Doctoral Dissertation

Out–of–domain Sentence Detection based on Deep Learning for Dialog Systems

Seonghan Ryu (류 성 한)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2018


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

대화시스템을 위한 딥러닝 기반의 영역 외 문장 검출

Out-of-domain Sentence Detection based on Deep Learning for Dialog Systems

Out–of–domain Sentence Detection based on Deep Learning for Dialog Systems

by

Seonghan Ryu
Department of Computer Science and Engineering
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of philosophy in the Computer Science and Engineering

Pohang, Korea
12. 11. 2017
Approved by
Hwanjo Yu (Signature)
Academic advisor

Out–of–domain Sentence Detection based on Deep Learning for Dialog Systems

Seonghan Ryu

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a doctoral degree from POSTECH

12. 11. 2017

Committee Chair	Hwanjo Yu	(Seal)
Member	Jong-Hyeok Lee	(Seal)
Member	Gary Geunbae Lee	(Seal)
Member	Seungjin Choi	(Seal)
Member	Jungyun Seo	(Seal)

DCSE 류 성 한. Seonghan Ryu
20120577
Out–of–domain Sentence Detection based on Deep Learning for Dialog Systems,
대화시스템을 위한 딥러닝 기반의 영역 외 문장 검출
Department of Computer Science and Engineering , 2018, 65p, Advisor : Hwanjo Yu. Text in English.

ABSTRACT

To ensure satisfactory user experience, dialog systems must be able to determine whether an input sentence is *in–domain* or *out–of–domain* (OOD). We assume that only in–domain sentences are available as training data because collecting enough OOD sentences in an unbiased way is a laborious and time–consuming job. This dissertation proposes a novel sentence–embedding method that represents sentences in a low–dimensional continuous vector space that emphasizes aspects that distinguish in–domain cases from OOD cases. We first used a large set of unlabeled text to pre–train *word representations* that are used to initialize sentence embedding. Then we used domain–category analysis as an auxiliary task to train sentence embedding for OOD sentence detection; the learned sentence representations are used to train one–class classifiers. We propose two one–class classifiers for OOD sentence detection: an autoencoder that generates high reconstruction errors for OOD sentences and a generative adversarial network in which the discriminator generates low scores for OOD sentences. We evaluated our OOD sentence detection method by experimentally comparing it

to state–of–the–art methods in a 14–domain dialog system; our proposed method achieved the highest accuracy in all tests.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

Contents

List of Tables

List of Figures

I. Introduction

Dialog systems provide natural–language interfaces between humans and machines [1]. An architecture (Fig. 1.1) of dialog systems consists of five processes:

- **Automatic speech recognition (ASR)**: recognition of text from a user utterance.
- **Natural language understanding (NLU)**: extraction of a machine–readable semantic frame from the text.
- **Dialog management (DM)**: management of dialog flow by choosing the next system action.
- **Natural language generation (NLG)**: generation of a textual representation of the system action.
- **Text–to–speech synthesis (TTS)**: synthesis of a system utterance from the text.

Because human conversation can range among topics, many studies have been conducted on multi–domain dialog systems [2, 3, 4, 5, 6]. Multi–domain dialog systems that adopts the distributed architecture [7] first select a domain based on an input sentence, then execute the domain–specific processes of that domain. However, these systems are restricted to a closed set of target domains, and therefore cannot provide appropriate responses to *out–of–domain* (OOD) requests. For example, a dialog system that was designed to cover `schedule` and `message` domains could receive OOD requests such as "*Would you recommend Italian restaurants for me?*" or "*Please record Game of Thrones.*". To improve user experience, the system must detect OOD requests and provide appropriate back–off responses rather than providing unrelated responses.

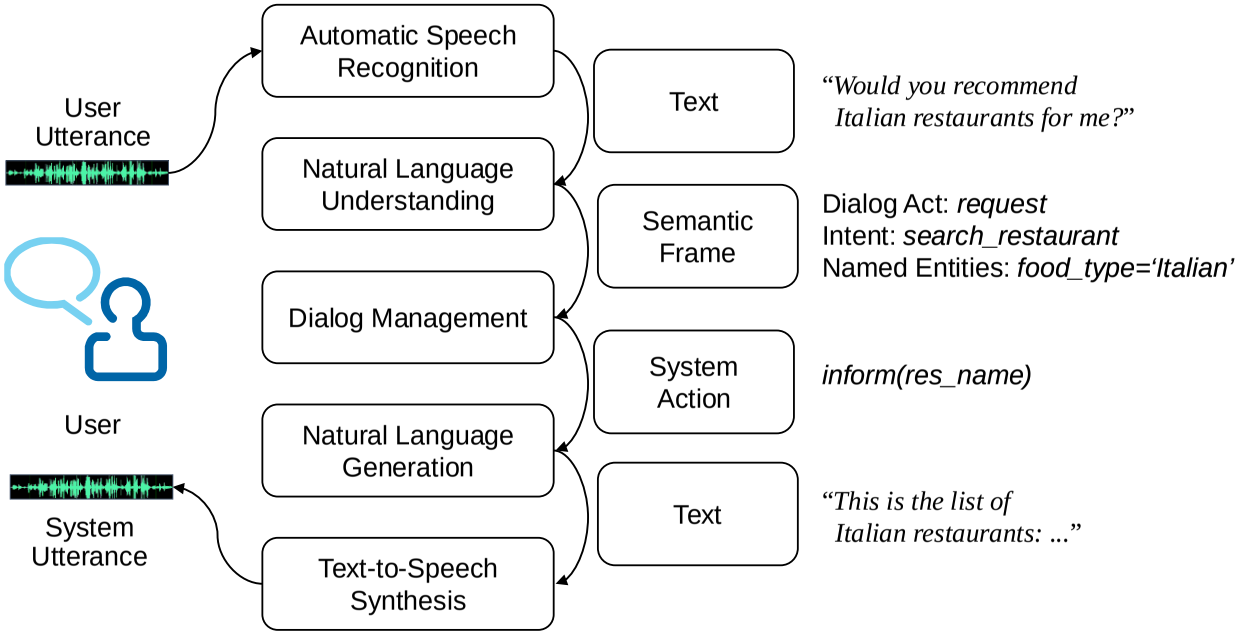


Figure 1.1: Architecture of dialog systems.

1.1 Task Definition

The main goal of this dissertation is to develop an accurate *OOD sentence detection* method. We define OOD sentence detection as a *binary classification* problem of determining whether the system can respond appropriately to an input sentence, i.e.,

$$f(x) = \begin{cases} \textit{In–domain}, & \text{if } x \text{ belongs to a domain } d \in D, \\ \textit{OOD}, & \text{otherwise,} \end{cases} \tag{1.1}$$

where x is an input sentence, D is a closed set of target–domain categories such as `schedule` or `message`.

Most previous studies [8, 9] use both in–domain sentences and OOD sentences to train OOD sentence detection. Although collecting in–domain sentences is a necessary step in building many data–driven dialog systems, collecting enough OOD sentences that cover all other domains is laborious and time–consuming. Therefore, the goal of this dissertation is to develop an accurate OOD sentence detection method that requires only in–domain sentences for training.

1.2 Motivation

In this work, we present a novel *sentence–embedding* method that represents sentences in a low–dimensional continuous vector space that emphasizes aspects that distinguish in–domain cases from OOD cases.

First, we use large set of unlabeled text to pre–train *distributed word representations* for the initialization of neural sentence–embedding. These representations can capture word meaning even for unknown or rare words in our training data.

Second, to train neural sentence embedding with only in–domain sentences, we use the similarity between two tasks: OOD sentence detection and *domain–*

category analysis [8, 10, 11, 12]. Domain–category analysis is a task that assigns one of a closed set of target domains to a given sentence; this analysis system can be trained using only in–domain sentences that are collected domain–by–domain. We think the task of OOD sentence detection is more similar to domain–category analysis than to other tasks such as sentiment analysis, so we expect the features (i.e., representation) of a sentence extracted by a domain–category analysis system to work for OOD sentence detection too. Therefore we adopt a feature extractor trained for domain–category analysis, and use it as a sentence embedding system for OOD sentence detection.

Afterwards, the learned representations of in–domain sentences to train *one–class classifiers* that distinguish in–domain sentences from OOD sentences. We propose two generative approaches for one–class classification: *autoencoders* and *generative adversarial networks*.

An autoencoder trained from in–domain sentences will have low reconstruction errors for in–domain sentences, so an input sentence can be classified into either in–domain or OOD based on its reconstruction error by the autoencoder.

A generative adversarial network [13] consists of a generator G and a discriminator D. We train D that distinguishes the in–domain sentences from the fake sentences generated by G, so we expect D to reject OOD sentences.

To the best of our knowledge, this is the first study that applies deep learning to solve the sentence representation and one–class classification problems of OOD sentence detection.

1.3 Data set

To demonstrate the effectiveness of our proposed method, we experimented on a data set of 6,268 Korean sentences. We used a Wizard–of–Oz approach to collect the data set with several groups of people from industries in Korea including LG Electronics and Mediazen. The data set consists of 706 OOD sentences about three domains (Table 1.1) and 5,562 in–domain sentences about 14 domains (Table 1.2). Eighty percent of the in–domain sentences were used to train the models; the remaining in–domain sentences and all OOD sentences were used for testing. Although we used Korean sentences to implement our method, it does not rely on language–specific processes except for word tokenization, and can therefore be applied to other languages.

Table 1.1: Out–of–domain data set.

Domain (size)	Example Sentences
Hotel (109)	*"Show me my hotel reservations"* *"Reserve the Hilton Hotel at Seoul."*
Message (95)	*"Please send this message to my dad."* *"Show the messages from Jun-soo."*
Smalltalk (502)	*"I don't play with you any more."* *"I heard my friend has divorced."*

Table 1.2: In–domain data set.

Domain (size)	Example Sentences
Airplane (113)	*"Show me my flight reservations."* *"Reserve two economy tickets to Seoul."*
Alarm (372)	*"Please wake me up after three hours."* *"Delete alarms at 6 o'clock."*
Bus (115)	*"Would you show me the route map of bus no. 505?"* *"Where are the bust stops near the POSCO office?"*
Call (181)	*"Call to my husband."* *"Show me this month's call history."*
Car Navigation (380)	*"Please guide me to the SK Gas Station."* *"Is parking available at Suwon Station?"*
Diet Talk (375)	*"I've drunk two bottles of water today."* *"What shall I do for dinner?"*
Exchange (107)	*"What's the exchange rate for KRW to USD?"* *"How much is $10,000 in KRW?"*
General (857)	*"Ok. Please do that."* *"I said no."*
Schedule (381)	*"Please set up a meeting at 3 pm."* *"Show me tomorrow's schedule."*
Songfinder (214)	*"Please play If I Leave"* *"Who sang Things That I Can Not Do For You?"*
Time (96)	*"What time is it now?"* *"What's the time difference with Hong Kong?"*
Train (121)	*"Check my train reservation."* *"Reservation a train from Seoul to Busan."*
TV (1606)	*"Please delete the recorded Infinite Challenge."* *"Who starred in today's Golden Fishery?"*

1.4 Evaluation Metrics

We use *equal error rate* (EER) to represent the accuracy of OOD sentence detection [14]. EER is the error rate at which false acceptance rate

$$\text{FAR} = \frac{\text{Number of } \textit{accepted} \text{ OOD sentences}}{\text{Number of OOD sentences}} \tag{1.2}$$

and false rejection rate

$$\text{FRR} = \frac{\text{Number of } \textit{rejected} \text{ ID sentences}}{\text{Number of ID sentences}} \tag{1.3}$$

are equal (Fig. 1.2).

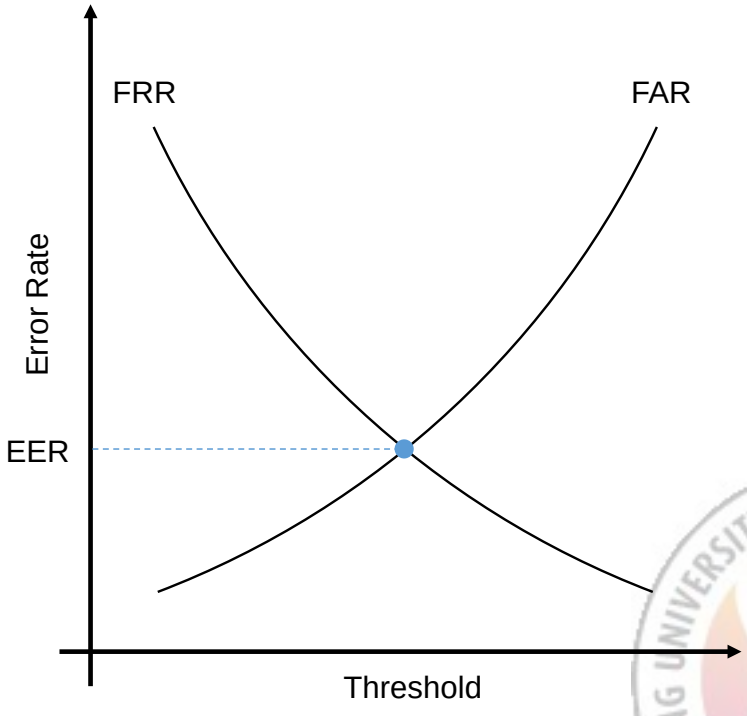


Figure 1.2: Concept of equal error rate.

1.5 Related Work

Previous studies [8, 9, 14] on OOD sentence detection use sentence representations based on bag–of–words models, which have limitations in representing rare or unknown words that are likely to appear in OOD sentences.

An *in–domain verification* method [14] uses only in–domain sentences to build domain–wise one–vs.–rest classifiers expected to generate high confidence scores for target–domain sentences, and then uses a low score as evidence that a sentence is OOD. However, many OOD sentences had high confidence scores in our data set (Fig. 1.3) because OOD sentences were not among the negative examples of the classifiers. Therefore, those confidence scores are not sufficiently reliable evidences of OOD.

An two–stage domain selection framework [8] uses both in–domain sentences and OOD sentences to build multi–domain dialog systems. The main contribution is a way to exploit discourse information to prevent erroneous domain switching, but whenever developers expand the domain of a dialog system they must reassess all OOD sentences because some will become in–domain due to the change of the boundary between in–domain and OOD.

Syntactic features and semantic features can be used to identify OOD sentences [9]. Web search queries are used as OOD sentences to eliminate the need to collect OOD sentences, but such queries are *noisy* because some are actually in–domain, and they cannot be obtained readily without using a commercial search engine.

