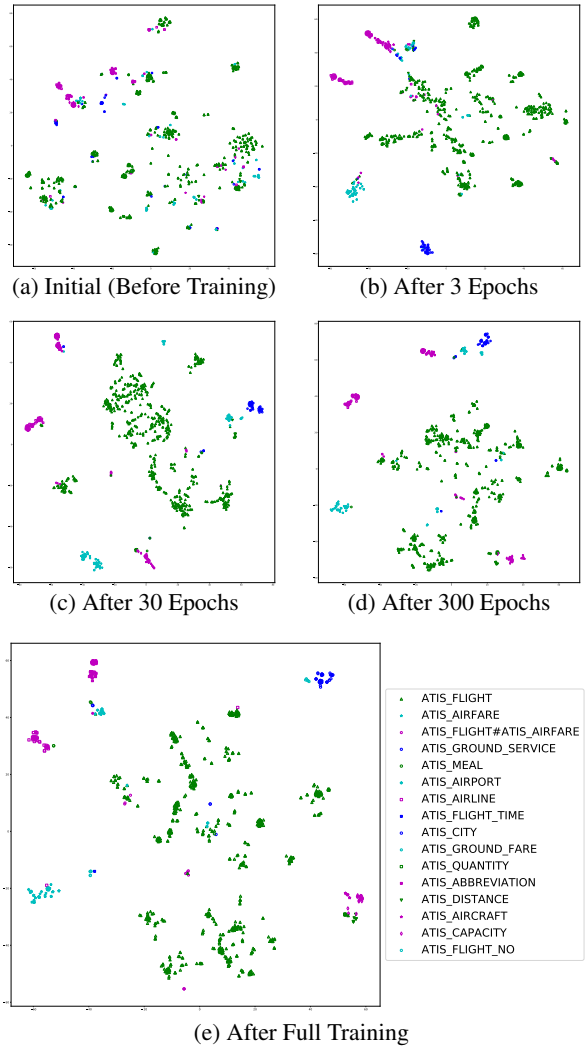


Figure 3: Re-ranking multiple NLU results using the semantic vector. The semantic vector from the text (v_t) functions as a pivot. We show three different NLU systems in this illustration.



(a) Initial (Before Training) (b) After 3 Epochs

(c) After 30 Epochs (d) After 300 Epochs

(e) After Full Training

Figure 4: Visualization of semantic vectors through training process. The plotted points are v_t from the text reader by t-sne processing in the testing sentences. The different colors and shape combinations represent different intent tags.

4 Experiments

For training and testing purposes, we used the ATIS2 dataset (Price, 1990). The ATIS2 dataset consists of an annotated intent and slot corpus for an air travel information search task. ATIS2 data set comes with a commonly used training and test split. For tuning parameters, we further split the training set into 90% training and 10% development set.

4.1 Validity of Learned Semantic Vector with Visualization

The intuition behind the proposed method is that semantically similar instances will be grouped together if the semantic vector learning is performed successfully. Figure 4 supports that the intuition is correct. In the early stages of training, the instances are scattered randomly; however, as the training progresses, semantically similar instances gather closer to each other. We observed that the proposed framework groups and depicts the sentences based on the intent tag remarkably well.

4.2 Multi-form Distance Measurement

In our framework, the instances having different forms (text or semantic frame) can be compared directly on a semantic vector space. To demonstrate that multi-form distance measurement works well, the sentence and semantic frame search results with a sentence and a semantic frame query are shown in Table 2.

