LSTM for Dialogue Breakdown Detection: Exploration of Different Model Types and Word Embeddings

Mariya Hendriksen, Artuur Leeuwenberg, and Marie-Francine Moens

Abstract One of the principal problems of human-computer interaction is miscommunication. Occurring mainly on behalf of the dialogue system, miscommunication can lead to dialogue breakdown, i.e., a point when the dialogue cannot be continued. Detecting breakdown can facilitate its prevention or recovery after breakdown occurred. In the paper, we propose a multinomial sequence classifier for dialogue breakdown detection. We explore several LSTM models each different in terms of model type and word embedding models they use. We select our best performing model and compare it with the performance of the best model and with the majority baseline from the previous challenge. We conclude that our detector outperforms the baselines during the offline testing.

1 Introduction

The importance of spoken dialogue systems has been steadily increasing over the years. Some of the reasons for such a popularity raise include their ability to provide instant twenty-four-hour service. and applicability across different domains such as website assistance, education, customer service, e-commerce, and entertainment.

Naturally, the usefulness of a dialogue system largely depends on its ability to interact with users. One of the major obstacles on the way to the goal is miscommunication. We define miscommunication as a situation when a dialogue system gives

Mariya Hendriksen
University of Amsterdam, Amsterdam, The Netherlands (work performed while at KU Leuven), e-mail: m.hendriksen@uva.nl

Artuur Leeuwenberg
KU Leuven, Heverlee, Belgium, e-mail: tuur.leeuwenberg@kuleuven.be

Marie-Francine Moens
KU Leuven, Heverlee, Belgium, e-mail: sien.moens@kuleuven.be

a user an inappropriate reply. In other words, we assume that miscommunication occurs on the behalf of a system.

Miscommunication can lead to dialogue breakdown, i.e., a point in dialogue when the interaction is interrupted with or without completion of the performed task [13]. In this perspective, a model capable of detecting dialogue breakdown points can enhance the quality of human-computer interaction. For instance, such a detector could help to avoid system responses which cause a breakdown or identify breakdowns after they occur and launch the procedures necessary to get out of the breakdown situation.

The rest of the paper is organized as follows: Section 2 describes the task of dialogue breakdown detection and the dataset which was used for model training. In the Section 3, we discuss the related work on dialogue breakdown detection models, applications of dialogue breakdown detection detectors, the taxonomy of errors causing dialogue breakdown and the alternative methods for system response assessment. We describe the proposed model in the Section 4 and explain the experiment setting in Section 5. The discussion of the results is presented in Section 6. In Section 7, we draw conclusions and suggest directions for future work.

2 Task description

We address the task of dialogue breakdown detection. The task was introduced as a challenge in [17]. Since the introductions, three challenges were held [5], [6]. We submit the model for the fourth challenge [7].

In particular, we aim to develop a system to predict whether a given system utterance causes a breakdown. The prediction is to be based both on the current utterance and on the dialogue history. Each system response is to be marked with one of the three labels:

- NB – not a breakdown: it is possible to continue the dialogue smoothly.
- PB – possible breakdown: it is difficult to continue the dialogue smoothly.
- B – breakdown: it is difficult to continue the dialogue.

For model development, we utilize the training dataset provided by the challenge organizers. The set consists out of 211 dialogues. Each dialogue in the set has a length of 20 or 21 utterances: 25 dialogues have length 20, whereas 186 dialogues have length 21.

Every system response is labelled by fifteen annotators. Hence, for every system utterance, the model predicts the probability distribution of labels and assign the label with the highest probability to given system response.

There are two primary types of metrics for performance evaluation: distribution-related and classification-related. Following the findings presented in [20], we focus on mean squared error MSE(NB,PB,B) as the primary metric.

3 Literature Review

In this paper, we describe the model for dialogue breakdown detection. Related work falls in the following areas: (1) existing dialogue breakdown detection models, (2) applications for dialogue breakdown detection systems, (3) analysis of errors causing dialogue breakdown, and (4) the alternative techniques for dialogue system response assessment.

There are exist several models for dialogue breakdown detection. The list includes Conditional Random Fields model which is used as a baseline in the challenge [7], Extremely Randomized Trees [11], Maximum Entropy model [6], Support Vector Machines [12], Memory Networks [9] and Recurrent Neural Networks (RNN) [14], in particular long short-term memory networks (LSTM)[12], [21]. Besides, some systems feature attention modules as part of their architecture [14], [9].