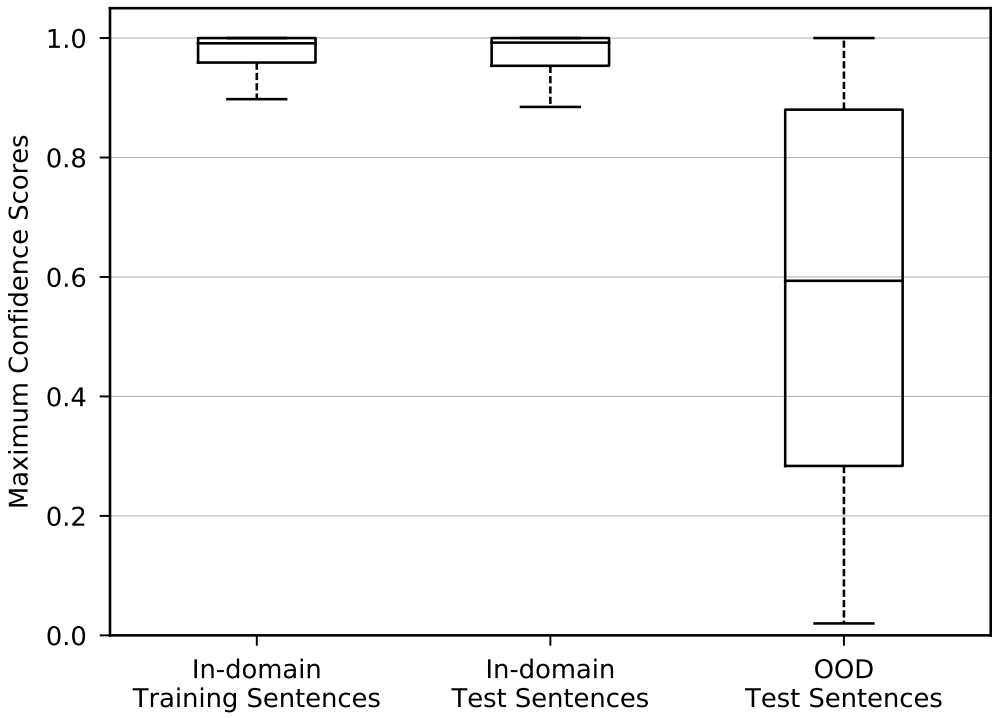


Figure 1.3: Maximum scores of domain–wise binary classifiers.

1.6 Dissertation Organization

The remainder of this dissertation is organized as follows: In Section II, we describe our sentence–embedding methods for out–of–domain sentence detection. In Section III, we explain our one–class classification methods for out–of–domain sentence detection. In Section IV, we conclude this dissertation.

II. Sentence Embedding for Out–of–domain Sentence Detection

Sentence embedding aims at representing input sentences in an m–dimensional continuous vector space. To process variable–length sentences, we use recurrent neural networks (RNNs) that repeatedly computes given operations for every word in a sentence (Fig. 2.1). In RNNs, the t^{th} hidden layer $\mathbf{h}^{(t)}$ is defined as

$$\mathbf{h}^{(t)} = \sigma_h(\mathbf{W}_h[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_h), \tag{2.1}$$

where $\mathbf{x}^{(t-1)}$ is t^{th} input to the RNN, $\mathbf{W}_h$ is a weight matrix, $\mathbf{b}_h$ is a bias vector, and

$$\sigma_h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in (-1, 1). \tag{2.2}$$

is a hyperbolic tangent function. The t^{th} output layer $\mathbf{y}^{(t)}$ is computed from the t^{th} hidden layer $\mathbf{h}^{(t)}$.

According to a target task, an RNN is designed to produce either multiple outputs as in Fig. 2.1 (many–to–many RNN) or only one output (many–to–one RNN). Many–to–many RNNs are used in various natural language processing tasks such as language modeling [15], machine translation [16, 17], chatbots [18], or dialog systems [19]. Many–to–one RNNs are mainly used for text classification [20, 21, 22]. We will use both the many–to–one and the many–to–many architectures for sentence embedding.

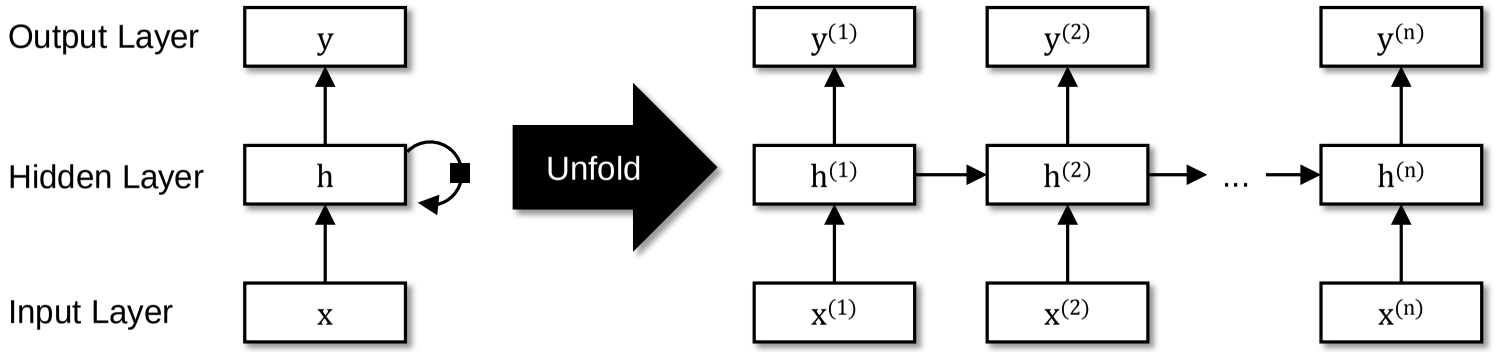


Figure 2.1: Recurrent neural network.

RNNs are trained using a backpropagation through time (BPTT) technique [23] that uses the previous time steps additionally to compute the gradient of the current time step. However, vanilla RNNs with BPTT have a vanishing gradient problem [24]: the gradient of a time step almost vanishes after few time steps, so those RNNs cannot understand the dependencies between far away words. To solve this problem, a gated recurrent unit (GRU) cell [25] or long short–term memory (LSTM) cell [26, 27] is applied to RNNs.

In the t^{th} GRU cell (Fig. 2.2), reset gate $\mathbf{r}^{(t)}$, update $\mathbf{z}^{(t)}$, and hidden state $\mathbf{h}^{(t)}$ are defined as

$$\mathbf{r}^{(t)} = \sigma_g(\mathbf{W}_r[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_r), \tag{2.3}$$

$$\mathbf{z}^{(t)} = \sigma_g(\mathbf{W}_z[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_z), \tag{2.4}$$

$$\tilde{\mathbf{h}}^{(t)} = \sigma_h(\mathbf{W}_h[\mathbf{r}^{(t)} \otimes \mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_h), \tag{2.5}$$

$$\mathbf{h}^{(t)} = (1 - \mathbf{z}^{(t)}) \otimes \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \otimes \tilde{\mathbf{h}}^{(t)}, \tag{2.6}$$

where $\otimes$ denotes an element–wise product, $\mathbf{W}_r$, $\mathbf{W}_z$, and $\mathbf{W}_h$ are weight matrices, $\mathbf{b}_r$, $\mathbf{b}_z$, and $\mathbf{b}_h$ are bias vectors, and

$$\sigma_g(x) = \frac{1}{1 + e^{-x}} \in (0, 1). \tag{2.7}$$

is a sigmoid function. Reset gate $\mathbf{r}^{(t)}$ indicates the element–wise proportions of preserved information in $\mathbf{h}^{(t-1)}$. Update gate $\mathbf{z}^{(t)}$ indicates the element–wise weights between $\tilde{\mathbf{h}}^{(t)}$ and $\mathbf{h}^{(t-1)}$.

In the t^{th} LSTM cell (Fig. 2.3), input gate $\mathbf{i}^{(t)}$, forget gate $\mathbf{f}^{(t)}$, memory cell

state $\mathbf{c}^{(t)}$, output gate $\mathbf{o}^{(t)}$, and hidden state $\mathbf{h}^{(t)}$ are defined as

$$\mathbf{i}^{(t)} = \sigma_s(\mathbf{W}_i[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_i), \tag{2.8}$$

$$\mathbf{f}^{(t)} = \sigma_s(\mathbf{W}_f[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_f), \tag{2.9}$$

$$\tilde{\mathbf{C}}^{(t)} = \sigma_h(\mathbf{W}_C[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_C), \tag{2.10}$$

$$\mathbf{c}^{(t)} = \mathbf{i}^{(t)} \otimes \tilde{\mathbf{C}}^{(t)} + \mathbf{f}^{(t)} \otimes \mathbf{c}^{(t-1)}, \tag{2.11}$$

$$\mathbf{o}^{(t)} = \sigma_s(\mathbf{W}_o[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_o), \tag{2.12}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \otimes \sigma_h(\mathbf{c}^{(t)}), \tag{2.13}$$

where $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_C$, and $\mathbf{W}_o$ are weight matrices, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_C$, and $\mathbf{b}_o$ are bias vectors. Forget gate $\mathbf{f}^{(t)}$ indicates the element-wise proportions of preserved information in previous memory cell $\mathbf{c}^{(t-1)}$. Input gate $\mathbf{i}^{(t)}$ indicates the element-wise proportions of preserved information in $\tilde{\mathbf{C}}^{(t)}$. Output gate $\mathbf{o}^{(t)}$ indicates the element-wise proportions of exposed information in $\mathbf{c}^{(t)}$.

We will assess vanilla RNNs, GRU networks, and LSTM networks for sentence embedding.

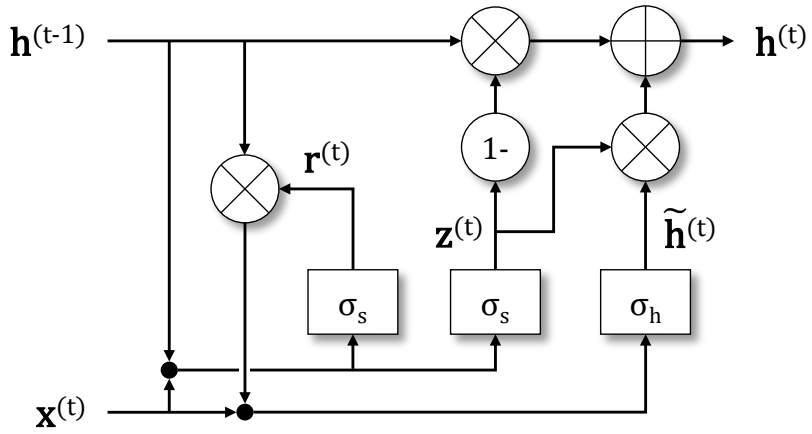


Figure 2.2: Gated recurrent cell.

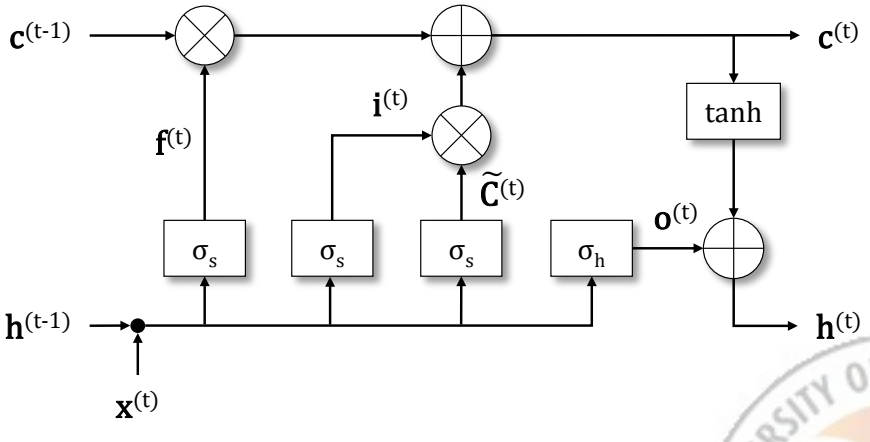


Figure 2.3: Long short–term memory cell.

2.1 Pre–training of Word Embedding

Before training *sentence* embedding, *words* must be represented in a low–dimensional continuous vector space in which semantically–similar words are located near each other. For example, '*London*' should be closer to '*Paris*' than to '*apple*' in the vector space. The pre–trained word representations would be *fine–tuned* when sentence embedding is learned using in–domain sentences (Sections 2.2 and 2.3). However, the amount of unlabeled text such as Wikipedia articles is bigger than the amount of in–domain data set, so the pre–training can increase both the accuracy and coverage of the word representations.

2.1.1 Methods

We utilize the distributional hypothesis of words: the meanings of words can be inferred from their accompanying context [28, 29]. We first extract a large set of unlabeled text from Korean Wikipedia articles, then use it to train a skip–gram neural network [30] that predicts surrounding words. Our skip–gram neural network (Fig. 2.4) predicts 10 surrounding words by using a v–dimensional hidden layer; we set v as 100. When the training set consists of k unique words in its vocabulary, the result of pre–training is a matrix $\mathbf{E} \in \mathbb{R}^{v \times k}$ in which the i^{th} column is a v–dimensional vector that represents the i^{th} word.

We use Gensim word2vec library [31] to implement the skip–gram neural network, and apply negative sampling technique [32] to the network for efficient learning. We chose initial learning rate 0.05 and decreased it linearly.

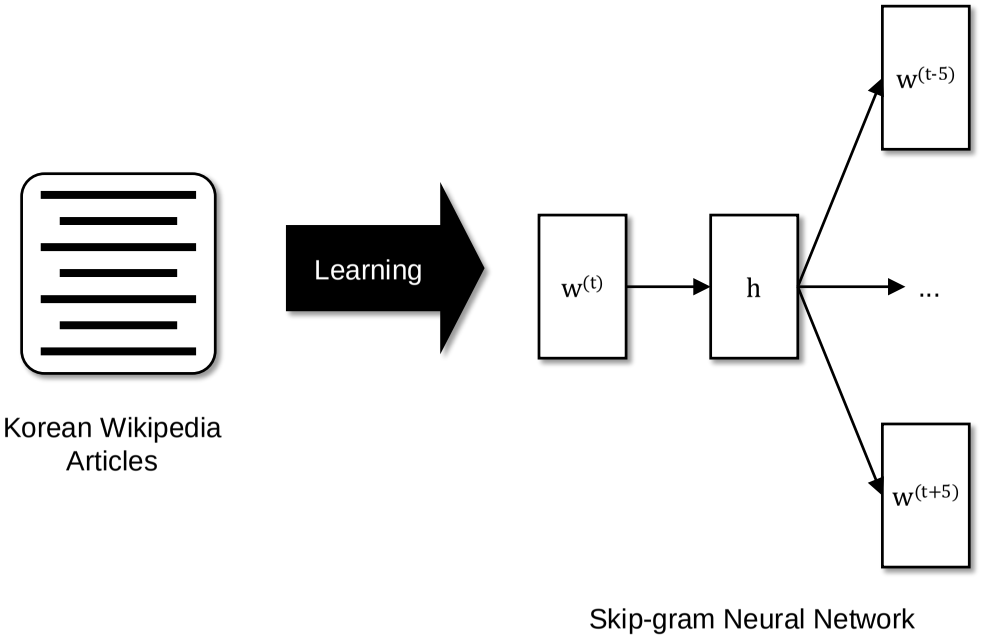


Figure 2.4: Learning process of a skip–gram neural network.

2.1.2 Results and Discussion

We used WikiExtractor[1] to extract plain text from Korean Wikipedia database archived on March 20, 2017[2]. The normalized plain text contains about 93 million words (101,091 unique words).

We set the dimension v of distributed word representation to 100, so each word is represented by a 100–dimensional vector. When we examine the 100–dimensional vectors of nine Korean words, their meaning is difficult to identify using the human eye (Fig 2.5).

Therefore, we use t–distributed Stochastic Neighbor Embedding (t–SNE) [33] implemented in the scikit–learn library [34] to reduce the dimension of word vectors to two. "t–SNE minimizes the divergence between two distributions: a distribution that measure pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low–dimensional points in the embedding" [35]; i.e., after embedding using t–SNE similar items in the original space should be similar also in the low–dimensional space.

