Attention-based Dialog Embedding for Dialog Breakdown Detection

Chanyoung Park[1] *, Kyungduk Kim*[2]*, Songkuk Kim*[1]

[1]Yonsei University, Korea
[2]Clova Dialogue, Naver Corp., Korea

chanpark@yonsei.ac.kr, kyungduk.kim@navercorp.com, songkuk@yonsei.ac.kr

Abstract

Despite the recent advent of dialog systems, there are still many challenges unresolved and the system often generates responses causing a breakdown in the interaction between a user and the system. The dialog breakdown significantly damages user experience, and thus detecting such failure is significant. We propose a model to detect dialog breakdown using a recurrent neural network and attention layer to embed a previous dialog context. This model determines the probability of breakdown using two representation vectors, an extracted dialog context vector and the sentence vector of the target sentence. We submitted this model to *Dialog Breakdown Detection Challenge 3* of *Dialog System Technology Challenge 6*, and the results showed that it significantly outperforms most of the other submitted models in estimating breakdown probability.

Index Terms: dialog system, dialog breakdown detection, dialog embedding, sentence embedding, attention

1. Introduction

Despite the recent advent of dialog systems, we still face a tremendous number of unresolved challenges and the system often generates responses that cause a breakdown in the interaction between a user and the system. Martinovsky [1] has defined this situation as *dialog breakdown*, which is a specific case in a conversation when the interaction is interrupted with or without completion of the performed task because one or both parties give up the conversation. Such breakdowns severely damage users' contentment with the system, and thus detecting the dialog breakdown and handling those miscommunications has been recognized as an essential task for a robust dialog system.

There has been extensive research on *belief tracking* and using *confidence score* to prevent any miscommunication in a dialog system. Belief refers to the distribution over dialog states, which indicates how likely the defined dialog states fit in with the situation and intention of a user [2]. Representative systems, including Higgins [3] and the Let's Go! Public [4] system, endeavored to overcome potential dialog errors. Those systems measure and track the confidence of the system, and detect the possibility of dialog breakdown. However, the previous methods have two fundamental limitations. First, the belief tracking and the confidence score are not enough to estimate the probability of dialog breakdown. The uncertainty of the dialog states not only comes from the system's incorrect response but also comes from other parts of the dialog system, such as speech recognition. Second, using belief tracking methods requires other information from natural language understanding (NLU) and dialog management (DM) modules, in addition to the text data. System specific data such as the intermediate output of a dialog system is neither easily accessible nor universally applicable.

This work was performed during an internship at Naver

This paper focuses on dialog breakdown detection solely based on the text of the dialog. Our breakdown detector has two advantages over the traditional belief tracker. First, the dialog text is much easier to collect and thus offers an opportunity to apply deep neural networks to the task. Compared with the system specific data, the dialog text data can be enriched by various sources, such as movie scripts, books, lyrics, and Twitter. The resulting large size of data can enable the training of deep neural networks to perform the task. Furthermore, a text-based breakdown detector can provide both online prediction and offline analysis on past dialog log data. Online miscommunication prediction allows a dialog system to be more robust via evoking an error recovery mode or a confirmation message. Offline analysis enables an automatic identification of the weakness in the system and evaluation between different dialog systems.

Additional to the text, contextual information has been shown to be useful for various natural language tasks. Several works have proposed models with a recurrent neural network (RNN) that incorporates the contextual information and showed improved performance on domain classification and intent prediction tasks [5, 6]. Other research has applied attention layer to the RNN architecture to build memory networks for the slot tagging task [7]. The model has been shown to be good at carrying over the long-term knowledge presented in sentences.

Motivated by the model presented in [7], this paper presents a model that embeds the context of the dialog using the RNN and attention layer. Our model estimates the probability of dialog breakdown with embedded dialog context and the target system utterance. The proposed model was evaluated in the English subtask of *Dialog Breakdown Detection Challenge 3* (DBDC 3) of *Dialog System Technology Challenge 6* (DSTC 6). The results show that our method outperforms other submitted models in estimating the exact probability of the breakdown.