Dialogue breakdown detection can be used to re-rank responses of a chat-oriented dialogue system. In [10], the authors suggest three re-ranking approaches: classification, regression, and probability-based approach. The classification technique was based on the classification of all the possible system responses. The regression method implied the application of linear regression with the probability distribution of breakdown labels and response scores as a feature set. In the case of the probability-based approach, the non-breakdown probability was used for re-ranking.

Other application examples include [18] where the author applies dialogue breakdown detection system for selecting tweets that can be used as responses of a chat-oriented dialogue system.

Another direction of research in the field is to investigate the errors causing a breakdown in chat-oriented dialogue systems. This is done in [4] where researchers present a taxonomy of this type of errors. Inspired by the Gricean maxims[1], the authors define utterance-level, environmental-level, and cooperativeness error. The breakdown detector based on the taxonomy of errors is presented in [8].

Breakdown detection is not the only way to evaluate chatbot responses. For instance, [22] offer a similar technique for assessment of chatbot responses in non-task-oriented dialogues. In particular, they suggest measuring the appropriateness of utterances and customer satisfaction.

4 Proposed Model

In this section, we describe the proposed dialogue breakdown detector as well as the motivation behind the selected architecture and components.

[1] the principles of effective communication in standard social setting [3]

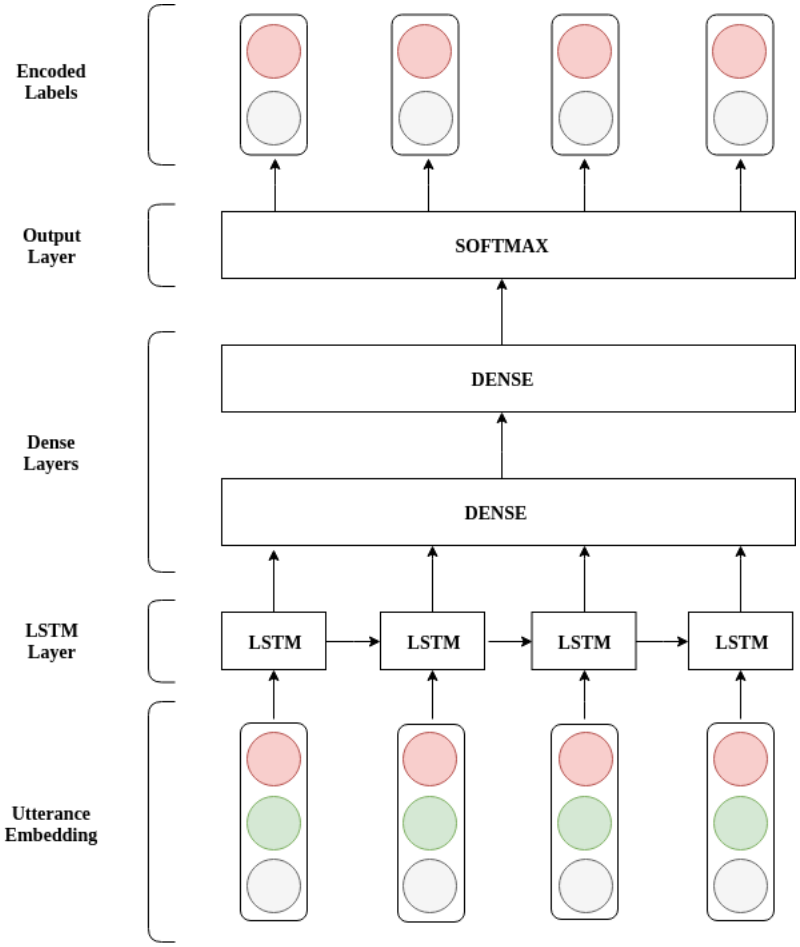

Fig. 1 Proposed model architecture: input layer, LSTM layer, two dense layers and a softmax output layer. The model takes dialogue representation as an input and assigns a label for every utterance.

4.1 Model Type

After considering the existing dialogue breakdown detectors and their performances, we chose LSTM for their ability to process sequential data and handle long-term dependencies. Several LSTM models are considered: vanilla LSTM, stacked LSTM, and bidirectional LSTM.