We chose 77 Korean words, and plotted them into a two–dimensional vector space (Fig 2.6). The result shows that semantically similar words tend to be located by each other in the vector space, so we will use these pre–trained word vectors to initialize sentence embedding (Sections 2.2 and 2.3).

[1]https://github.com/attardi/wikiextractor
[2]https://dumps.wikimedia.org/kowiki/

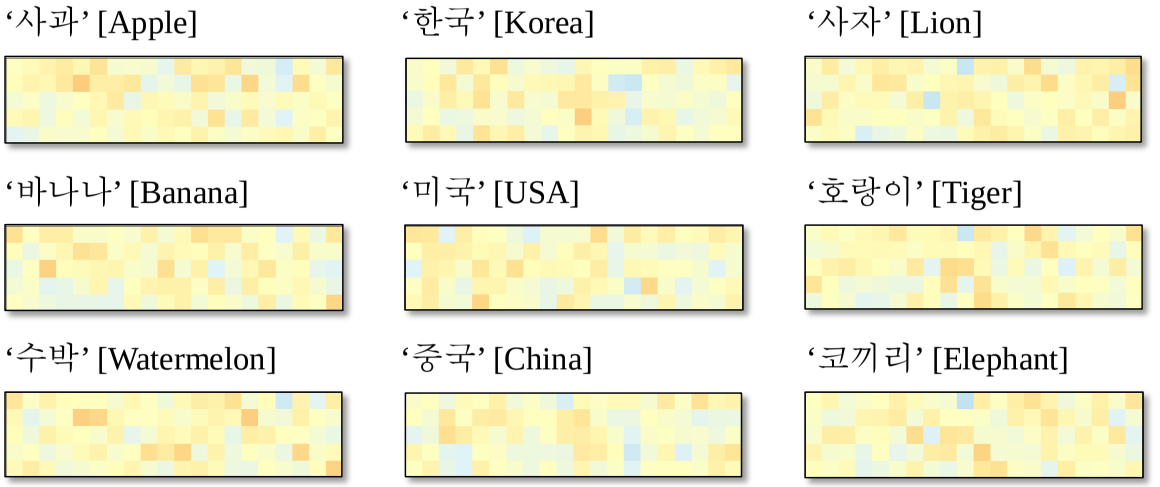


Figure 2.5: 100–dimensional word vectors of the 9 Korean words.

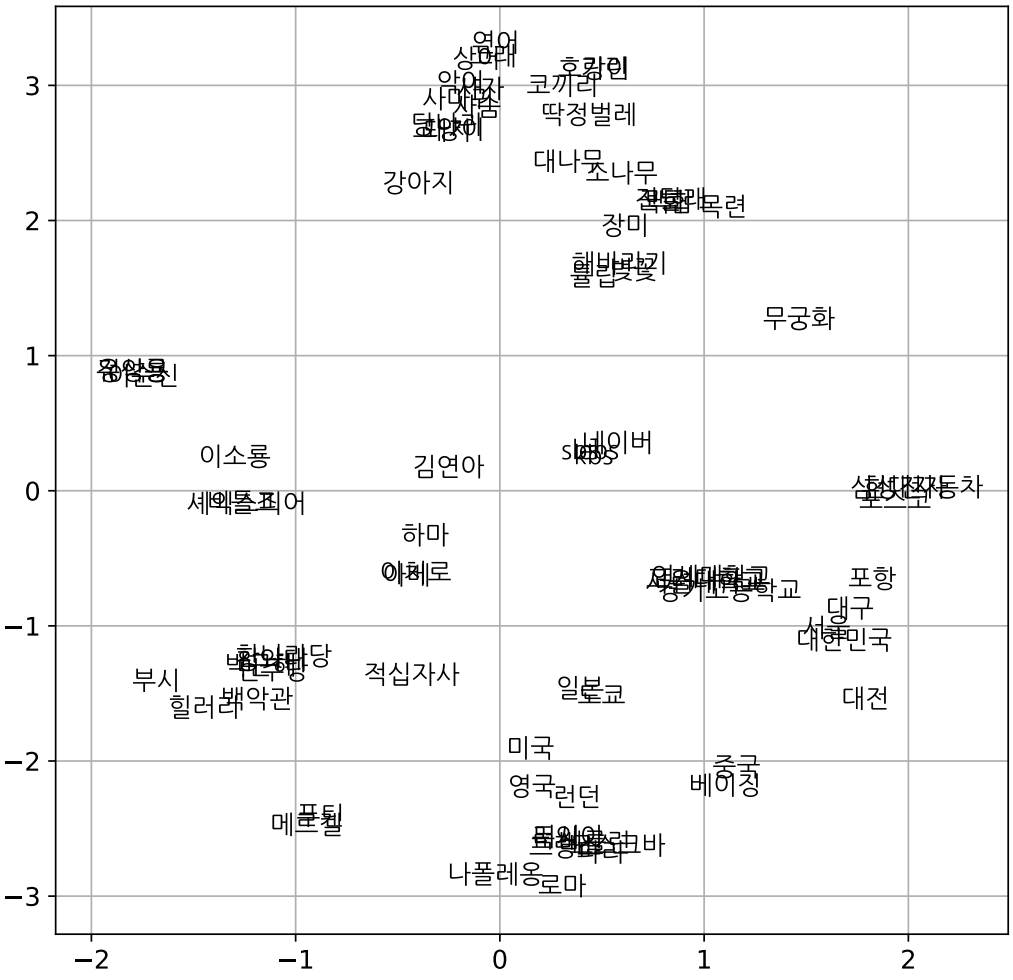


Figure 2.6: 77 Korean words in the two–dimensional vector space.

2.2 Sentence Embedding by Recurrent Neural Network for Domain–category Analysis

We obtained the word representations in Section 2.1. Neural bag–of–words [36] is a simple sentence–embedding method that computes a sentence representation as the element–wise average of its word representations; so word–order information is not captured.

Some researchers have used RNNs for sentence embedding that considers word orders [2, 37, 38, 39, 40, 41]. However, because we cannot define an objective function based on classification error between in–domain cases and OOD cases, these methods are not directly applicable to our problem in which only in–domain sentences are available as a training set. To solve this problem, we exploit the similarity between OOD sentence detection and domain–category analysis.

We propose a new sentence–embedding method **DCA–RNN**, which uses domain–category analysis as the supervised objective of the RNN. Because the extracted features of DCA–RNN are also important in OOD sentence detection, we expect the last hidden layer of the DCA–RNN to represent the input sentence suitable for OOD sentence detection. This is a sort of transfer–learning approach [42] that learns knowledge from an *auxiliary task* (i.e., domain–category analysis), then applies the knowledge to a *target task* (i.e., out–of–domain sentence detection).

2.2.1 Methods

We train our RNN (Fig. 2.7) to predict the domain–category y of an input sentence $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)})$. Let $\mathbf{x}^{(t)} \in \mathbb{N}^k$ be the one–hot vector representation of the t^{th} word in a given sentence, where k unique words are in the vocabulary. The dense vector representation $\mathbf{w}^{(t)} \in \mathbb{R}^{2v}$ of the t^{th} word is computed from $\mathbf{x}^{(t)}$, where v is the dimension of the pre–trained word representations (Section 2.1).

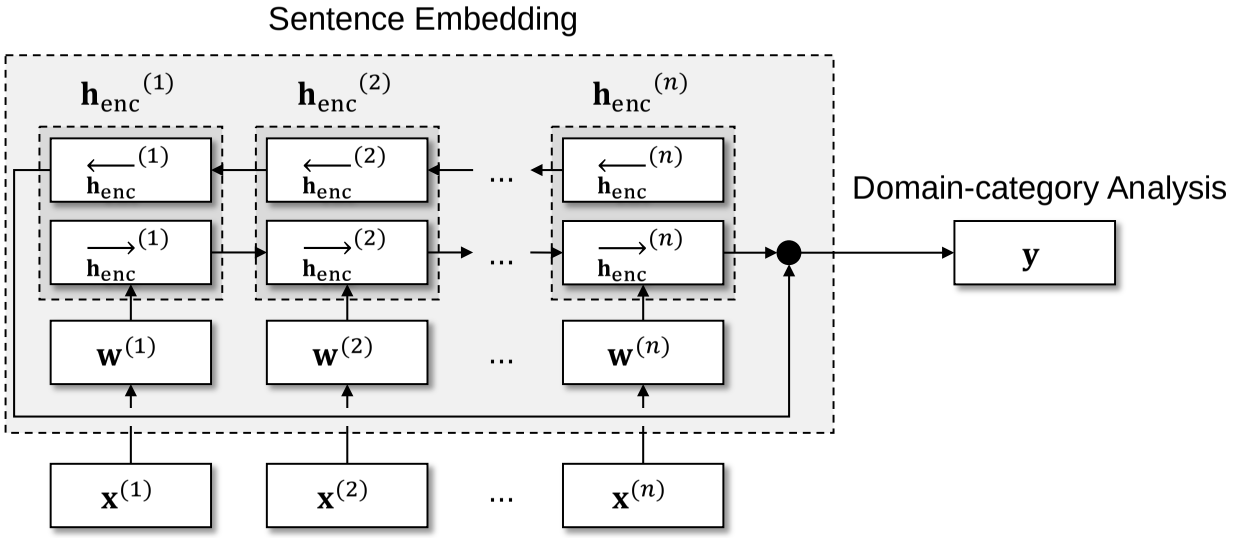


Figure 2.7: Recurrent neural network for domain–category analysis.

The pre–trained word representations are fine–tuned based on back–propagation from the objective function of the RNN; fine–tuning finds *task–specific* word representations, whereas pre–trained word representations describe *general* word meaning. However, some words in OOD sentences appear rarely or never in in–domain sentences; because of this imbalance, the word representations cannot be fined–tuned accurately.

To prevent this problem, we use a *two–channel* approach: a *non–static* channel is fine–tuned during training, whereas a *static* channel is fixed. This multi-channel idea has been used earlier for sentiment analysis [43] but not for OOD sentence detection. In addition, we apply *dropout* [44] to the non–recurrent layers in our RNN. Dropout is a regularization technique that randomly drops some nodes in neural networks during training. Especially, dropout prevents our RNN from becoming biased toward the non–static channel.

So we define $\mathbf{w}^{(t)} \in \mathbb{R}^{2v}$ of the t^{th} word is defined as

$$\mathbf{w}^{(t)} = [\mathbf{E}_s \mathbf{x}^{(t)}, \mathbf{E}_n \mathbf{x}^{(t)}], \tag{2.14}$$

where $\mathbf{E}_s \in \mathbb{R}^{v \times k}$ and $\mathbf{E}_n \in \mathbb{R}^{v \times k}$ are the weight matrices for the static channel and the non–static channel, respectively; both $\mathbf{E}_s$ and $\mathbf{E}_n$ are initialized to $\mathbf{E}$ pre–trained word representations in Section 2.1, but only $\mathbf{E}_n$ is fine–tuned during the training; a dropout rate of 50% is applied to both $\mathbf{E}_s \mathbf{w}^{(t)}$ and $\mathbf{E}_n \mathbf{w}^{(t)}$.

We use bidirectional structure [45, 46] of the RNN to prevent it from becoming biased toward the last few words, so those weights are defined independently for the forward encoder ($\overrightarrow{\mathbf{h}_{\text{enc}}}^{(1)}, \overrightarrow{\mathbf{h}_{\text{enc}}}^{(2)}, \cdots, \overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}$) and the backward encoder ($\overleftarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(n-1)}, \cdots, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}$). So t^{th} hidden states $\overrightarrow{\mathbf{h}_{\text{enc}}}^{(t)}$ and $\overleftarrow{\mathbf{h}_{\text{enc}}}^{(t)}$ of the encoder are computed by recurrent operations from the distributed word representations $(\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \cdots, \mathbf{w}^{(t)})$ and $(\mathbf{w}^{(n)}, \mathbf{w}^{(n-1)}, \cdots, \mathbf{w}^{(t)})$ respectively. We assess the three RNN types: vanilla RNN, GRU network, and LSTM network. We expect GRU and LSTM to prevent the vanishing gradient problem of the

RNN.

The output domain–category layer $\mathbf{y} \in \mathbb{R}^{|D|}$ is computed based on $[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}]$, which is the concatenation of the last hidden layers of both the forward encoder and the backward encoder, so that

$$\mathbf{y} = \text{softmax}(\mathbf{W}_y[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}] + \mathbf{b}_y), \tag{2.15}$$

where $\mathbf{W}_y$ is a weight matrix and $\mathbf{b}_y$ is a bias vector; a dropout rate of 50% is applied to $[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}]$; softmax is an activation function that normalizes an input vector to a discrete probabilistic distribution:

$$\text{softmax}(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K. \tag{2.16}$$

The RNN is trained to minimize the categorical cross entropy between the gold–standard domain category y and the predicted output y'; categorical cross entropy between two probability distributions p and q is

$$H(p, q) = -\sum_{x} p(x) \log q(x). \tag{2.17}$$

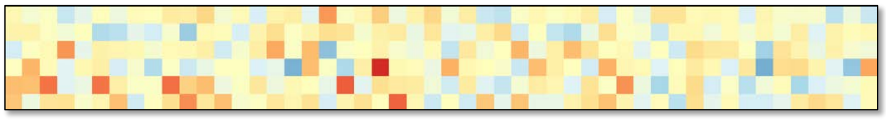
As a result, given a sentence, $[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}]$ of the trained network represents the sentence in a vector space that emphasizes the distinguishing aspects among domain categories.

To implement our DCA–RNN, we use the Tensorflow library [47]. We train our models by using Adam [48] optimizer with a mini–batch size of 128 and an initial learning rate of 0.001 that is decreased linearly during 200 epochs. All weights except pre–trained word representations are initialized from a zero–centered Normal distribution with standard deviation 0.1.

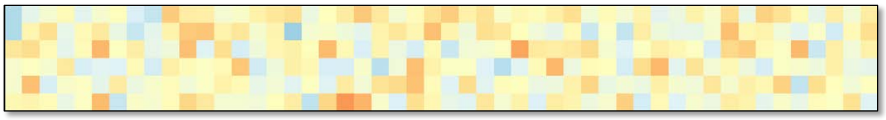
2.2.2 Results and Discussion

We computed the distributed representations of whole sentences in our data set (Tables 1.1, 1.2) by DCA–RNN. The dimension of the hidden layers in the

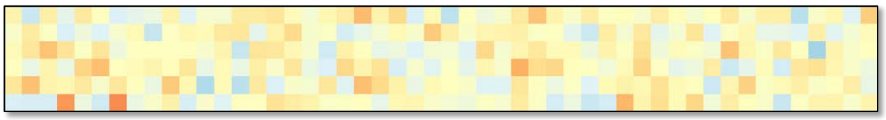
① “*When was Good Day released?*” (Songfinder)



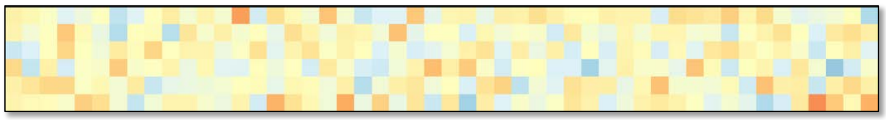
② “*I have an appointment November 20 at 4 pm.*” (Schedule)



③ “*What exercise should I do in the evening?*” (Diet Talk)



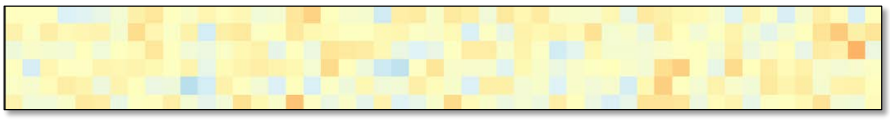
④ “*Please delete the recorded Infinite Challenge.*” (TV)



(a) In–domain sentences

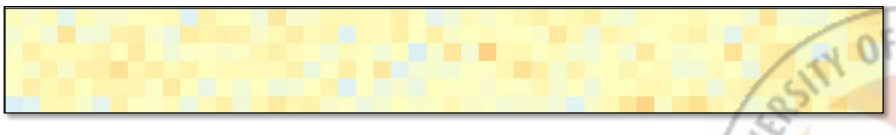
① “*Please send this message to my dad.*” (Message)

② “*Show me my hotel reservations.*” (Hotel)



③ “*I don’t play with you any more.*” (Smalltalk)

④ “*I heard my friend has divorced.*” (Smalltalk)



(b) Out–of–domain sentences

Figure 2.8: Distributed representations of eight sentences.