The remainder of this paper is organized as follows. Section 2 reviews related work of dialog breakdown detection and the dialog context embedding. Section 3 describes the architecture of the proposed model and its training process. A detailed description of the DBDC 3 tasks and provided dataset is introduced in Section 4. Section 5 presents the evaluation results and analyzes the results. Finally, we discuss future work in Section 6 and conclude in Section 7.

2. Related Works

2.1. Breakdown Detection

Error in the dialog system significantly damages the user experience. Therefore, several strategies to detect and handle miscommunication have been suggested. Walker et al. [8] forecasted whether ongoing dialog will fail with a corpus-based approach. Bohus and Rudnicky [9] estimated the system's understanding of the user's utterance with an introduction to utterance-level confidence annotation. The method combines features from au-

tomatic speech recognition (ASR), NLU, and DM to determine the confidence score. Schmitt et al. [10] proposed a scheme to model and predict the quality of interaction at arbitrary points during an interaction. This approach also used the automatically extractable features from ASR, NLU, and DM modules. Recent research [11] presented a data-driven approach using both automatically extractable features from the system logs of DM and manually annotated features.

All the previously introduced works worked on the *task-oriented dialog systems*, such as providing bus schedule information to people. Furthermore, because many of those approaches were part of the entire dialog system, they had access to the information from ASR, NLU, and DM modules. However, our system's main goal is to detect the breakdown by only using the text of the dialog. Therefore, we cannot use the confidence score for NLU slots and belief score of DM states. More thorough analysis of each utterance text is needed for our task.

2.2. Dialog Context Embedding

As we only have the dialog text data, accurately representing the context of a dialog is one of the most important parts of the system. There have been many successful works on word representations [12, 13]. These researchers have shown that word embedding can well represent the semantic meaning behind words. However, compared with the words, representing phrases and sentences has been considered a more challenging task. There is considerable work on using RNN to represent phrases and sentences [14, 15, 16, 7]. In their studies, it has been reported that representing a sentence in a dialog is especially more challenging because it is necessary to consider the context coming from the previous sentences. Even with the same text, sentences may have different meaning depending on the context of the preceding sentences. Therefore, representing the overall context of the previous dialog has been an important part of sentence representation tasks.

While most prior works on sentence embedding exploited only the text of the target sentence, there are a few works that attempted to represent a sentence in the context of the dialog. The most common approach is using a memory-like network [17] to keep a global context of the dialog. Models using this approach generally consist of two separate parts; one part represents the entire context of the dialog like the memory, and the other part embeds the target sentence based on the embedded context. Huang et al. [15] introduced a language model that makes use of both local and global context to compute a score that should be large for the actual next word. They designed a global semantic vector as a weighted average of the words in the document. Chen et al. [7] also chose a similar approach in representing sentence context to make an RNN slot tagger. They also used two separate encoders for context and the target sentence, but they added the attention distribution in representing the memory from the contextual sentence encoder. They measured the attention distribution over history utterances and used it as a weight for each memory. Our model basically follows the model presented in [7]. We also use two different encoders for the sentence and the context and had one attention layer over the RNNs so that only the relevant part of the memory should be considered in determining the breakdown of the dialog.

3. Model Description

In this section, we illustrate the proposed model's architecture. We first present a brief overview of the entire architecture, then elaborate each part in the following subsections. Our model is implemented with Pytorch and is publicly available at `https://github.com/naver/attention-dialog-embedding`.

3.1. Model Overview

Figure 1 illustrates the overall architecture of our dialog context embedding model. S_t indicates the target system utterance for which we want to estimate the probability of dialog breakdown. The rest of the dialog, from U_1 to U_t, consists of the dialog context. The system mainly consists of three parts: sentence embedding, attention masking, and distribution regression. We need to embed both target system utterance and the dialog context because our model incorporates both of them to estimate the probability of the dialog breakdown. We first embedded every sentence in the dialog using the GloVe vector and RNN's hidden state as illustrated in the representation of U_1 in Figure 1. With the sentence representations of the previous dialog, we extracted one context vector using an attention layer to mask each sentence. In the end, we used a feed-forward neural network (FFNN) to map the resulting vectors to estimate the probability of dialog breakdown.