4.2 Word Embedding Model

Another important aspect to consider when working with textual data is text representation. For this reason, we considered several word embedding types:

word2vec Google News

The first type of the word representations we use is the word2vec vectors pretrained on Google News corpora[2]. We use those vectors because they were featured in the LSTM model which demonstrated the best performance in terms of accuracy during DBDC3 [12]. The vectors were produced by a bag-of-words model (BoW) trained with negative sampling with window size 5. Each word is represented with an embedding of size 300.

GloVe Twitter

The second type of word representation we use is GloVe trained on Twitter data[3]. We use this embedding type because of the proximity of the Twitter domain to the task domain[14]. The vectors were obtained by training on 2 billions of tweets with representations for 27 billion tokens and a vocabulary of 1,2 million. The vectors are presented in 25d, 50d, 100d and 200d, we decided to use 200d vectors. The words are uncased.

GloVe Common Crawl

The third type of word embedding model is GloVe Common Crawl[4]. It contains representations for 840 billion tokens with the vocabulary of 2.2 million. The words are cased and the dimensionality of the vectors is 300d. We use this embedding type because, unlike the GloVe Twitter, it is domain-independent.

4.3 Model Architecture

The proposed model takes dialogue representations as input. Dialogue representations are composed out of utterance representations which, in turn, are created out

[2] Google News 100B 3M words, URL: https://github.com/3Top/ word2vec-api, last checked on 07-04-2019.

[3] GloVe: Twitter, URL: https://nlp.stanford.edu/data/glove.twitter.27B.zip, last checked on 27-03-2019.

[4] GloVe: Common Crawl, URL: https://nlp.stanford.edu/data/glove.840B.300d.zip, last checked on 27-03-2019.

of word representations. During the training phase, the training loss is computed and models parameters are optimized.

4.3.1 Input Representation

Word representations are acquired with pretrained word embedding vectors. We represent all the out of vocabulary (OOV) tokens with the token *unk*. Each sentence embedding is represented as the average of the token embeddings that comprise the sentence. Consequently, each utterance embedding has the same dimensionality as the word embedding. Dialogue is represented as a sequence of user and system utterances. We pad dialogues to ensure that each dialogue has a length of 21 utterances.

4.3.2 Target Representation

Each system response in a dialogue is labelled with *B* (breakdown), *PB* (possible breakdown) or *NB* (not a breakdown) label. Besides, we introduce label *U* to mark user utterance. The labels are represented with one-hot encoding. Therefore, each sequence of dialogue labels is represented as 21×4 matrix.

4.3.3 Loss

In the given task, for each utterance in the dialogue, the network predicts a probability for each of the labels and compares it with the ground truth. Hence, the model should use a loss function that would compare the labels probability distribution with the ground truth and penalize incorrect label prediction. One of the suitable objective function for this task is cross-entropy function[1]. The function measures the difference between two probability distributions. The function is computed as follows:

$$H(y,\hat{y}) = -\sum_{i=1} y_i log(\hat{y}_i) \tag{1}$$

where y is the ground truth label and $\hat{y}$ is the predicted label.

4.3.4 Optimization

We optimize model with mini-batch gradient descent of a batch size of 4. We use Root Mean Square Propagation (RMSProp)[19] which is the extension of Resilient Backpropagation (Rprop) learning[16]. RMSProp combines the robustness of Rprop, the efficiency of mini-batches and the effective averaging of gradients over mini-batches.

5 Experiment

In this section, we discuss dataset preprocessing, define the baselines, explain the training pipeline and present the results.

5.1 Data preprocessing

Before starting preprocessing, we have to make several decisions:

- Do we keep user turns? Following the findings described in [12], we decided to keep the user turns as they enhance detector performance.
- How do we feed user turns to the model? Initially, we considered concatenating user turns with the corresponding system response into exchange pairs. Such an approach would allow avoiding the introduction of an extra label. However, results shown in [21] testify that such concatenation decreases model performance. Hence, we resolve to mark each user utterance with an extra label U and feed them to the models as a separate utterance.

After making the decisions, we can start dataset preprocessing. Since we use three different types of embedding models, we prepare one dataset for each of the types. We do it in order to take all the particularities of the embedding models into account. Two major preprocessing steps were applied to all three datasets: tokenization and replacement of apostrophe contractions.