RNNs is set to 150, so sentence representations are 300–dimensional vectors. When the distributed representations of four in–domain sentences (Fig. 2.8a) and four OOD sentences (Fig. 2.8b) are reshaped to 5×60 matrices for visualization, the human eye cannot easily understand their meaning. Nevertheless, we use these sentence representations in one–class classification for OOD sentence detection (Section III).

Our ultimate goal is to detect OOD sentences, but we also evaluated domain–category analysis in this section. We used 80% of in–domain sentences in our data set (Table 1.2) to train our models, and used the remaining 20% of in–domain sentences for the test. We used accuracy as the evaluation measure of domain–category analysis.

We compared our method to support vector machines (SVMs) [49]; SVMs are linear classifiers that are trained by finding hyperplanes that produces the largest margin between different classes. We used the C–support vector classification implementation in the scikit–learn library [34], and applied three types of kernels: linear, polynomial, and radial basis function (RBF).

We applied one of three sentence representation methods to SVMs.

- **Bag–of–words** represents a sentence as a vector in which the i^{th} element is the frequency of the i^{th} keyword in the sentence. We use n–gram by increasing n from 1 to 3 to capture local word order; only the result with the best n is presented.
- **TF–IDF**. A sentence is represented as a vector in which the i^{th} element is the product of the term frequency (TF) and the inverted document frequency (IDF) of the i^{th} keyword in the sentence. We use n–gram as in bag–of–words.
- **Neural bag–of–words** [36]. A sentence is represented as the element–wise average of its word representations obtained in Section 2.1.

Table 2.1: Accuracies [%] of domain–category analysis based on support vector machines.

Feature	Best Kernel	Accuracy
Bag–of–words (1–gram)	Linear	93.45
TF–IDF (2–gram)	RBF	94.44
Neural bag–of–words	RBF	97.02

SVMs have penalty parameters for the error term and kernel coefficient. To find the best hyper–parameter setting, we use 80% of the training data as temporal training data and the remaining part as temporal validation data.

The combination of neural bag–of–words features and polynomial kernel was the most accurate (97.02%) in the domain–category analysis experiment based on SVMs (Table 2.1). The best result without neural bag–of–words feature (94.44%) was obtained by TF–IDF 2–gram features and RBF kernel. The results show that pre–trained word embedding (Section 2.1) is useful for domain–category analysis although word order information is not captured.

For domain–category analysis based on DCA–RNN, we compared four variations of word embedding: random, static, non–static, and two–channel; the first method initializes word embedding randomly; the others were described in Section 2.2.1. We performed each experiment 20 times, and recorded the average accuracy of domain–category analysis.

In the domain–category analysis experiment based on RNNs (Table 2.2), two–channel word embedding (99.07% with LSTM) was most accurate. Fine–tuning the pre–trained word representations (i.e., non–static) was more accurate than using the pre–trained word representations without fine–tuning (i.e., static), but they cooperated complementary when they were used simultaneously.

In contrast, the results obtained using the RNN (99.07%), the GRU (99.01%) network, and the LSTM network (99.07%) did not differ significantly (Table 2.2).

Table 2.2: Accuracies [%] ± s.d. (n = 20) of domain–category analysis based on recurrent neural networks.

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	96.10 ± 0.51	94.45 ± 0.35	98.83 ± 0.32	99.07 ± 0.29
GRU	98.06 ± 0.46	97.81 ± 0.29	98.93 ± 0.22	99.01 ± 0.22
LSTM	97.83 ± 0.33	97.47 ± 0.39	98.86 ± 0.20	99.07 ± 0.22

The advantage of GRU and LSTM cells is that they capture long–term dependency in sentences. However, voice commands to virtual assistants are usually short (Fig. 2.9); the average number of words in in–domain sentences was 5.16 (s.d. 3.02, n = 5562), so GRU networks and LSTM networks could not demonstrate their advantage.

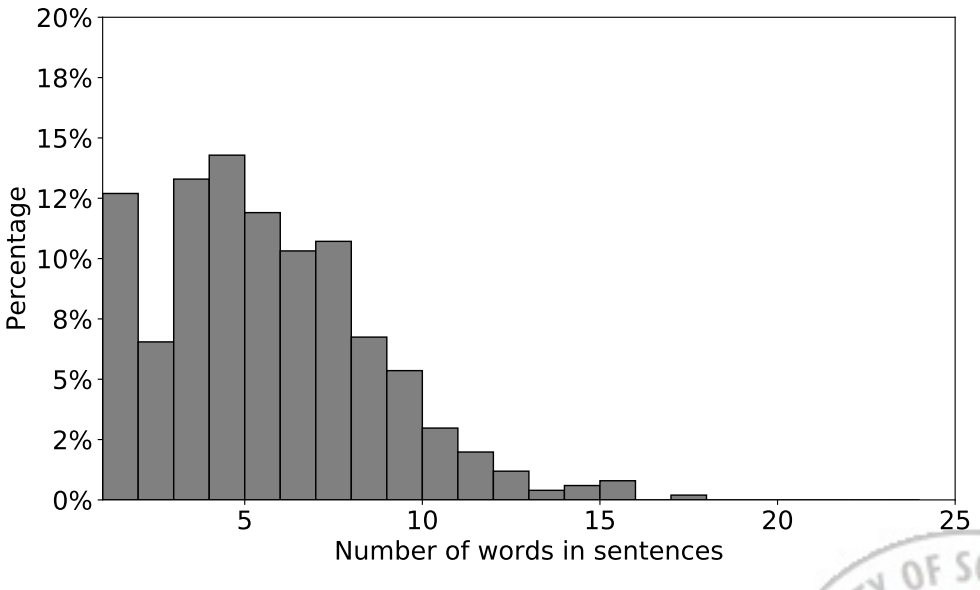


Figure 2.9: Histogram of number of words in in–domain sentences.

2.3 Sentence Embedding by Recurrent Neural Network for Sequence–to–sequence

The feature extractors trained for domain–category analysis focuses only on features about domain–categories, but overall sentence meaning is also important. Therefore, we assess another sentence–embedding method **S2S–RNN**, which uses sequence–to–sequence learning as the supervised objective of the RNN.

Sequence–to–sequence [17, 18, 19] is a task that generates a sentence from an input sentence. We expect the feature extractor of S2S–RNN to focus on overall sentence meaning. A similar approach [50] has been used to learn the representation of videos (i.e., the sequences of images) in an unsupervised manner. We also assess **Joint–RNN**, which uses both domain–category analysis and sequence–to–sequence as the supervised objectives.

2.3.1 Methods

S2S–RNN consists of an *encoder* and a *decoder* for sequence–to–sequence: the encoder converts an input sentence to a distributed representation; the decoder converts the distributed representation to the original input sentence. We can use the encoder of the S2S–RNN as a sentence embedding system for OOD sentence detection. The structure of the S2S–RNN (Fig. 2.10) is similar to the DCA–RNN, so the details of the RNN are in Section 2.2.

Encoder

The encoder gets an input sentence $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)})$, where $\mathbf{x}^{(t)} \in \mathbb{N}^k$ is the one–hot vector representation of the t^{th} word in a given sentence, and k unique words are in the vocabulary. The dense vector representation $\mathbf{w}^{(t)} \in \mathbb{R}^{2v}$ of the t^{th} word is computed from $\mathbf{x}^{(t)}$ as in Section 2.2, where v is the dimension of the pre–trained word representations (Section 2.1).

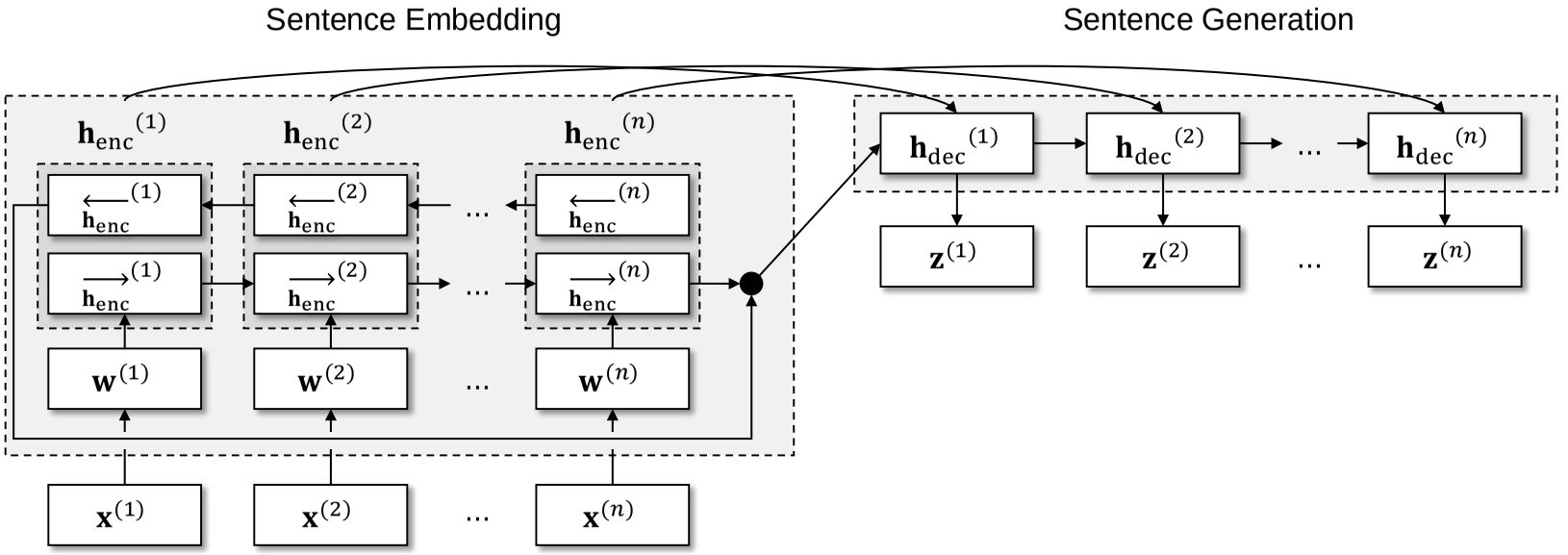


Figure 2.10: Recurrent neural network for sequence–to–sequence.

We use bidirectional structure [45, 46] of the RNN to prevent it from becoming biased toward the last few words, so those weights are defined independently for the forward encoder ($\overrightarrow{\mathbf{h}_{\text{enc}}}^{(1)}, \overrightarrow{\mathbf{h}_{\text{enc}}}^{(2)}, \cdots, \overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}$) and the backward encoder ($\overleftarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(n-1)}, \cdots, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}$). So t^{th} hidden states $\overrightarrow{\mathbf{h}_{\text{enc}}}^{(t)}$ and $\overleftarrow{\mathbf{h}_{\text{enc}}}^{(t)}$ of the encoder are computed by recurrent operations from the distributed word representations ($\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \cdots, \mathbf{w}^{(t)}$) and ($\mathbf{w}^{(n)}, \mathbf{w}^{(n-1)}, \cdots, \mathbf{w}^{(t)}$) respectively; a dropout rate of 50% is applied to each hidden state of the encoder. We assess the three RNN types as in Section 2.2.

Decoder

We set the initial state of the decoder to $[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(t)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}]$. We apply forward only structure to the decoder instead of bidirectional structure; the t^{th} hidden state ${\mathbf{h}_{\text{dec}}}^{(t)}$ of the decoder is computed by recurrent operations from the hidden states of the encoder ($[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(1)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}], [\overrightarrow{\mathbf{h}_{\text{enc}}}^{(2)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(2)}], \cdots, [\overrightarrow{\mathbf{h}_{\text{enc}}}^{(t)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(t)}]$). We assess the three RNN types as in Section 2.2.

The output sentence ($\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(n)}$) of the decoder is same to the input sentence ($\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}$). The t^{th} word $\mathbf{z}^{(t)}$, the one–hot vector representation of the t^{th} word in a given sentence, is computed based on ${\mathbf{h}_{\text{dec}}}^{(t)}$, so that

$$\mathbf{z}^{(t)} = \text{softmax}(\mathbf{W}_z {\mathbf{h}_{\text{dec}}}^{(t)} + \mathbf{b}_z), \tag{2.18}$$

where $\mathbf{W}_z$ is a weight matrix and $\mathbf{b}_z$ is a bias vector.

The RNN is trained to minimize the average of categorical cross entropy (Eq. 2.17) between the actual output word $\mathbf{z}^{(t)}$ and the predicted output word $\mathbf{z}'^{(t)}$:

$$\frac{1}{n} \sum_{t=1}^{n} H(\mathbf{z}^{(t)}, \mathbf{z}'^{(t)}). \tag{2.19}$$

As a result, given a sentence, $[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}]$ of the trained network represents the sentence in a vector space that emphasizes the distinguishing aspects

Table 2.3: Accuracies [%] ± s.d. (n = 20) of domain–category analysis based on recurrent neural networks in which sequence–to–sequence is also used.

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	96.36 ± 0.63	94.48 ± 0.28	98.81 ± 0.30	98.72 ± 0.35
GRU	98.16 ± 0.37	97.79 ± 0.33	98.72 ± 0.26	98.64 ± 0.30
LSTM	97.90 ± 0.45	97.51 ± 0.27	98.59 ± 0.25	98.79 ± 0.27

among domain categories.

Joint Task

Joint–RNN, which is joint learning of both domain–category analysis and sequence–to–sequence, is implemented by adding domain–category y to S2S–RNN. The domain-category y is computed from the last hidden states of the encoder as

$$\mathbf{y} = \text{softmax}(\mathbf{W}_y[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(n)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}] + \mathbf{b}_y). \tag{2.20}$$

The RNN is trained to minimize both the sequence–to–sequence loss (eq. 2.19) and the categorical cross entropy (Eq. 2.17) between the gold standard domain category y and the predicted domain–category y'.

All implementation details are the same as the DCA–RNN (Section 2.2).

2.3.2 Results and Discussion

We computed the distributed representations of whole sentences in our data set (Tables 1.1, 1.2) by either S2S–RNN or Joint–RNN. We apply these sentence representations to one–class classification for OOD sentence detection (Section III).

We also evaluate domain–category analysis of Joint–RNN. The experimental data and the evaluation measure are the same as in Section 2.2.

The best accuracy (98.81%) in the domain–category analysis experiment (Table 2.3) based on Joint–RNN was obtained by the vanilla RNN and two–channel word embedding. Joint–RNN was still more accurate the SVMs (Table 2.1), but less accurate than DCA–RNN (Table 2.2). The feature extractor of Joint–RNN focuses on features about not only domain–category but also overall sentence meaning, so that feature extractor cannot help losing the accuracy of domain–category analysis. In the next section, we will evaluate the effects of the proposed sentence–embedding methods for OOD sentence detection.

III. One–class Classification for Out–of–domain Sentence Detection

Another important part of OOD sentence detection is one–class classification that uses the training data of a target class to distinguish target items from uninteresting items.

Local outlier factor [51] is a one–class classification algorithm that compares the local density of a point to the local densities of its neighbors, and considers the point that has lower density than their neighbors as an outlier. The local density of a point is defined by the distance to its nearest neighbors. However, this method suffers from the curse of dimensionality, and a point with a low density is not always an outlier when sparse training data are used.

One–class SVM [52] is a one–class classification algorithm that treats the origin as a negative example to learn a decision function. However, one–class SVMs were very sensitive to the parameters and choice of kernel [53].

3.1 One–class Classification based on Autoencoder

We defined OOD sentence detection $f(x)$ as a binary classification problem (Section I). However, unlike most other binary classification problems, we assume that only in–domain sentences are available as training data. In this dissertation, we apply autoencoders to one–class classification for OOD sentence detection. An autoencoder consists of an encoder and a decoder; the characteristic of the autoencoder is that the reconstruction error is low for *interesting* data but high for *uninteresting* data. Therefore, we expect the autoencoder trained on in–domain sentences to generate high reconstruction errors for OOD sentences.

Following the idea of one–class classification based on autoencoders [54, 55], we use the reconstruction error of the autoencoder as evidence that a sentence is OOD.

3.1.1 Methods

Recurrent Autoencoder

We introduced S2S–RNN and Joint–RNN in Section 2.3. Their encoders compute a distributed sentence representation $[\overrightarrow{\mathbf{h}_{\text{enc}}}^{(t)}, \overleftarrow{\mathbf{h}_{\text{enc}}}^{(1)}]$ from an input sentence $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)})$, where $\mathbf{x}^{(t)} \in \mathbb{N}^k$ is the one–hot vector representation of the t^{th} word in a given sentence, and k unique words are in the vocabulary. Their decoders generate output sentence $(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(n)})$ of the decoder from the distributed sentence representation. Therefore, we directly use either S2S–RNN or Joint–RNN as a recurrent autoencoder that detects OOD sentences of which sequence–to–sequence losses (Eq 2.19) are greater than a threshold θ as:

$$f(x) = \begin{cases} \textit{In–domain}, & \text{if } \frac{1}{n}\sum_{t=1}^{n} H(\mathbf{x}^{(t)}, \mathbf{z}'^{(t)}) < \theta, \\ \textit{OOD}, & \text{otherwise}, \end{cases} \tag{3.1}$$

where $\mathbf{z}'^{(t)})$ is the t^{th} predicted output word, and $H(p, q)$ is the categorical cross entropy (eq. 2.17).

A similar approach has been used for multi–sensor anomaly detection [56].

Feedforward Autoencoder

We propose feedforward autoencoders (Fig. 3.1) that use distributed sentence representations computed by sentence embedding (Section II), whereas recurrent autoencoders (Section 3.1.1) use one–hot vector representations of words. A feedforward autoencoder consists of an encoder ψ and a decoder ψ. Let $\mathbf{x} \in \mathbb{R}^m$ be an input distributed sentence representation. Then the encoding layer $\phi(\mathbf{x}) \in \mathbb{R}^{\frac{m}{2}}$

and the decoding layer $\psi(\phi(\mathbf{x})) \in \mathbb{R}^m$ are defined as

$$\phi(\mathbf{x}) = \sigma_h(\mathbf{W}_\phi \mathbf{x} + \mathbf{b}_\phi), \tag{3.2}$$

$$\psi(\phi(\mathbf{x})) = \sigma_h(\mathbf{W}_\psi \phi(\mathbf{x}) + \mathbf{b}_\psi), \tag{3.3}$$

where $\mathbf{W}_\phi$ and $\mathbf{W}_\psi$ are weight matrices, and $\mathbf{b}_\phi$ and $\mathbf{b}_\psi$ are bias vectors. We use all in–domain sentences to train this autoencoder by minimizing the reconstruction error $\|\mathbf{x} - \psi(\phi(\mathbf{x}))\|^2$.

Finally, we use the learned autoencoder to detect OOD sentences of which reconstruction errors are greater a threshold θ as:

$$f(x) = \begin{cases} \textit{In–domain}, & \text{if } \|\mathbf{x} - \psi(\phi(\mathbf{x}))\|^2 < \theta, \\ \textit{OOD}, & \text{otherwise.} \end{cases} \tag{3.4}$$

To implement our autoencoders for one–class classification, we use the Tensorflow library [47]. We train our models by using Adam [48] optimizer with a mini–batch size of 128 and an initial learning rate of 0.001 that is decreased linearly during 500 epochs. All weights are initialized from a zero–centered Normal distribution with standard deviation 1.0.

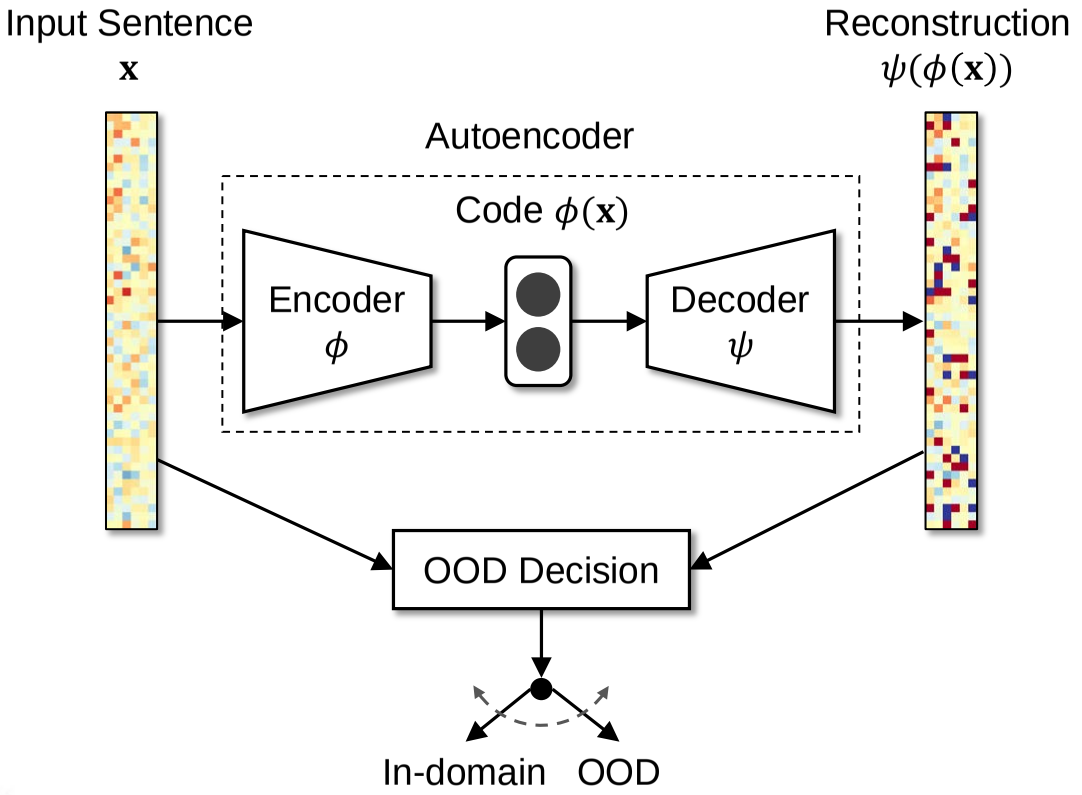
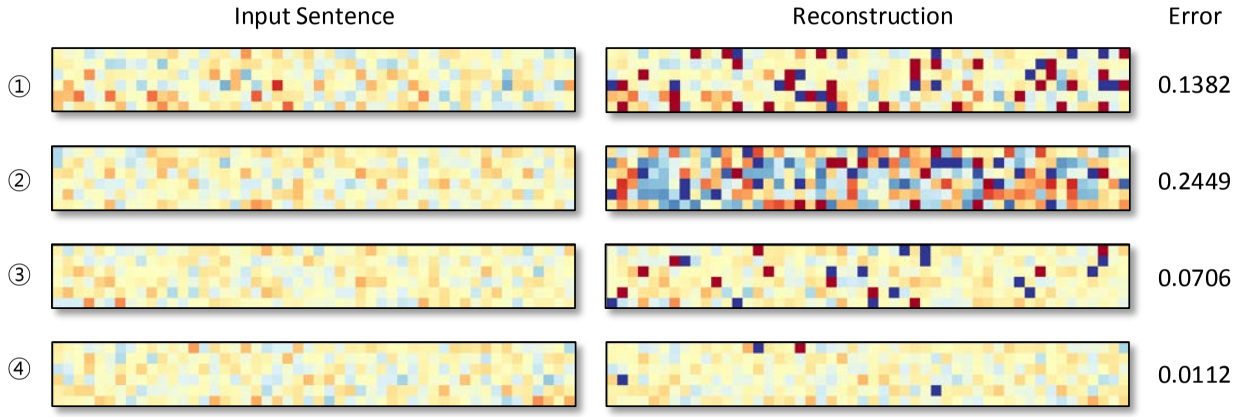


Figure 3.1: Feedforward autodencoder for out–of–domain sentence detection.

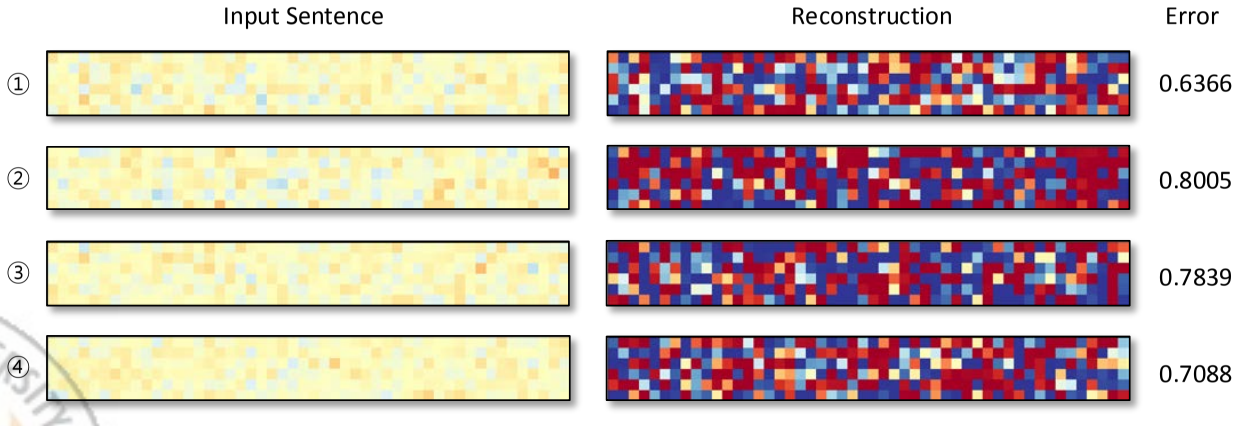
3.1.2 Results and Discussion

We used 80% of in–domain sentences in our data set (Table 1.2) to train our models, and used the remaining 20% of in–domain sentences and all OOD sentences (Table 1.1) for the test.

We first verified the characteristic of autoencoders: an autoencoder trained on interesting data generates high reconstruction errors for uninteresting data. We built the autoencoder in which DCA–RNN with LSTM cell and two–channel is used for sentence embedding. In the eight sample sentences (Fig. 3.2), the reconstruction errors of in–domain sentences and OOD sentences were obviously low and high respectively. We could obtain the same result in all data (Fig. 3.3), so we can say that the characteristic of autoencoders is valid in our data set.



(a) In–domain sentences



(b) Out–of–domain sentences

Figure 3.2: Reconstructed sentences and their reconstructions by the feedforward autoencoder.

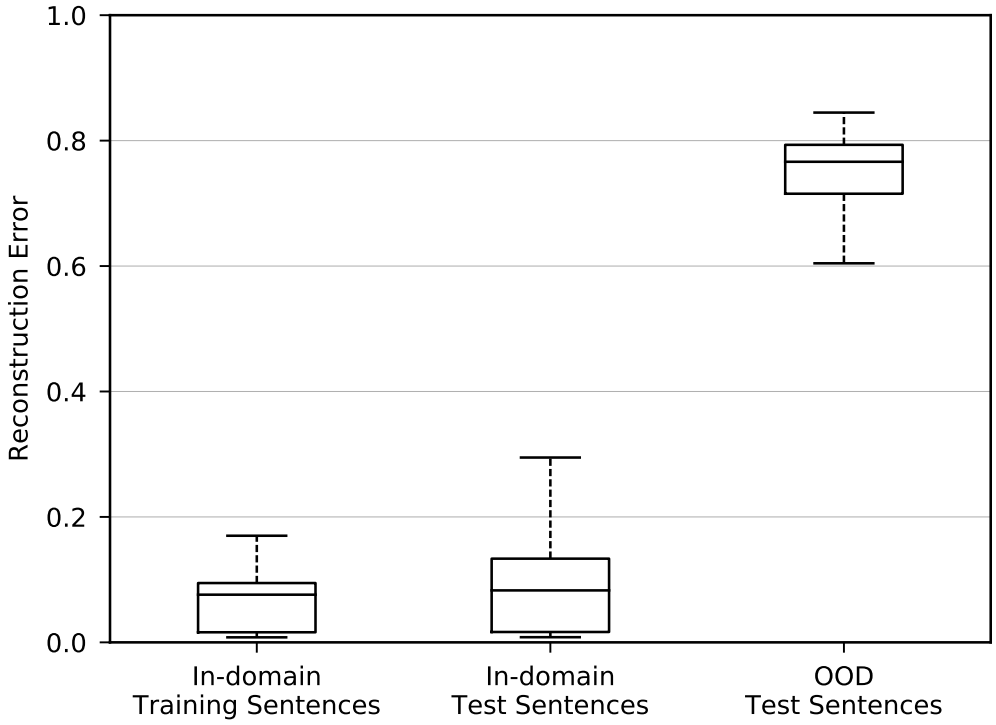


Figure 3.3: Reconstruction errors by the feedforward autoencoder.

Table 3.1: Equal error rates [%] of out–of–domain sentence detection based on the baseline one–class classification methods.

(a) In–domain verifiers

Feature	Best Kernel	EER
Bag–of–words (1–gram)	RBF	18.66
TF–IDF (1–gram)	Polynomial	18.43
Neural bag–of–words	Polynomial	18.15

(b) Local outlier factors

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	29.87	27.80	15.29	16.04
GRU	17.99	20.54	13.33	14.31
LSTM	28.46	23.64	12.19	13.33

(c) One–class SVMs

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	28.25	38.38	11.60	11.92
GRU	21.09	30.46	15.01	15.13
LSTM	27.64	28.62	18.04	13.76

As the baselines of OOD sentence detection, we used three one–class classification methods: in–domain verifier, local outlier factor, and one–class SVM that were introduced in Section 1.5 and this section. In–domain verifier uses symbolic word representations; one–class SVM and local outlier factor use distributed sentence representations computed by our sentence–embedding methods. In the experiments (Table 3.1), the best EERs were 18.15% for in–domain verifier, 12.19% for local outlier factor, and 11.60% for one–class SVM.