Table 2 shows that *text to text* search is very well done with the learned vector. The retrieved sentence patterns are similar to the given text, and the vocabulary presented is also similar. On the other hand, in the case of the *text to semantic frame* search, The sentence patterns are similar, but the content words such as city name are not similar. In fact, this is what we predicted, because the content loss for reconstruction property is measured on *reduced semantic frame* which does not include slot-values. In *semantic frame to text* search, we can find similar behaviors. Retrieved results have almost same intent tag and slot-tags, but have dif-

No.	Text		Semantic Frame
"Show Delta Airlines from Boston to Salt Lake"			
1	Show Delta Airlines flights from Boston to Salt Lake	1	ATIS_FLIGHT airline_name: Delta Airlines fromloc.city_name: Boston toloc.city_name: Salt Lake
2	Show Delta Airlines flights from Boston to Salt Lake City	2	ATIS_FLIGHT airline_name: American Airlines fromloc.city_name: Phonenix toloc.city_name: Milwaukee
3	List Delta flights from Seattle to Salt Lake City	3	ATIS_FLIGHT airline_name: Delta Airlines fromloc.city_name: Montreal toloc.city_name: Orlando

(a) Text as Query

No.	Text		Semantic Frame
[ATIS_FLIGHT] flight_mod(last), depart_date.day_name(Wednesday), fromloc.city_name(Oakland), toloc.city_name(Salt Lake City)			
1	Get last flight from Oakland to Salt Lake City on Wednesday	1	ATIS_FLIGHT flight_mod: last depart_date.day_name: Wednesday fromloc.city_name: Oakland toloc.city_name: Salt Lake City
2	Get last flight from Oakland to Salt Lake City on Wednesday or first flight from Oakland to Salt Lake City on Thursday	2	ATIS_FLIGHT flight_mod: first depart_date.day_name: Thursday fromloc.city_name: Oakland toloc.city_name: Salt Lake City
3	Get first flight from Oakland to Salt Lake City on Thursday	3	ATIS_FLIGHT flight_mod: last depart_date.day_name: Wednesday fromloc.city_name: Oakland toloc.city_name: Salt Lake City or flight_mod: first depart_date.day_name: Thursday fromloc.city_name: Oakland toloc.city_name: Salt Lake City

(b) Semantic Frame as Query

Table 2: Example of most similar instance search results in the test data according to the proposed framework. Top-3 text and semantic frame retrieved instances given a single query are shown in left and right side respectively.

ferent city or airport names which are corresponding to slot-values. If we could include the slot-value generation in the reconstruction loss with large data, a better multi-form semantic search result might be expected.

To measure the quantitative search performance, *precision at K* are reported in Table 3. Precision at K corresponds to the number of same sentence pattern instances in the top K results. From the search result, we can conclude that the learned semantic vectors keep sentence pattern (intent tag and slot-tags) information very well.

K	Text Query			SF Query		
	I	S	J	I	S	J
1	98.30	63.15	62.93	99.43	70.75	70.52
3	99.09	72.45	72.00	99.55	77.78	77.44
5	99.09	75.17	74.72	99.77	78.80	78.68
10	99.09	76.98	76.42	99.77	80.39	80.27

Table 3: Same sentence pattern (intent and slot-tags should be matched) search performance (Precision @K). SF, I, S and J stand for semantic frame, intent, slot-tag and joint of intent and slot-tag.

4.3 Re-ranking

We prepared 11 NLU systems for re-ranking. Nine intent-/slot-combined classifiers and two in-

Identifier	Combinations	
C1	CNN	CRF
C2		RNN
C3		RNN+CRF
C4	MaxEnt	CRF
C5		RNN
C6		RNN+CRF
C7	SVM	CRF
C8		RNN
C9		RNN+CRF
C10	Joint Liu (Liu and Lane, 2016)	
C11	Joint Tur (Hakkani-Tür et al., 2016)	

Table 4: Multiple NLU systems for re-ranking.

tent/slot joint classifiers were implemented. For the combined classifiers, three intent classifiers and three slot sequential classifiers were prepared and combined. For the joint classifiers, those of Liu and Lane (2016) and Hakkani-Tür et al. (2016) were each implemented. Here, we did not significantly tune the NLU systems, as the purpose of this paper is to learn the semantic vector, not to build the state-of-the-art NLU systems.

A maximum-entropy (MaxEnt)- and a support vector machine (SVM)-based intent classifier were implemented as a traditional sentence classification method. Both classifiers share the same feature set (1-gram, 2-gram, 3-gram, and 4-gram around each word). Also, a convolutional-neural network-based (CNN-based) (Kim, 2014) sentence classification method was implemented.