3.2. Sentence Embedding

Our embedding of a sentence is the concatenation of two vectors: an average of the GloVe vector of all words in the sentence and the last RNN hidden state of each sentence. We used GloVe vectors of dimension 100 trained by the Twitter data. We used one from Twitter data rather than Common Crawl data because it is more closely related to the general chat domain of our task. After that, we put each word into the RNN encoder with GRU activations to produce the hidden state. With all words put through the RNN, it outputs the last hidden state of the last word, which is believed to represent the context of the input sentence.

3.3. Attention Between Sentences

After calculating representations of each sentence from a dialog, we put an attention layer to extract a context vector of the entire dialog. The attention layer outputs the weight of each sentence from past dialogs, and the context vector is computed as a weighted sum of the sentence embeddings. The intuition behind using attention layer to get a context vector is that how much a sentence is relevant to a target sentence may vary. For example, it is likely that the sentence immediately before the target sentence is more important than other previous sentences in determining if the response is a breakdown. Another example could be a redundant sentence. If a target sentence conveys contradictory information to the previous utterance, people tend to define it as a dialog breakdown. Therefore, we measured attention of each previous sentence to get its importance to the context in determining the breakdown.

3.4. Feed-Forward Neural Network

The last layer is to map all the representations to the actual distribution of breakdown labels. We used an FFNN with a dropout layer to prevent overfitting of the model. For an input vector, we concatenated a dialog context vector and the sentence vector of a target sentence. The FFNN layer then maps the input vector to three-dimensional vectors representing the breakdown label distribution.

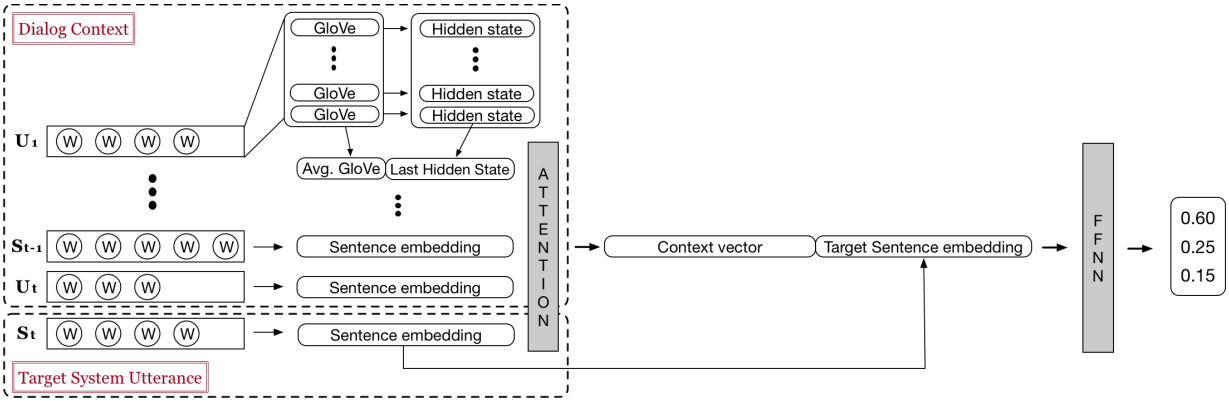

Figure 1: *The illustration of the proposed structure of dialog breakdown detector.*

3.5. Model Training and Hyper-parameters

We used the Adam optimizer [18] and learning rate decay method at the training step. The mean square error (MSE) of the output label distribution is used as the objective function of the model. The given data are randomly split into training and development data, where the proportion of data used for training is 90%. There were several hyper-parameters that we had to tune for the performance: learning rate, size of hidden states, dropout rate, and learning rate decay rate. We conducted several experiments to find the optimal hyper-parameters and used the 30th epoch's result of such parameters as the submission entry.

4. Description of DBDC 3 and Dataset

The given dataset of DBDC 3 consists of conversations between a dialog system and a user. Table 1 shows an example dialog snippet of the dataset. There are 415 dialogs in the dataset and each dialog consists of 20 or 21 utterances. For 10 system utterances in each dialog, 30 human annotators evaluated its breakdown label by three criteria: whether it causes a breakdown(*B*), possible breakdown(*PB*), or not a breakdown but a natural response for the previous user utterance (*NB*).

The number of annotators for each label gives two different kinds of target data, the distribution of three labels and one classification label determined by majority voting. As we have two types of target data, the challenge also has two independent tasks, estimating the distribution and the classification label. For example, if 20 annotators labeled a target utterance as B and five annotators each labeled the same utterance as PB and NB, then the distribution of (*B*, *PB*, *NB*) would be (0.66, 0.17, 0.17) and the classified label will be *B*. The distribution task is estimating the label distribution as close as possible to (0.66, 0.17, 0.17) and the classification task is trying to predict the given utterance's breakdown label as *B*. A detailed description of the dataset used in training and evaluation can be found in the DBDC 3 task description paper [19].

5. Evaluation and Analysis

We present the evaluation metrics and analysis of the results in this section. The method proposed in this paper corresponds to NCDS in the DBDC 3 submissions. The proposed model was officially evaluated by the DBDC 3 organizing committee on the two independent tasks, distribution estimating and classification. We first introduce the evaluation metrics of each task, then present evaluation results followed by our analysis.

Table 1: *Example dialog and annotation data*

Speaker	utterance	NB	B	PB
User1	hi there	0	0	0
System1	Hi.	29	1	0
User2	so, what's up	0	0	0
System2	Can you at least let me in?	16	11	3
User3	sure, please come in and sit down please	0	0	0
System3	I saw on the Today Show that it reduces your risk of getting cancer by fifteen percent.	9	5	16
User4	sitting down???	0	0	0
System4	Okey-dokey.	10	4	16
User5	whatsapp	0	0	0
System5	I do not know what that is. Please explain!	25	4	1
User6	a messaging app for the mobile phone	0	0	0
System6	Ok, thank you for explaining it!	15	3	12

5.1. Evaluation Metrics

For the distribution task, there were two main metrics to measure the performance of the models: *JS Divergence* (JSD) and *MSE*. Both metrics measure the distance between the predicted distribution and the target distribution, but with different distance metrics; JSD uses Jensen–Shannon Divergence, and MSE uses mean squared error. JSD is known to be slightly more accurate than MSE because it determines the similarity between two probability distributions, not only the difference [20].

In the classification task, we measure the *accuracy*, *precision*, *recall*, and *F-score* of the predicted labels. Accuracy in the task indicates the number of correctly classified labels divided by the total number of labels to be classified. Precision, recall, F-score are for the classification of the *B* labels.

5.2. Description of Submitted Entries

We submitted three entries for the DBDC 3. All three runs have the same structure explained in 3, and share the same hyper-parameter set, {learning rate: 0.0005, hidden state size: 256, dropout ratio: 0.1}. Differences between models are detailed

below.

- Run 1: Best Dialog Embedding model in terms of MSE loss
- Run 2: Best Dialog Embedding model in terms of accuracy
- Run 3: Same as Run 1 but the model has an additional 15 manually extracted features (presence of punctuation marks, length, sentiment polarity, similarity between previous sentences) concatenated at the end of the FFNN layer input

5.3. Distribution Task

The table 2 presents the distribution metrics results of the top 10 models of DBDC 3. There are two baselines presented, majority baseline and conditional random field (CRF)-based baseline. The majority baseline outputs only the most frequent dialog breakdown label in the development set with the probability of 1. The CRF-based baseline labels utterance sequences with the three breakdown labels by using CRFs.

Our first and second entries are doing the best among all models in terms of both JSD and MSE yet did not exceed the majority baseline. Although it is interesting that the majority baseline performs better than any other submitted entry, it lacks the prediction capacity. Our third entry did not perform as well as the other two entries and it indicates that the additional extracted features do not provide extra enhancement on the models' regression power.

Table 2: *Results of JS Divergence and Mean Squared Error*

Model+Run	JSD	Model+Run	MSE
Majority Baseline	0.0393	Majority Baseline	0.0224
NCDS run1	**0.0412**	**NCDS run1**	**0.0237**
RSL17BD run2	0.0412	**NCDS run2**	**0.0237**
NCDS run2	**0.0412**	RSL17BD run2	0.0241
RSL17BD run3	0.0426	RSL17BD run3	0.0250
RSL17BD run1	0.0432	RSL17BD run1	0.0254
KTH run2	0.0481	KTH run2	0.0281
NCDS run3	**0.0668**	PLECO run1	0.0415
PLECO run1	0.0714	**NCDS run3**	**0.0437**
PLECO run2	0.0774	PLECO run2	0.0448
SWPD run1	0.0807	SWPD run1	0.0471
SAM2017 run1	0.2823	SAM2017 run1	0.1441
KTH run3	0.3268	KTH run3	0.1670
CRF Baseline	0.4409	CRF Baseline	0.2185
KTH run1	0.4445	KTH run1	0.2240

5.4. Classification Task

The table 3 shows accuracy and F-score result of models. Compared with the distribution task, our three submitted entries did not perform well on the classification task. This difference in performance between the two tasks is believed to be a result of our target data used in the training process. As explained earlier, we used the MSE of output labels as our loss function for model training. The MSE loss between the predicted and target distribution can be a good training objective for the distribution task because it is the direct evaluation metric of the task. However, the MSE loss of the actual distribution does not guarantee good performance on the classification task. In the training process, we observed that the accuracy does not necessarily improve when the MSE, which was our main loss function, decreases.

After the submission, without changing the model itself, we performed an additional experiment by changing the model's training target data to be more suitable for the classification task. We changed the target label distribution to the majority voting distribution. For example, if the label distribution was (0.60, 0.25, 0.15), we changed it to (1,0,0). In this way, the system learns the result of majority voting rather than the original distribution. With the changed target label distribution, the system showed much better performance as presented in Table 3 as *NCDS POSTRUN*. It showed the highest F1 score and second highest in accuracy among the submitted models. This implies that our model also works well for the classification task, but we need to set different target data for each task.

Table 3: *Results of classification accuracy and F1 score*

Model+Run	Accuracy	Model+Run	F1(B)
KTH run2	0.4415	**NCDS POSTRUN**	**0.5395**
NCDS POSTRUN	**0.4400**	PLECO run1	0.3636
RSL17BD run2	0.4310	PLECO run2	0.3565
SWPD run1	0.4295	CRF Baseline	0.3543
CRF Baseline	0.4285	KTH run1	0.3487
RSL17BD run1	0.4265	KTH run3	0.3373
KTH run3	0.4220	Majority Baseline	0.3343
RSL17BD run3	0.4200	SWPD run1	0.3210
SAM2017 run1	0.4060	RSL17BD run2	0.3201
Majority Baseline	0.3720	**NCDS run3**	**0.3198**
NCDS run2	**0.3655**	RSL17BD run1	0.3126
NCDS run1	**0.3605**	RSL17BD run3	0.3025
NCDS run3	**0.3565**	KTH run2	0.2949
KTH run1	0.3375	SAM2017 run1	0.2413
PLECO run1	0.2950	**NCDS run2**	**0.2097**
PLECO run2	0.2900	**NCDS run1**	**0.2076**

6. Further Discussions

In our work, we used a simple sentence representation method of using static word representation and the RNN's hidden states. Throughout the experiments, we observed that the change in sentence embedding method significantly affects the overall performance of the model. Therefore, some recent researches suggesting better methods for sentence representations [21] are expected to improve the model performance, thus applying such methods in our model would be interesting future work. For example, as fine-tuning of static word representations has been shown to improve the performance of a model [22], we can try applying fine-tuning to the GloVe before averaging for the sentence embedding. Furthermore, as we observed a lot of typos and derivatives of slang from user input utterances, using character-based representation may also improve the quality of the word representations.

7. Conclusions

In this work, we proposed a deep neural network model for the dialog breakdown detection task. The model uses RNN and attention layer to embed the context of the dialog and to estimate the probability of a dialog breakdown of a given target utterance. Its performance, evaluated in DBDC 3, showed that the model significantly outperforms other methods in the distribution task. In the classification task, we conducted an additional experiment with changed target data tailored to the task. The

result showed that our model also performs well on the classification task, and setting different target data for each task is important.

8. Acknowledgments

We would like to thank the organizing committees of the DSTC and especially the DBDC organizers for their help and for kindly providing the data. This work was supported by ICT R&D program of MSIP/IITP (R7124-16-0004, Development of Intelligent Interaction Technology Based on Context Awareness and Human Intention Understanding).

9. References

[1] B. Martinovsky and D. Traum, "The error is the clue: Breakdown in human-machine interaction," UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY CA INST FOR CREATIVE TECHNOLOGIES, Tech. Rep., 2006.

[2] S. Young, "Using pomdps for dialog management," in *Spoken Language Technology Workshop, 2006. IEEE*. IEEE, 2006, pp. 8–13.

[3] J. Edlund, G. Skantze, and R. Carlson, "Higgins-a spoken dialogue system for investigating error handling techniques." in *INTERSPEECH*, 2004.

[4] A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi, "Lets go public! taking a spoken dialog system to the real world," in *in Proc. of Interspeech 2005*. Citeseer, 2005.

[5] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 136–140.

[6] Y. Shi, K. Yao, H. Chen, Y.-C. Pan, M.-Y. Hwang, and B. Peng, "Contextual spoken language understanding using recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5271–5275.

[7] Y.-N. V. Chen, D. Hakkani-Tr, G. Tur, J. Gao, and L. Deng, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding." ISCA, June 2016. [Online]. Available: https://www.microsoft.com/en-us/research/publication/contextualslu/

[8] M. Walker, I. Langkilde, J. Wright, A. Gorin, and D. Litman, "Learning to predict problematic situations in a spoken dialogue system: Experiments with how may i help you?" in *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, ser. NAACL 2000. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 210–217. [Online]. Available: http://dl.acm.org/citation.cfm?id=974305.974333

[9] D. Bohus and A. I. Rudnicky, "Integrating multiple knowledge sources for utterance-level confidence annotation in the cmu communicator spoken dialog system," Tech. Rep., November 2002. [Online]. Available: https://www.microsoft.com/en-us/research/publication/integrating-multiple-knowledge-sources-utterance-level-confidence-annotation-cmu-communicator-spoken-dialog-system/

[10] A. Schmitt, B. Schatz, and W. Minker, "Modeling and predicting quality in spoken human-computer interaction," in *Proceedings of the SIGDIAL 2011 Conference*. Association for Computational Linguistics, 2011, pp. 173–184.

[11] R. Meena, J. Lopes, G. Skantze, and J. Gustafson, "Automatic detection of miscommunication in spoken dialogue systems." in *SIGDIAL Conference*, 2015, pp. 354–363.

[12] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 384–394.

[13] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[14] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 151–161. [Online]. Available: http://dl.acm.org/citation.cfm?id=2145432.2145450

[15] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ser. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 873–882. [Online]. Available: http://dl.acm.org/citation.cfm?id=2390524.2390645

[16] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[17] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] R. Higashinaka, K. Funakoshi, M. Inaba, Y. Tsunomori, T. Takahashi, and N. Kaji, "Overview of dialogue breakdown detection challenge 3," in *Proceedings of Dialog System Technology Challenge 6 (DSTC6) Workshop*, 2017.

[20] B. Catania and L. C. Jain, *Advanced Query Processing: Volume 1 Issues and Trends*. Springer Publishing Company, Incorporated, 2014.

[21] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," *arXiv preprint arXiv:1511.08198*, 2015.

[22] F. Dernoncourt, J. Y. Lee, O. Uzuner, and P. Szolovits, "De-identification of patient notes with recurrent neural networks," *Journal of the American Medical Informatics Association*, vol. 24, no. 3, pp. 596–606, 2017.