Table 3.2: Equal error rates [%] ± s.d. (n = 20) of out–of–domain sentence detection based on recurrent autoencoders.

(a) S2S–RNN autoencoder

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	37.89 ± 0.58	18.42 ± 0.47	17.03 ± 0.58	16.90 ± 0.98
GRU	35.28 ± 0.66	18.27 ± 0.51	17.72 ± 0.72	17.66 ± 0.78
LSTM	35.61 ± 0.70	18.47 ± 0.30	16.73 ± 0.69	16.53 ± 0.74

(b) Joint–RNN autoencoder

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	36.15 ± 0.46	19.03 ± 0.43	17.32 ± 0.63	17.18 ± 0.66
GRU	34.98 ± 0.72	18.79 ± 0.33	16.94 ± 0.65	16.87 ± 0.56
LSTM	36.53 ± 0.55	18.34 ± 0.39	16.97 ± 0.36	16.92 ± 0.70

For OOD sentence detection based on recurrent auteoncoder (Section 3.1.1), we compared four variations of word embedding. We performed each experiment 20 times, and recorded the average EER of OOD sentence detection.

In the experiments (Table 3.2), the best results were 16.53% for S2S–RNN autoencoder and 16.87% for Joint–RNN autoencoder. These results mean that recurrent autoencoders are less accurate than one–class SVM. We think the reason is that the recurrent autoencoders use symbolic words to compute reconstruction errors: an in–domain sentence can have high reconstruction error when the generated sentence is semantically similar but superficially dissimilar to the input sentence. Therefore, sentences on which the recurrent autoencoders cause large reconstruction errors are not necessarily OOD.

Table 3.3: Equal error rates [%] ± s.d. (n = 20) of out–of–domain sentence detection based on feedforward autoencoders.

(a) Sentence embedding based on DCA–RNN

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	23.47 ± 0.63	35.64 ± 7.20	10.81 ± 0.35	11.06 ± 0.49
GRU	18.24 ± 0.66	32.83 ± 15.07	9.99 ± 0.41	10.19 ± 0.97
LSTM	21.17 ± 1.28	27.14 ± 9.77	9.41 ± 0.23	9.24 ± 0.43

(b) Sentence embedding based on S2S–RNN

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	40.28 ± 0.84	33.78 ± 0.84	45.24 ± 1.11	44.18 ± 1.15
GRU	40.04 ± 1.06	34.86 ± 1.50	49.89 ± 3.36	43.61 ± 1.93
LSTM	44.20 ± 1.62	36.86 ± 2.03	47.38 ± 2.33	45.09 ± 3.07

(c) Sentence embedding based on Joint–RNN

RNN Cell	Word Embedding			
	Random	Static	Non–static	Two–channel
RNN	24.48 ± 0.65	39.49 ± 7.67	11.43 ± 0.75	12.92 ± 0.69
GRU	19.44 ± 0.94	31.01 ± 11.97	10.46 ± 0.57	10.18 ± 0.48
LSTM	21.48 ± 1.07	23.78 ± 7.32	10.28 ± 0.87	10.15 ± 0.68

Finally, we measured OOD sentence detection based on the feedforward autoencoders. We compared DCA–RNN, S2S–RNN, and Joint–RNN for sentence embedding, in which four variations of word embedding are used. We performed each experiment 20 times, and recorded the average EER of OOD sentence detection.

In the experiments (Table 3.3), the best EER (9.24%) was obtained by sentence embedding based on DCA–RNN with LSTM cell and two–channel word embedding. We present five observations from the experiments.

(1) Feedforward autoencoder was the most accurate one–class classification method ($p < 0.01$). We have already verified the autoencoder's characteristic that generates high reconstruction errors for OOD sentences. Because the best result was obtained by the feedforward autoencoder, we can say high reconstruction error by the feedforward autoencoder is the most reliable evidence of OOD.

(2) Domain–category analysis was a suitable auxiliary task in sentence embedding for OOD sentence detection ($p < 0.01$). In contrast, the sequence–to–sequence learning methods cannot optimize the sentence representations for OOD sentence detection, so those sentence–embedding methods were not accurate. However, sequence–to–sequence was not useful in sentence embedding for OOD sentence detection.

(3) LSTM cell was more accurate than RNN cell and GRU cell ($p < 0.01$). We used LSTM cells to prevent the RNN from the vanishing gradient problem. The results of vanilla RNNs and LSTM networks did not differ significantly in the domain–category analysis experiment because the average number of words in in–domain sentences was small (5.16, s.d. 3.02, $n = 5562$). However, because OOD sentences were longer (7.05, s.d. 3.33, $n = 706$) (Fig. 3.4) than in–domain sentences on average, LSTM networks demonstrated their advantage in OOD sentence detection.

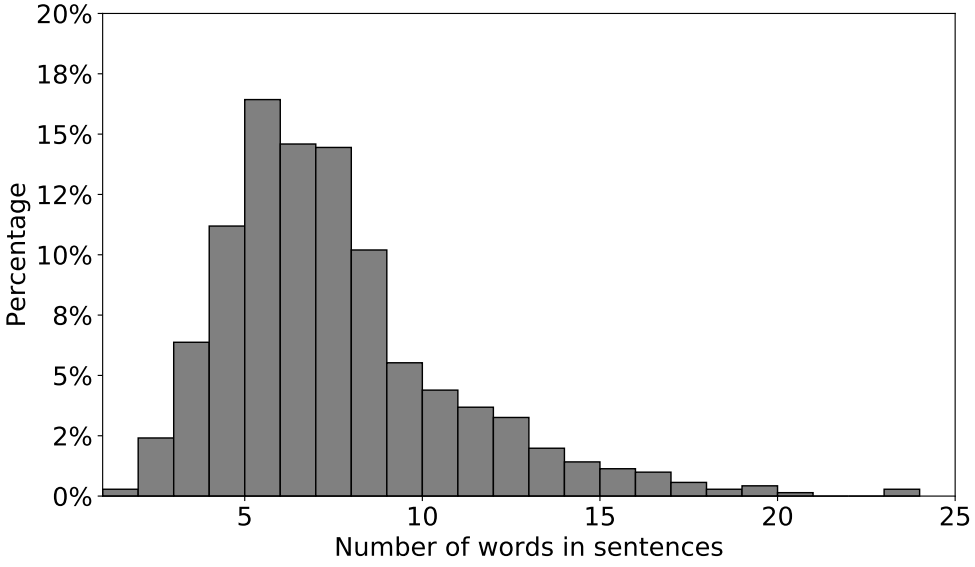


Figure 3.4: Histogram of number of words in out–of–domain sentences.

(4) Two–channel word embedding could understand unknown words and rare words accurately, but the difference was not significant ($p > 0.05$). We have reached the same result in the domain–category analysis experiment, but OOD sentence detection takes much advantage of two–channel word embedding. We think the reason is that understanding unknown or rare words is more important in OOD sentence detection than in domain–category analysis.

(5) The autoencoder had a limitation in expandability. When we initialize the weights carefully and apply the dropout technique, the trained autoencoder reconstructs any input accurately (Fig. 3.5). This result means that the reconstruction errors by the *ideal* autoencoder are not reliable evidence of OOD, so in OOD sentence detection, autoencoders have little room for improvement.

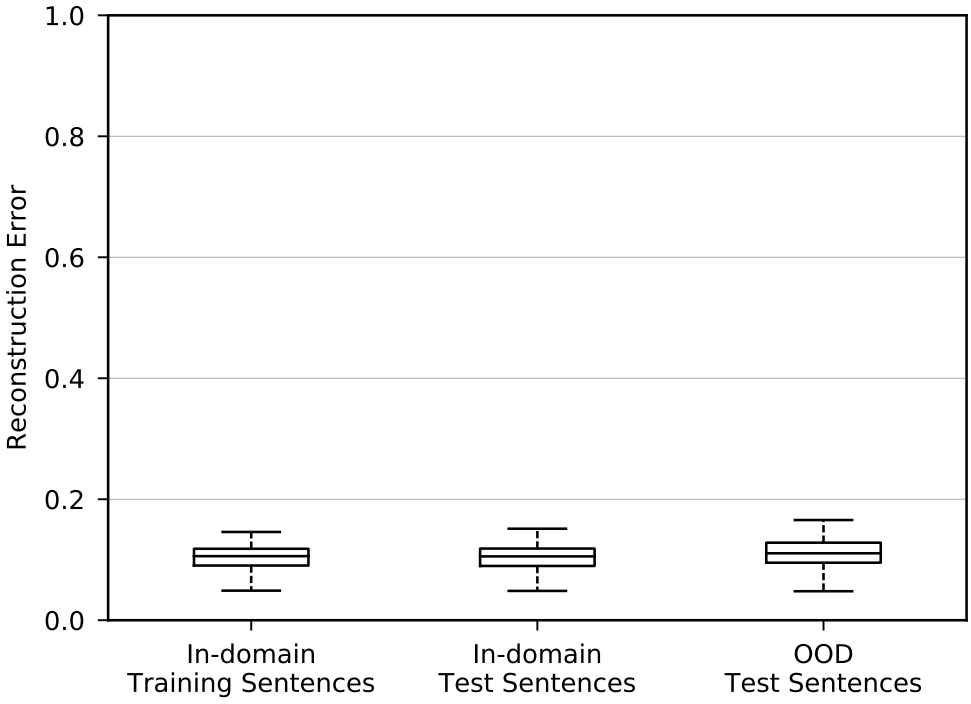


Figure 3.5: Reconstruction errors by the regularized feedforward autoencoder.

3.2 One–class Classification based on Generative Adversarial Network

A generative adversarial network (GAN) [13] consists of two adversarial models: generator G and discriminator D. G generates artificial data to deceive D. D distinguishes real data from the artificial data generated by G. GAN is an unsupervised algorithm because learning G and D does not require label. Standard GANs are trained based on the objective function $V(D, G)$ as

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log 1 - D(G(\boldsymbol{z}))]. \quad (3.5)$$

So GAN is a minimax two–player game because G minimizes $V(D, G)$, and D maximizes $V(D, G)$.

Recently GANs have been used for many computer–vision and image–processing tasks including image–generation from text [57], image–domain transfer [58], image super–resolution [59], image in–painting [60], and object detection [61]. GANs are also used for natural language processing tasks including language modeling [62], machine translation [63], and dialog generation [64].

We propose to use GANs to obtain a one–class classifier for OOD sentence detection. When we train G to generate sentences similar to in–domain sentences and D to classify real in–domain sentences and fake sentences generated by G, we expect D to reject OOD sentences. Therefore, we use the low confidence score by D about an input sentence as the evidence that the sentence is OOD.

3.2.1 Methods

We use m-dimensional continuous sentence representations computed by DCA–RNN with LSTM cell and two–channel word embedding because that was the best sentence–embedding method for a feedforward autoencoder (Section 3.1).

Let $p_{\boldsymbol{z}}(\boldsymbol{z})$ be a continuous uniform distribution $(-1, 1)$. We define G that

generates fake data $G(\mathbf{z}) \in \mathrm{R}^m$ from input noise $\mathbf{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$, $\mathbf{f}$ that extracts features from either real data $\mathbf{x}$ of $(\mathbf{x}, \mathbf{y}) \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})$ or $G(\mathbf{z})$, and D that measures the probability of either $\mathbf{f}(\mathbf{x})$ or $\mathbf{f}(G(\mathbf{z}))$ from the real data as

$$G(\mathbf{z}) = \sigma_h(\mathbf{W}_g\mathbf{z}), \tag{3.6}$$

$$\mathbf{f}(\mathbf{x}) = \sigma_h(\mathbf{W}_h\mathbf{x} + \mathbf{b}_f), \tag{3.7}$$

$$\mathbf{f}(G(\mathbf{z})) = \sigma_h(\mathbf{W}_h G(\mathbf{z}) + \mathbf{b}_f), \tag{3.8}$$

$$D(\mathbf{f}(\mathbf{x})) = \sigma_g(\mathbf{W}_d\mathbf{f}(\mathbf{x}) + \mathbf{b}_d), \tag{3.9}$$

$$D(\mathbf{f}(G(\mathbf{z}))) = \sigma_g(\mathbf{W}_d\mathbf{f}(G(\mathbf{z})) + \mathbf{b}_d), \tag{3.10}$$

where $\mathbf{W}_g$, $\mathbf{W}_f$, and $\mathbf{W}_d$ are weight matrices, $\mathbf{b}_f$ and $\mathbf{b}_d$ are bias vectors, and y is true domain–category. So we define two objective functions

$$L_{\mathrm{D}} = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}[\log D(\mathbf{f}(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(\mathbf{f}(G(\mathbf{z}))))], \tag{3.11}$$

$$L_{\mathrm{G}} = -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log D(\mathbf{f}(G(\mathbf{z})))]. \tag{3.12}$$

Because $G(\mathbf{z})$ continuously changes during the training, f of traditional GAN also changes according to the changing $G(\mathbf{z})$. Thus we define $C \in \mathbb{R}^{|D|}$ that computes domain–category of $\mathbf{f}(\mathbf{x})$ as

$$C(\mathbf{f}(\mathbf{x})) = \mathrm{softmax}(\mathbf{W}_c\mathbf{f}(\mathbf{x}) + \mathbf{b}_c), \tag{3.13}$$

where $\mathbf{W}_c$ is a weight matrix and $\mathbf{b}_c$ is a bias vector. We expect f trained by the losses of both D and C to be more stable than f trained by the loss of only D, so we define an objective function

$$L_{\mathrm{C}} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}[H(C(\mathbf{f}(\mathbf{x})), \mathbf{y})], \tag{3.14}$$

where $H(p, q)$ is the categorical cross entropy (eq. 2.17) and $\mathbf{y}$ is the true domain–category.

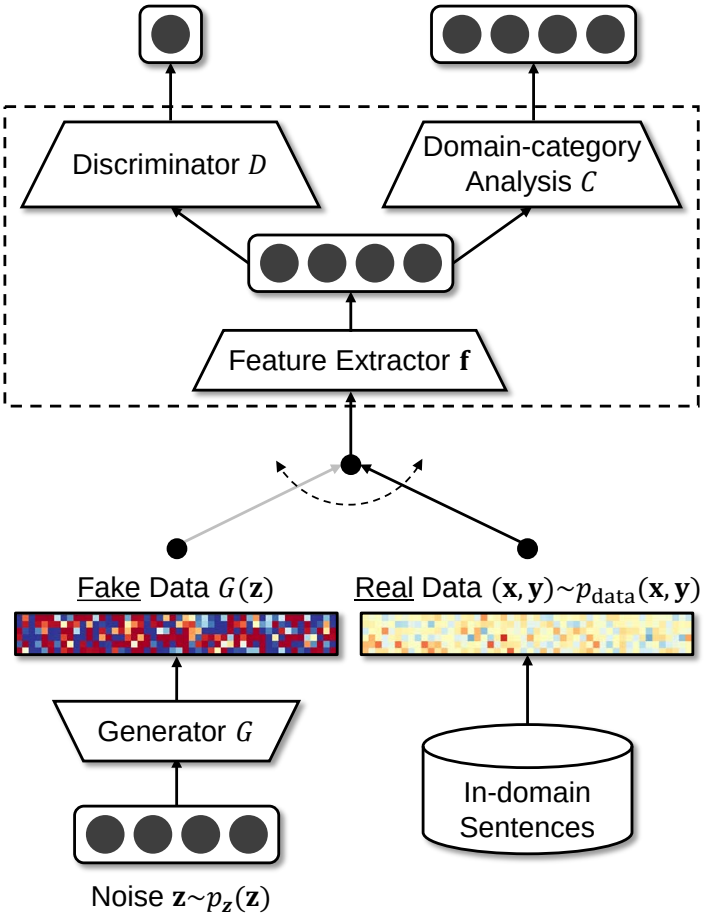


Figure 3.6: Generative adversarial network for out–of–domain sentence detection.

Algorithm 1 Training process of generative adversarial network for OOD sentence detection.

for number of training iterations **do**

Sample minibatch of real data $(\mathbf{x}, \mathbf{y}) \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})$.

Sample minibatch of noise $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$.

Update D, C, and $\mathbf{f}$ by the gradient of $L_{\text{D}} + L_{\text{C}}$ (Eq. 3.11, 3.14).

Sample minibatch of noise $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$.

Update G by the gradient of $L_{\text{G}} + L_{\text{f}}$ (Eq. 3.12, 3.15).

end for

In addition, GAN suffers from a *mode collapse* problem in which G generates samples with a low variance. To solve the problem, we remove the biases in the generator because the G was trained to use the biases mainly instead of the weights to generate data.

Second, we apply feature matching [65]. The authors say "feature matching address the instability of GANs by specifying a new objective for the generator that prevents it from overtraining on the current discriminator. Instead of directly maximizing the output of the discriminator, the new objective requires the generator to generate data that matches [sic] the statistics of the real data, where we use the discriminator only to specify the statistics that we think are worth matching". So G is trained to generate high variance sentence $G(\mathbf{z})$ by additional objective function

$$L_{\text{f}} = ||\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}\mathbf{f}(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\mathbf{f}(G(\mathbf{z}))||_2^2. \tag{3.15}$$

Based on our design of GAN, we train D, C, $\mathbf{f}$, and G as Algorithm 1. To implement our GAN for one–class classification, we use the Tensorflow library [47]. We train our models by using Adam [48] optimizer with a mini–batch size of 256 and an initial learning rate of 0.01 that is decreased linearly during 500 epochs. All weights are initialized from a zero–centered Normal distribution with standard deviation 1.0.

Table 3.4: Equal error rates [%] ± s.d. (n = 20) of out–of–domain sentence detection based on generative adversarial networks.

Method	EER
GAN with biases	15.93 ± 5.82
GAN	9.18 ± 0.30
GAN with DCA task	9.17 ± 0.40
GAN with FM loss	9.04 ± 0.30
GAN with DCA task and FM loss	8.96 ± 0.34

3.2.2 Results and Discussion

We used 80% of in–domain sentences in our data set (Table 1.2) to train our models, and used the remaining 20% of in–domain sentences and all OOD sentences (Table 1.1) for the test. We used EER (Section 1.4) as the evaluation measure of OOD sentence detection.

We have three variations of vanilla GAN: to remove the biases, to add domain–category analysis (DCA) task, and to add feature matching (FM) loss. So we assessed five settings of GAN. We performed each experiment 20 times, and recorded the average EER of OOD sentence detection.

In the experiments (Table 3.4), the best EER (8.96%) was obtained by the GAN in which all three of our variations are applied ($p < 0.05$). This result means that (1) removing the biases and using the feature matching prevented the generator from mode collapse problem and (2) using domain–category analysis as an auxiliary task stabilized the training of feature extractor.

The proposed GAN was more accurate than the feedforward autoencoder ($p < 0.05$) for OOD sentence detection, so we can say that the discriminator scores (Fig. 3.7) of the GAN were more reliable evidence for OOD sentence detection than the reconstruction errors by the feedforward autoencoder.

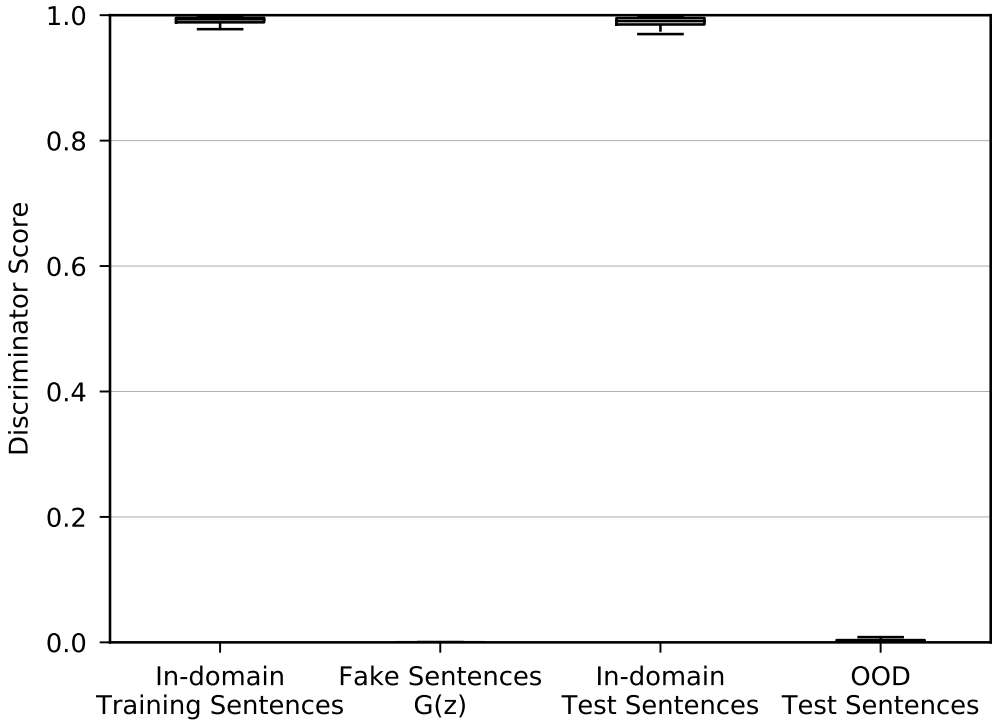


Figure 3.7: Scores by the discriminator of the generative adversarial network.

IV. Conclusion

4.1 Concluding Remarks

In this dissertation, we addressed two topics of OOD sentence detection learned from only in–domain sentences: (1) sentence embedding that extracts meaningful features about the domain of the input sentence and (2) one–class classification that rejects OOD sentences.

To approach the sentence embedding problem (Section II), we used a large set of unlabeled text to pre–train the distributed representations of Korean words. We trained an RNN for domain–category analysis and used the feature extractor of the RNN as a sentence–embedding system for OOD sentence detection. We applied LSTM cells to prevent the vanishing gradient problem and used two–channel word embedding to prevent the distributed word representations from being biased towards in–domain training data.

To approach the one–class classification problem (Section III), we proposed to use feedforward autoencoders. Because we trained a feedforward autoencoder that reconstructs in–domain sentences, we used a high reconstruction error by the autoencoder as evidences that an input sentence is OOD. Our second method for one–class classification is to use the discriminator of a GAN. We trained a discriminator that distinguishes in–domain sentences from fake sentences produced a generator, and used a low discriminator score as evidence that an input sentence is OOD. We removed the biases in the generator, added domain–category analysis task, and added feature matching loss to improve the GAN. We found that the discriminator of the GAN is more accurate than the feedforward autoencoder for OOD sentence detection.

4.2 Future Research Directions

Joint learning of sentence embedding and one–class classification. We learned sentence embedding and one–class classification *separately*. We first obtain the distributed sentence representations, and then used those representations to train one–class classifiers including the feedforward autoencoder and the GAN. However, the two tasks can be learned *jointly* in a single neural network. Although the result of the joint learning was unstable and inaccurate in our separate experiment, we think one–class classifiers can be improved by the joint learning.

Multi–task learning of natural language understanding. We tackled two problems: OOD sentence detection and domain–category analysis, of natural language understanding component of dialog systems. Recently, some studies [66, 67] have been conducted on *multi–task learning* of natural language understanding. Because our attempt to use domain–category analysis as an auxiliary task for OOD sentence detection has succeeded, we expect that other tasks such as intent analysis or named entity recognition can increase the accuracy of OOD sentence detection.

OOD sentences detection in end–to–end dialog systems. Recent studies use end–to–end learning [68, 69, 70, 71, 72] to solve many problems [68] in traditional dialog systems including a credit–assignment problem and process interdependence. However, responding appropriately OOD sentences is still a challenge issue in end–to–end dialog systems because most of them are trained only for a few target domains. We think end–to–end dialog systems in the real world also need to detect OOD sentences to improve user experience.

요 약 문

대화 시스템은 사람과 기계 사이의 상호작용을 위한 대화 인터페이스를 제공한다. 예를 들어 사용자는 대화 시스템을 통해 일정을 관리하고, 날씨 정보를 얻고, 음악을 검색 할 수 있다. 이러한 대화 시스템은 최근 음성 인식 및 자연어 처리 기술의 비약적인 발전에 따라 차세대 인터페이스로 주목 받고 있다. 그러나 일반적인 대화 시스템은 특정한 영역에 대한 서비스만을 제공할 수 있으므로 사용자로부터 처리 할 수 없는 요청을 받는 경우 적합한 응답을 제공 할 수 없다. 그러므로 대화 시스템이 사용자로부터 영역 외 요청을 받은 경우 이를 감지해 요청을 거절하거나 대안을 제시 함으로써 사용자에게 향상된 대화 경험을 제공 할 수 있다.

본 박사 학위 논문은 대화 시스템을 위한 영역 외 문장 검출 방법을 제안한다. 제안하는 방법론은 데이터 기반 대화 시스템 개발 과정에서 이미 수집된 영역 내 문장만을 학습에 사용하며 영역 외 문장은 필요로 하지 않는다. 학습을 위해 다양한 영역에 대한 충분한 양의 영역 외 문장을 수집하는 것은 많은 노력과 시간을 필요로 하며, 서비스 과정에서 대화 시스템의 영역의 정의가 변경될 때 마다 기존에 수집한 모든 영역 외 문장들의 영역 외 여부를 재검토 할 필요가 있기 때문이다.

제안하는 영역 외 문장 검출 방법론은 두 단계로 구성된다. 첫 번째 단계에서는 사용자로부터 입력 받은 문장을 분산 표현으로 변환한다. 서로 다른 단어로 구성된 문장일지라도 영역 외 문장 검출 관점에서 그 의미가 유사하다면 그 분산 표현이 벡터 공간 상에서 가까운 거리를 갖도록 하기 위함이다. 이를 위해 입력 문장의 영역 범주를 분석하는 재귀 신경망(recurrent neural network)을 학습한다. 해당 재귀 신경망에 의해 추출된 특징은 영역에 대한 정보를 담고 있으므로 영역 외 문장 검출에 사용한다.

제안하는 방법론의 두 번째 단계에서는 추출된 분산 표현을 바탕으로 입력 문장의

영역 외 여부를 판별한다. 이를 위해 분산 표현을 저차원으로 압축하는 인코더(encoder)와 원래대로 복원하는 디코더(decoder)로 구성된 오토인코더(autoencoder)를 학습한다. 학습된 오토인코더는 영역 외 문장에 대해서는 높은 복원 오류를 가지므로 이를 영역 외 문장 검출에 사용한다. 두 번째 단계를 위한 다른 방법은 실제 분산 표현과 만들어진 분산 표현을 구별하는 구분자(discriminator)와 구분자를 속이기 위해 분산 표현을 만들어내는 생성자(generator)로 구성된 생성적 적대 신경망(generative adversarial network)을 학습하는 것이다. 학습된 구분자는 영역 외 문장에 대해서는 낮은 점수를 부여하므로 이를 영역 외 문장 검출에 사용한다. 실험을 통해 이 박사 학위 논문이 제안한 방법론이 영역 외 문장을 정확하게 검출 할 수 있음을 검증하였다.

References

[1] Michael F. McTear. *Spoken Dialogue Technology: Toward the Conversational User Interface.* Springer, 2004.

[2] Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. Multi–domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of Interspeech*, 2016.

[3] Ridong Jiang, Rafael E. Banchs, Seokhwan Kim, Kheng Hui Yeo, Arthur Niswar, and Haizhou Li. Web–based multimodal multi-domain spoken dialogue system. In *Proceedings of IWSDS*, 2014.

[4] Donghyeon Lee, Minwoo Jeong, Kyungduk Kim, Seonghan Ryu, and Gary Geunbae Lee. Unsupervised spoken language understanding for a multi–domain dialog system. *IEEE/ACM Trans. Audio, Speech, Language Process.*, 21:2451–2464, 2013.

[5] Seonghan Ryu, Jaiyoun Song, Sangjun Koo, Soonchoul Kwon, and Gary Geunbae Lee. Detecting multiple domains from user's utterance in spoken dialog system. In *Proceedings of IWSDS*, 2015.

[6] Choong-Nyoung Seon, Hyunjung Lee, Harksoo Kim, and Jungyun Seo. Improving domain action classification in goal–oriented dialogues using a mutual retraining method. *Pattern Recogn. Lett.*, 45, 2014.

[7] Bor-shen Lin, Hsin-min Wang, and Lin-shan Lee. A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history. In *Proceedings of IEEE ASRU*, 1999.

[8] Mikio Nakano, Shun Sato, Kazunori Komatani, Kyoko Matsuyama, Kotaro Funakoshi, and Hiroshi G. Okuno. A two–stage domain selection framework for extensible multi–domain spoken dialogue systems. In *Proceedings of SIGDIAL*, 2011.

[9] Gokhan Tur, Anoop Deoras, and Dilek Hakkani-Tur. Detecting out–of–domain utterances addressed to a virtual personal assistant. In *Proceedings of Interspeech*, 2014.

[10] Kazunori Komatani, Naoyuki Kanda, Mikio Nakano, Kazuhiro Nakadai, Hiroshi Tsujino, Tetsuya Ogata, and Hiroshi G. Okuno. Multi–domain spoken dialogue system with extensibility and robustness against speech recognition errors. In *Proceedings of SIGDIAL*, 2009.

[11] Qi Li, Gokhan Tur, Dilek Hakkani-Tur, Xiang Li, Tim Paek, Asela Gunawardana, and Chris Quirk. Distributed open–domain conversational understanding framework with domain independent extractors. In *Proceedings of IEEE SLT*, 2014.

[12] Gokhan Tur, Rukmini Iyer, Larry Heck, and Dilek Hakkani-Tur. Translating natural language utterances to search queries for SLU domain detection using query click logs. In *Proceedings of ICASSP*, 2012.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of NIPS*, 2014.

[14] Ian Lane, Tatsuya Kawahara, Tomoko Matsui, and Satoshi Nakamura. Out–of–domain utterance detection using classification confidences of multiple topics. *IEEE/ACM Trans. Audio, Speech, Language Process.*, 15:150–161, 2007.

[15] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Interspeech*, 2010.

[16] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, 2014.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc Le. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, 2014.

[18] Oriol Vinyals and Quoc V. Le. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*, 2015.