A conditional random field (CRF)-based sequential classifier was implemented as a traditional slot classifier. Also, an RNN- and an RNN+CRF-based sequential classifier were implemented as a deep learning method. Bidirectional LSTMs were used to build the simple RNN-based classifier. By placing a CRF layer on top of the bidirectional LSTM network (Lee, 2017), an RNN+CRF-based network was implemented. In addition, two joint NLU systems (Liu and Lane, 2016; Hakkani-Tür et al., 2016) are prepared by reusing their codes, which are publicly accessible[1][2] [3].

Table 4 shows the summary of the NLU systems that we prepared and used for the re-ranking experiments.

Table 5 shows the performance of all the NLU systems, the proposed re-ranking algorithm's performance, and the oracle performance. Typical choices in re-ranking NLU results are majority voting and score-based ranking. In the majority voting method, the semantic frame most predicted by the NLU systems is selected. The score of the NLU scoring method in Table 5 is the prediction probability. In the case of joint NLU classifiers (C10 and C11), the joint prediction probabilities are used for the score. In the case of combination NLU systems (C1 to C9), the product of the intent and slot prediction probabilities is used for the score.

NLU systems	Intent	Slot		
	acc.	prec.	rec.	f_m
C1	90.70	94.36	89.61	91.92
C2	90.70	92.33	92.46	92.40
C3	90.70	93.53	92.39	92.96
C4	94.10	94.36	89.61	91.92
C5	94.10	92.33	92.46	92.40
C6	94.10	93.53	92.39	92.96
C7	91.84	94.36	89.61	91.92
C8	91.84	92.33	92.46	92.40
C9	91.84	93.53	92.39	92.96
C10	96.03	93.68	92.64	93.16
C11	93.54	94.74	94.00	94.37
random	92.86	93.96	92.29	93.12
majority vote (baseline)	94.10	**95.63**	93.68	94.64
NLU score (baseline)	95.58	94.81	93.89	94.35
re-ranked (Euclidean)	**97.05**	93.74	91.96	92.84
re-ranked (cosine)	**97.05**	95.40	**94.11**	**94.75**
oracle	97.85	96.77	95.29	96.02

Table 5: NLU performance of multiple NLU systems and re-ranked results. Acc., prec., rec., and f_m stand for accuracy, precision, recall, and f-measure, respectively.

The proposed distance-based re-ranking method using semantic vector shows superior selection performance at both intent and slot-filling tasks. It is noteworthy that the re-ranked intent prediction performance (acc. 97.05) is relatively close to the oracle intent performance (acc. 97.85), which is the upper bound. Compared to the baseline re-ranker (NLU score), the proposed re-ranker (cosine) achieves 33.25% and 7.07% relative error reduction for intent prediction and slot-filling task, respectively.

[1] https://github.com/yvchen/JointSLU.git

[2] https://github.com/DSKSD/RNN-for-Joint-NLU

[3] The reported performance of C10 and C11 in their paper were not reproduced with the open code.

5 Related Work

The task of spoken NLU consists of intent classification and domain entity slot filling. Traditionally, both tasks are approached using statistical machine-learning methods (Schwartz et al., 1997; He and Young, 2005; Dietterich, 2002). Recently, with the advances in deep learning, RNN-based sequence encoding techniques have been used to detect the intent or utterance type (Ravuri and Stolcke, 2015), and RNN-based neural architectures have been employed for slot-filling tasks (Mesnil et al., 2013, 2015). The combinations of CRF and neural networks have also been explored by Xu and Sarikaya (2013).

Recent works have focused on enriching the representations for neural architectures to implement NLU. For example, Chen et al. focused on leveraging substructure embeddings for joint semantic frame parsing (Chen et al., 2016). Kim et al. utilized several semantic lexicons, such as WordNet, PPDB, and the Macmillan dictionary, to enrich the word embeddings, and later used them in the initial representation of words for intent detection (Kim et al., 2016).

Previous NLU works have used statistical modeling for the intent and slot-filling tasks, and input representation. None of the work performed has represented the text and semantic frame as a vector form simultaneously. To our best knowledge, this is the first presentation of a method for learning the distributed semantic vector for both text and semantic frame and it's applications in NLU research.