In general, all the datasets are tokenized with `TweetTokenizer`[5]. This tokenizer is a part of `casual` submodule of `nltk.tokenize` package, it was selected because the domain of its primary use is closely related to the domain of the task. In particular, `TweetTokenize` is able to handle emoticons the dataset contain.

In addition to tokenization, extra rules are applied to common apostrophes contractions. For example, the contraction *that's* is transformed to *that is*.

Besides the mentioned preprocessing steps, we remove punctuation signs from the dataset for word2vec Google News vectors because the model did not know any punctuation signs. Additionally, we lowercase all the words in the dataset for GloVe Twitter because the pretrained word vectors are uncased.

As mentioned in the section 2, dialogue length varies from 20 to 21 utterances per dialogue. In such a case, we can either truncate or pad the dialogues. Since the first approach implies a loss of certain parts of data, we implement padding as a more suitable option.

[5] NLTK 3.4 documentation, URL: http://www.nltk.org/api/nltk.tokenize.html, last checked on 27-03-2019.

5.2 Model Training

The model hyperparameters were determined by a grid search. The model was trained for a maximum of 100 epochs. To prevent overfitting, we employ two regularization techniques: early stopping and dropout (both standard dropout and recurrent dropout).

5.3 Baselines

We compare the performance of our model with the two baselines defined during DBDC3. First is the majority baseline, the second model is the best performing model of DBDC3 which was an attention-based detector [14].

5.4 Model Selection

The next step is to select the best performing model out of the nine models we experiment with.

In order to investigate which type of LSTM produces the best performance, we compare the metrics results. We do this by calculating the average metric score for each $Model \times Metric$ pair across three embedding types. The results are presented in the Table 1. Overall, it can be concluded that, given that all the metrics have equal importance, the best performance is obtained by vanilla LSTM, stacked LSTM is the second best, and Bi-LSTM is the worst.

Table 1 Average metric scores for every LSTM type

LSTM type	MSE(NB,PB,B)
Vanilla LSTM	0.0213
Stacked LSTM	0.0224
Bi-LSTM	0.0222

Next, we turn to the investigation of the relationship between model performance and its embedding type. In analogy with the above-mentioned idea, we calculate an average performance score for each metric. The results presented in the Table 2, allow to conclude that GloVe Common Crawl demonstrate the best performance, the GloVe Twitter being the second best, the word2vec Google News is the worst.

Table 2 Average metric scores for every word embedding type

Embedding type	MSE(NB,PB,B)
word2vec Google News	0.0351
GloVe Twitter	0.0301
GloVe Common Crawl	0.0213

6 Discussion & Implications

After running experiments with the created models, we concluded that the vanilla LSTM with GloVe Common Crawl embedding demonstrates the best performance. For this reason, we compare it with the selected baselines. The comparison is given in the Table 3. As can be seen, our model outperforms the baselines.

Table 3 Comparison of the created model with the baseline models.

Model	MSE(NB,PB,B)
Majority baseline	0.0224
NCDS	0.0237
Proposed model	0.0213

6.1 Analysis of Error Patterns

Comprehension of patterns in the type of errors that the best models make could provide additional insight into their improvement.

During the experiments, we found out that embedding type impacts model performance. In particular, we discovered that there is a negative correlation between the proportion of OOV and model performance. Hence, we investigate the type of OOV tokens. In general, GloVe pretrained on Common Crawl does not know 245 tokens in the dataset (or 115 unique tokens).

Emoticons

One of the significant parts of OOV tokens were emoticons. The emoticons are the strong cues of how the user feels and hence can help to understand when the conversation goes in the wrong direction. Therefore, they should be taken into account. The model's vocabulary includes basic emoticons such as ☺ or ☹ but not more complex ones such as ☹ or 😎. Besides, some dialogues featured emoticons

represented as capitalized descriptions in square brackets (e.g., [SMILING FACE WITH SUNGLASSES], [SPEAK-NO-EVIL MONKEY]). As a possible solution to the problem, it might be helpful to replace the complex emoticons with their basic counterparts to facilitate the model's understanding by reducing variance.

Apostrophe Contractions

Many of the tokens were not recognized because they represented apostrophe contractions. The examples included both straightforward cases like *isn't* (*is not*) and ambiguous situations such as *he's* which depending on context can be interpreted either like *he has* or *he is*. In general, the problem can be resolved by the introduction of extra contraction replacement rules. In the case of ambiguous situations, it might be better to apply a disambiguation procedure first.

Abbreviations

Another significant group of unknown tokens includes abbreviations, e.g., *ConvAI (conversational Artificial Intelligence)*. The issue can be addressed by abbreviation expansion. For instance, we could create a dictionary containing the most common abbreviations.

Misspellings

Some other words are not recognized because of misspelling. The most common type of mistakes is skipping white-space and thereby merging words. E.g., 'questionWho' and 'wait.Where'. Splitting these words while preprocessing the datasets can help to address the issue. Moreover, there were standard spelling mistakes such as 'seee' or 'tommorow'. Such standard mistakes can be resolved by adding a spellchecker.

7 Conclusion & Future Work

In this paper, we present a model for dialogue breakdown detector. Offline testing demonstrates that our model outperforms the best-performing model from DBDC3. Additionally, we investigate how model architecture and word embedding model influence detector performance.

Future work will focus on further exploration of model architectures and word embeddings. In particular, it would be interesting to explore different types of attention mechanisms and investigate such embedding models such as BERT [2] and

EMLo [15]. Another direction of future work includes dataset expansion, specifically in terms of different languages and modalities.

References

[1] De Boer PT, Kroese DP, Mannor S, Rubinstein RY (2005) A tutorial on the cross-entropy method. Annals of operations research 134(1):19–67

[2] Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805

[3] Grice HP (1975) Logic and conversation. 1975 pp 41–58

[4] Higashinaka R, Funakoshi K, Araki M, Tsukahara H, Kobayashi Y, Mizukami M (2015) Towards taxonomy of errors in chat-oriented dialogue systems. In: Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pp 87–95

[5] Higashinaka R, Funakoshi K, Inaba M, Arase Y, Tsunomori Y (2016) The dialogue breakdown detection challenge 2. Proceedings of SIG-SLUD

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

[7] Higashinaka R, D'Haro LF, Shawar BA, Banchs R, Funakoshi K, Inaba M, Tsunomori Y, Takahashi T, ao Sedoc J (2019) Overview of the dialogue breakdown detection challenge 4. In: Proc. WOCHAT

[8] Horii T, Araki M (2015) A breakdown detection method based on taxonomy of errors in chat-oriented dialogue (in Japanese)

[9] Iki T, Saito A (2017) End-to-end character-level dialogue breakdown detection with external memory models

[10] Inaba M, Takahashi K (2018) Improving the performance of chat-oriented dialogue systems via dialogue breakdown detection

[11] Kato S, Sakai T (????) RSL17BD at DBDC3: Computing similarities based on term frequency and word embedding vectors. In: Proc. of DSTC6, year=2017

[12] Lopes J (2017) How generic can dialogue breakdown detection be? the KTH entry to DBDC3

[13] Martinovsky B, Traum D (2006) The error is the clue: Breakdown in human-machine interaction. Tech. rep., Institute for Creative Technologies, University of Southern California

[14] Park C, Kim K, Kim S (2017) Attention-based dialog embedding for dialog breakdown detection. In: Dialog System Technology Challenges Workshop (DSTC6)

[15] Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proc. of NAACL

[16] Riedmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Neural Networks, 1993., IEEE International Conference on, IEEE, pp 586–591
[17] Ryuichiro H, Funakoshi K, Kobayashi Y, Michimasa I (2016) The dialogue breakdown detection challenge: Task description, datasets, and evaluation metrics.
[18] Sugiyama H (2016) Utterance selection based on sentence similarities and dialogue breakdown detection on NTCIR-12 STC task. In: Proc. of NTCIR
[19] Tieleman T, Hinton G (2012) Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, URL https://www.youtube.com/watch?v=ZgLyc7CYlWw
[20] Tsunomori Y, Higashinaka R, Takahashi T, Inaba M (2018) Evaluating dialogue breakdown detection in chat-oriented dialogue systems. In SEMDIAL
[21] Xie Z, Ling G (2017) Dialogue breakdown detection using hierarchical bidirectional lstms. In: Proceedings of the Dialog System Technology Challenges Workshop (DSTC6)
[22] Zeng Z, Luo C, Shang L, Li H, Sakai T (2017) Test collections and measures for evaluating customer-helpdesk dialogues. Proceedings of EVIA