[19] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. In *Proceedings of AAAI*, 2015.

[20] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, 2015.

[21] K. Arkhipenko, I. Kozlov, J. Trofimovich, K. Skorniakov, A. Gomzin, and D. Turdakov. Comparison of neural network architectures for sentiment analysis of russian tweets. In *Proceedings of Dialogue*, 2016.

[22] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi–task learning. In *Proceedings of IJCAI*, 2016.

[23] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Netw.*, 1:339–356, 1988.

[24] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.–Based Syst.*, 6:107–116, 1998.

[25] Junyoung Chung, KyungHyun Cho, Çaglar Gülçehre, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*, 2014.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short–term memory. *Neural Comput.*, 9:1735–1780, 1997.

[27] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural Comput.*, 12:2451–2471, 2000.

[28] John Rupert Firth. A synopsis of linguistic theory 1930–1955. *Studies in Linguistic Analysis*, pages 1–32, 1957.

[29] Magnus Sahlgren. The distributional hypothesis. *Ital. J. Linguist.*, 20:33–53, 2008.

[30] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, 2013.

[31] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of LREC*, 2010.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

[33] Laurens van der Maaten and Geoffrey Hinton. Visualizing high–dimensional data using t–SNE. *J. Mach. Learn. Res.*, 9:2579–2605, 2008.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit–learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[35] Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.*, 15:1–21, 2014.

[36] Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributional semantics. In *Proceedings of ACL*, 2014.

[37] Bing Liu and Ian Lane. Recurrent neural network structured output prediction for spoken language understanding. In *Proceedings of NIPS*, 2015.

[38] Vedran Vukotic, Christian Raymond, and Guillaume Gravier. Is it time to switch to word embedding and recurrent neural networks for spoken language understanding? In *Proceedings of Interspeech*, 2015.

[39] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. Recurrent neural networks for language understanding. In *Proceedings of Interspeech*, 2013.

[40] Suman Ravuri and Andreas Stolcke. Recurrent neural network and LSTM models for lexical utterance classification. In *Proceedings of Interspeech*, 2015.

[41] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. Spoken language understanding using long short-term memory neural networks. In *Proceedings of IEEE SLT*, 2014.

[42] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22:1345–1359, 2010.

[43] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, 2014.

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[45] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.*, 18:602–610, 2005.

[46] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans Sig. Process.*, 45:2673–2681, 1997.

[47] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large–scale machine learning on heterogeneous distributed systems, 2015. Software available from tensorflow.org.

[48] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

[49] Corinna Cortes and Vladimir Vapnik. Support–vector networks. *Machine Learning*, 20, 1995.

[50] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *Proceedings of ICML*, 2015.

[51] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density–based local outliers. In *Proceedings of ACM SIGMOD*, 2000.

[52] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high–dimensional distribution. *Neural Comput.*, 13:1443–1471, 2001.

[53] Larry M. Manevitz and Malik Yousef. One–class SVMs for document classification. *J. Mach. Learn. Res.*, 2:139–154, 2001.

[54] Larry Manevitz and Malik Yousef. One–class document classification via neural networks. *Neurocomputing*, 70:1466–1481, 2007.

[55] Liheng Xu, Kang Liu, and Jun Zhao. Joint opinion relation detection using one-class deep neural network. In *Proceedings of COLING*, 2014.

[56] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM–based encoder–decoder for multi–sensor anomaly detection. In *Proceedings of ICML Anomaly Detection Workshop*, 2016.

[57] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of ICRL*, 2016.

[58] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross–domain image generation. In *arXiv*, 2016.

[59] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo–realistic single image super–resolution using a generative adversarial network. In *Proceedings of CVPR*, 2016.

[60] Raymond Yeh, Chen Chen, Teck-Yian Lim, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with perceptual and contextual losses. In *Proceedings of CVPR*, 2017.

[61] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of CVPR*, 2017.

[62] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *arXiv*, 2017.

[63] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving neural machine translation with conditional sequence generative adversarial nets. In *arXiv*, 2017.

[64] Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *arXiv*, 2017.

[65] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Proceedings of NIPS*, 2016.

[66] Bing Liu and Ian Lane. Attention–based recurrent neural network models for joint intent detection and slot filling. In *Proceedings of Interspeech*, 2016.

[67] Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. Multi–domain joint semantic

frame parsing using bi–directional RNN–LSTM. In *Proceedings of Interspeech*, 2016.

[68] Tiancheng Zhao and Maxine Eskénazi. Towards end–to–end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of SIGDIAL*, 2016.

[69] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. A network-based end–to–end trainable task–oriented dialogue system. In *Proceedings of EACL*, 2016.

[70] Antoine Bordes and Jason Weston. Learning end–to–end goal–oriented dialog. In *Proceedings of ICLR*, 2017.

[71] Xuesong Yang, Yun-Nung Chen, Dilek Z. Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. End–to–end joint learning of natural language understanding and dialogue manager. In *Proceedings of ICASSP*, 2017.

[72] Bing Liu and Ian Lane. An end–to–end trainable neural network model with belief tracking for task–oriented dialog. In *Proceedings of the Interspeech*, 2017.

감 사 의 글

먼저 그 동안 아들을 믿고 지원해주신 부모님께 감사의 말씀을 드립니다. 항상 응원해주시고 종종 집에 갈 때마다 반갑게 맞아주셔서 힘을 낼 수 있었습니다. 앞으로 살아가면서 두 분의 사랑과 은혜에 보답하겠습니다. 항상 맡은 일에 성실한 동생에게도 응원을 보냅니다.

박사과정 동안 많은 가르침을 주신 이근배 교수님, 유환조 교수님 감사합니다. 학위 논문을 심사해주신 이종혁 교수님, 서정연 교수님, 최승진 교수님, 그리고 논문 영문교정을 맡아주신 Derek J. Lactin 교수님께도 감사의 말씀을 드립니다.

그 동안 연구실에서 즐거운 시간을 함께 보내고 많은 도움을 주신 영선누나, 종훈형, 석환형, 세천형, 형종형, 규송형, 경덕형, 동현형, 인재형, 준휘형, 홍석형, 용희, 지수, 상준, 선영, 병수, 상도, 순철, 대환 감사합니다. 특히 연구실의 남은 후배들이 좋은 결실을 거두기를 희망합니다.

인공지능 분야의 연구자가 되겠다는 꿈을 품고 2012년에 시작한 박사과정을 어느덧 마무리하게 되었습니다. 원하고 계획했던 것들을 모두 이루지 못한 것은 아쉽지만, 힘든 시간들을 견뎌내고 매듭을 지었다는 것이 자랑스럽게 여겨집니다. 사회에 진출해서도 포스텍 박사로서 부끄럽지 않도록 제가 맡은 일에 최선을 다하겠습니다.

2017년 12월, 포스텍 ISOFT 연구실의 정든 자리에서.

Curriculum Vitae

Name : Seonghan Ryu (류성한)

Education

2012. 3. – 2018. 2. **Pohang University of Science and Technology**

Ph.D. in Computer Science and Engineering

2007. 2. – 2012. 2. **Dongguk University**

B.S. in Computer Science and Engineering

Experience

2012. 3. – 2018. 2. **Pohang University of Science and Technology**

Research Assistant

Research Project

2016. 1. – 2017.12. **Intelligent Interaction Research for Life Companionship**

Ministry of Trade, Industry and Energy

2016. 3. – 2017. 4. **Development of Question Answering System based on Linked Data**

Ministry of Trade, Industry and Energy

2014. 5. – 2015. 1. **Development of Cognitive Dialog System**

Ministry of Science, ICT and Future Planning

2012. 3. – 2015. 9. **Development of Dialog System for Smart TV**

Samsung Electronics

Publication

International Journals

1. Seonghan Ryu, Seokhwan Kim, Junhwi Choi, Hwanjo Yu, and Gary Geunbae Lee, "Neural sentence embedding for out–of–domain sentence detection using only in–domain sentences in dialog systems," Pattern Recognit. Lett., 88(1), 2017.
2. Byeongchang Kim, Seonghan Ryu, and Gary Geunbae Lee, "Two–stage multi–intent detection for spoken language understanding", Multimed. Tools Appl., 76(9), 2017.
3. Junhwi Choi, Seonghan Ryu, Kyusong Lee, and Gary Geunbae Lee, "One–step error detection and correction approach for voice word processor", IE-ICE Trans. Inf. Syst., 98(8), 2015.
4. Donghyeon Lee, Minwoo Jeong, Kyungduk Kim, Seonghan Ryu, and Gary Geunbae Lee, "Unsupervised spoken language understanding for a multi–domain dialog system", IEEE Trans. Audio Speech Lang. Process., 21(11), 2013.

5. Hyungjong Noh, Seonghan Ryu, Donghyeon Lee, Kyusong Lee, Cheongjae Lee, and Gary Geunbae Lee, "An example-based approach to ranking multiple dialog states for flexible dialog management," IEEE J. Sel. Top. Signal Process., 6(8), 2012.

International Conferences & Workshops

1. Seonghan Ryu, Hwanjo Yu and Gary Geunbae Lee, "Two–stage approach to named entity recognition using Wikipedia and DBpedia," In *Proceedings of IMCOM*, 2017 (Poster).
2. Seonghan Ryu, Jaiyoun Song, Sangjun Koo, Soonchoul Kwon, and Gary Geunbae Lee, "Detecting multiple domains from user's utterance in spoken dialog system," In *Proceedings of IWSDS*, 2015.
3. Seonghan Ryu, Donghyeon Lee, Gary Geunbae Lee, Kyungduk Kim, and Hyungjong Noh, "Exploiting out–of–vocabulary words for out–of–domain detection in dialog systems," In *Proceedings of BigComp*, 2014.
4. Seonghan Ryu, Donghyeon Lee, Injae Lee, Sangdo Han, Gary Geunbae Lee, Myungjae Kim, and Kyungduk Kim, "A hierarchical domain model–based multi–domain selection framework for multi–domain dialog systems", In *Proceedings of COLING*, 2012 (Poster).
5. Sangjun Koo, Seonghan Ryu, and Gary Geunbae Lee, "Implementation of generic positive–negative tracker in extensible dialog system," Proceedings of *IEEE ASRU*, 2015.
6. Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee, "Exploiting knowledge base to generate responses for natural language dialog listening agents," In *Proceedings of SIGDIAL*, 2015 (Poster).
7. Sangdo Han, Hyosup Shim, Byungsoo Kim, Seonyeong Park, Seonghan Ryu,

and Gary Geunbae Lee, "Keyword question answering system with report generation for linked data," In *Proceedings of BigComp*, 2015 (Poster).

8. Sohyeon Jung, Seonghan Ryu, Sangdo Han, and Gary Geunbae Lee, "DietTalk: diet and health assistant based on spoken dialog system," In *Proceedings of IWSDS*, 2015 (Poster).

9. Sangjun Koo, Seonghan Ryu, Kyusong Lee, and Gary Geunbae Lee, "Scalable summary–state pomdp hybrid dialog system for multiple goal drifting requests and massive slot entity instances," Proceedings of IWSDS, 2015 (Poster).

10. Kyusong Lee, Seonghan Ryu, Paul Hongsuck Seo, Seokhwan Kim, and Gary Geunbae Lee, "Grammatical error correction based on learner comprehension model in oral conversation," In Proceedings of IEEE SLT, 2014.

11. Seonyeong Park, Donghyeon Lee, Junhwi Choi, Seonghan Ryu, Yonghee Kim, Soonchoul Kwon, Byungsoo Kim, and Geunbae Lee, "Hierarchical dirichlet process topic modeling for large number of answer types classification in open domain question answering," In Proceedings of AIRS, 2014 (Poster).

12. Yonghee Kim, Jeesoo Bang, Junhwi Choi, Seonghan Ryu, Sangjun Koo, "Acquisition and use of long–term memory for personalized dialog systems," In *Proceedings of MA3HMI Workshop*, 2014.

13. Junhwi Choi, Seonghan Ryu, Kyusong Lee, Yonghee Kim, Sangjun Koo, Jeesoo Bang, Seonyeong Park, and Gary Geunbae Lee, "ASR independent hybrid recurrent neural network based error correction for dialog system applications," In *Proceedings of MA3HMI Workshop*, 2014.

14. Jeesoo Bang, Kyusong Lee, Seonghan Ryu, and Gary Geunbae Lee, "Vowel–reduction feedback system for non–native learners of English," In *Proceedings*

of ICASSP, 2014 (Poster).

15. Junhwi Choi, Donghyeon Lee, Seonghan Ryu, Kyusong Lee, and Gary Geunbae Lee, "Engine–independent ASR error management for dialog systems," In *Proceedings of IWSDS*, 2014.

16. Injae Lee, Seokhwan Kim, Kyungduk Kim, Donghyeon Lee, Junhwi Choi, Seonghan Ryu, and Gary Geunbae Lee, "A two–step approach for efficient domain selection in multi–domain dialog systems," In *Proceedings of IWSDS*, 2012.

Domestic Journals

1. 류성한, 이근배, "대화 모델링을 위한 음성 언어 이해 기술," 대한전자공학회 학회지, 지능형 음성언어 소프트웨어의 최근 연구/개발 동향 특집, 41(3), 2014.

Domestic Conferences & Workshops

1. 남대환, 류성한, 유환조, 이근배, "한국어 개방형 정보추출을 위한 신경망 모델," 제 43회 정보과학회 동계학술발표회, 2016년 12월.

2. 최준휘, 류성한, 유환조, 이근배, "음성 인식 오류 수정을 위한 Trie 기반 사전을 이용한 Guided Sequence Generation," 제 28회 한글 및 한국어 정보처리 학술대회 논문집, 2016년 10월.

3. 최준휘, 류성한, 이규송, 박선영, 유환조, 이근배, "단어열 패턴 매칭과 Recurrent Neural Network를 이용한 하이브리드 음성 인식 오류 수정 방법," 제 27회 한글 및 한국어 정보처리 학술대회 논문집, 2015년 10월.

4. 정소현, 류성한, 이근배, "다이어트 톡 : 다이어트 및 건강 관리를 위한 대화 시스템", 제 41회 한국정보과학회 동계학술발표회 논문집, 2014년 12월.

5. 박선영, 이동현, 김용희, 류성한, 이근배, "질의응답 시스템을 위한 반교사 기반의 정답유형 분류," 제 25회 한글 및 한국어 정보처리 학술대회 논문집, 2013년

10월.

6. 최준휘, 이동현, 김용희, 류성한, 이인재, 이근배, “무결절적 오류수정 방법을 이용한 음성 문자 작성시스템,” 2012 한국 음성학회 봄 학술대회, 2012년 5월.

Awards

1. 2017년 국어정보처리시스템 경진대회 은상, 한국정보과학회 언어공학연구회, 2017년 10월.
2. 2013년 한글 및 한국어 정보처리 학술대회 우수 논문상, 한국정보과학회 언어공학연구회, 2013년 10월.

Teaching

2017. 2. – 2017. 6 **Pohang University of Science and Technology**
Teaching assistant, Artificial Intelligence

2016. 3. – 2016. 6 **Pohang University of Science and Technology**
Teaching assistant, Computer Network

2012. 9. – 2012.12 **Pohang University of Science and Technology**
Teaching assistant, Programming & Problem solving

2011. 3. – 2011. 6 **Dongguk University**
Teaching assistant, Data Structures and Practice


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교