In general natural language processing literature, many *raw text to vector* studies to learn the vector representations of text have been performed. Mikolov et al. (2013); Pennington et al. (2014); Collobert et al. (2011) proposed word to vector techniques. Mueller and Thyagarajan (2016); Le and Mikolov (2014) introduced embedding methods at the sentence and document level. Some attempts have shown that in this embedding process, certain semantic information such as analogy, antonym, and gender can be obtained in the vector space.

Further, many *structured text to vector* techniques have been introduced recently. Preller (2014) introduced a logic formula embedding method while Bordes et al. (2013); Do et al. (2018) proposed translating symbolic structured knowledge such as Wordnet and freebase.

We herein introduce a novel semantic frame embedding method by simultaneously executing the *raw text to vector* and *structured text to vector* method in a single framework to learn semantic representations more directly. In this framework, the text and semantic frame are each projected onto a vector space, and the distance loss between the vectors is minimized to satisfy *embedding correspondence*. Our research goes a step further to guarantee that the learned vector indeed keep the semantic information by checking the *reconstruction* the symbolic semantic frame from the vector.

In learning the parameters by minimizing the vector distances, this work is similar to a Siamese constitutional neural network (Chopra et al., 2005; Mueller and Thyagarajan, 2016) or an autoencoder (Hinton and Salakhutdinov, 2006); however, the weights are not shared or transposed in this work.

6 Conclusion

In this study, we have proposed a new method to learn a correspondence embedding model for NLU. To learn a valid and meaningful distributed semantic representation, two properties - *embedding correspondence* and *reconstruction* - are considered. By minimizing the distance between the semantic vectors which are the outputs of text and semantic frame reader, the semantically equivalent vectors are placed very close in the vector space. In addition, reconstruction consistency from a semantic vector to symbol semantic frame was jointly enforced to prevent the method from learning trivial degenerate mappings (e.g. mapping all to zeros).

Through various experiments with ATIS2 dataset, we confirmed that the learned semantic vectors indeed contain semantic information. Semantic vector visualization and the results of similar text and semantic frame search showed that semantically similar instances are actually located near on the vector space. Also, using the learned semantic vector, re-ranking multiple NLU systems can be implemented without further learning by comparing semantic vector values of text and semantic frame.

Based on the results of the proposed research, various research directions can be considered in the future. A semantic operation or algebra on a vector space will be a very promising research topic. Furthermore, with enough training data and appropriate modification to our method, adding

text reconstruction constraint can be pursed and generating text directly from a semantic vector would be possible, somewhat resembling problem settings of neural machine translation tasks.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Yun-Nung Chen, Dilek Hakanni-Tür, Gokhan Tur, Asli Celikyilmaz, Jianfeng Guo, and Li Deng. 2016. Syntax or semantics? knowledge-guided joint semantic frame parsing. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 348–355. IEEE.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Thomas Dietterich. 2002. Machine learning for sequential data: A review. *Structural, syntactic, and statistical pattern recognition*, pages 227–246.

Kien Do, Truyen Tran, and Svetha Venkatesh. 2018. Knowledge graph embedding with multiple relation projections. *arXiv preprint arXiv:1801.08641*.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Yulan He and Steve Young. 2005. Semantic processing using the hidden vector state model. *Computer speech & language*, 19(1):85–106.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minwoo Jeong and Gary Geunbae Lee. 2006. Jointly predicting dialog act and named entity for spoken language understanding. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 66–69. IEEE.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 414–419. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Changki Lee. 2017. Lstm-crf models for named entity recognition. *IEICE Transactions on Information and Systems*, 100(4):882–887.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(3):530–539.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Anne Preller. 2014. From logical to distributional models. *arXiv preprint arXiv:1412.8527*.

Patti J Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Suman V Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. In *INTERSPEECH*, pages 135–139.

Richard Schwartz, Scott Miller, David Stallard, and John Makhoul. 1997. Hidden understanding models for statistical sentence understanding. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1479–1482. IEEE.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE.