BETOLD:
A Task-Oriented Dialog Dataset for Breakdown Detection

Silvia Terragni, Bruna Guedes, Andre Manso,
Modestas Filipavicius, Nghia Khau and **Roland Mathis**
Telepathy Labs GmbH
Zürich, Switzerland
{firstname.lastname}@telepathy.ai

Abstract

Task-Oriented Dialog (TOD) systems often suffer from dialog breakdowns - situations in which users cannot or do not want to proceed with the conversation. Ideally TOD systems should be able to detect dialog breakdowns to prevent users from quitting a conversation and to encourage them to interact with the system again. In this paper, we present BETOLD, a privacy-preserving dataset for breakdown detection. The dataset consists of user and system turns represented by intents and entity annotations, derived from NLU and NLG dialog manager components. We also propose an attention-based model that detects potential breakdowns using these annotations, instead of the utterances' text. This approach achieves a comparable performance to the corresponding utterance-only model, while ensuring data privacy.

1 Introduction

Task-Oriented Dialog (TOD) systems (Zhang et al., 2020) enable users to complete specific tasks, such as booking a reservation at a restaurant. Unlike open-domain dialog systems (Huang et al., 2020), where the aim is to maximize user engagement, in TOD systems it is crucial to optimally assist a user to fulfill the task at hand. Detecting dialog breakdowns due to miscommunications potentially paves the way to intervene and rescue the dialog, improve customer satisfaction and motivate the user to continue interacting with the system (Brandtzæg and Følstad, 2018).

A dialog breakdown is often defined as a point in the dialog where the user gives up the conversation without completing the task, often due to not understanding the intended meaning of user's utterance (Martinovski and Traum, 2003; Higashinaka et al., 2015). If conversational system engineers can understand when and why a conversation is likely to break down, they can build systems that prevent broken dialogs, or design conversational breakdown recovery strategies (Benner et al., 2021).

The Dialog Breakdown Detection Challenge (DBDC) has motivated the academic community's interest in the breakdown detection problem, which is the goal of predicting the occurrence of a breakdown at some point in the conversation (Higashinaka et al., 2016). This challenge also came with the release of English and Japanese datasets for addressing this task. Despite the great value of these proposed datasets, they only provide the sequence of user and system utterances. Utterances are indeed useful to detect a breakdown, however, in certain contexts, especially in industry, a stakeholder may decide not to share and release the texts for privacy preserving purposes (Xu et al., 2021).

The utterances produced by a user during a task-oriented conversation often contain privacy-sensitive information. Let us consider a dialog system in a company that handles issues relating to human resources as an example. The system may receive data regarding an employee's health status or compensation, i.e., data that a company is unwilling to share. In this paper, we demonstrate that even without access to the text of the conversation, it is still possible to identify a breakdown. In fact, traditional dialog systems often provide synthetic annotation of the user and system utterances as the intents and entities, originating from the Natural Language Understanding (NLU) and Natural Language Generation (NLG) components (Wahde and Virgolin, 2022). In particular, the NLU component classifies the user utterances into intents (`book_appointment`) and extracts entities (`user_name="John Smith"`). The NLG component consists of a closed set of possible system utterances (each defined by a unique intent), often parameterized by or supplemented with entities (e.g. "`restaurant_name` is open on `day_of_the_week`", where `restaurant_name` and `day_of_the_week`

Proceedings of the 2nd Workshop on When Creative AI Meets Conversational AI (CAI2), COLING 2022, pages 23–34
October 12–17, 2022.

		BETOLD	DBDC	DSTC2
Annotation Features	Task-oriented dialogs	Yes	partially	Yes
	Task domain	phone repair	mixed	restaurant, tourist info
	Annotation by	automatic	human	human
	Has intents	Yes	Yes	Yes
	Has entities	Yes	No	Yes
Breakdown Features	BD* annotated by	system	human	human
	BD label	Yes	Yes	NO**
	BD defined as	caller hangup & transfer request	nonsensical system reply	inferred from intent
	Number of classes	2	3	2
	Partial BD label	No	Yes	No
	BD initiated by caller	Yes	No	Yes
System Abilities	ASR and TTS	Yes	No	No
Statistics	# Dialogs	13,524	615	2,115
	# utterances per dialog (median)	10	20	12

Table 1: Comparison of our BETOLD and other two publicly available datasets used for breakdown detection. *BD stands for "breakdown". **User's intent "restart" could be used as a substitute for breakdown.

are two entities).

In our study, we propose a simple yet effective way to *automatically* create a new breakdown detection dataset, given an existing TOD system. In particular, we consider a phone-call scenario, in which a customer talks to a digital assistant to schedule a service appointment for their mobile phone. In our experience, two central events indicating caller's frustration with the current dialog state and projected outcomes are: a) hang-ups, b) requests to talk to a human agent. Therefore, we consider caller hang-ups and transfer to human requests as dialog breakdowns.

We also release a novel task-oriented dialog dataset BETOLD (Breakdown Expectation for Task-Oriented Long Dialogues) with the proposed annotation schema. The dataset contains real human-agent conversations, between customers and our modular dialog system. The system automatically annotates the user utterances with NLU intents and entities, and generates appropriate NLG responses which contain NLG intents and accompanying entities.

Finally, we propose an attention-based model, capable of taking these features into account. Our results show that, instead of relying only on the word tokens of the utterances, the use of NLU and NLG intents is sufficient to confidently predict a breakdown in a task-oriented conversation, therefore reaching satisfactory results while guaranteeing the privacy of the data.

2 Related Work

2.1 Datasets for Breakdown Detection

Only few task-oriented dialog breakdown datasets are openly available. We report the most relevant ones in Table 1. Overall, they are small and human-annotated, and with varying definitions of dialog breakdown.

The Dialogue Breakdown Detection Challenge (DBDC) offers a small dataset with 615 English conversations, annotated with system utterances that cause dialog breakdown (Higashinaka et al., 2016). It contains three classes: breakdown, possible breakdown, and no breakdown. However, no intents and entities annotations are available. Additionally, opposite to BETOLD, most of conversations in DBDC are open-domain. Another open-source alternative is the Dialog State Tracking Challenge (DSTC2) dataset, which unfortunately lacks breakdown annotations (Williams et al., 2014). However, 7 out of 2,115 conversations have an intent "Restart" which. If more prevalent, this could be used as a substitute for breakdown.

Similar to our work, Gorin et al. (1997) annotated 10,000 TODs between customers and agents. Subsequent experiments with the same dataset identified user hangup and requests to transfer to human agent as a specific learning problem, as proposed in our work (Walker et al., 2000). However, the dataset is not publicly available. Further examples of closed-source studies feature predicting in-

teraction quality from automatically extracted features (Schmitt et al., 2011) or manually-annotated features by AMT workers (Meena et al., 2015). These studies used publicly available un-annotated datasets, however the authors did not release their annotations.

2.2 Models for Breakdown Detection

The initial approaches to the breakdown identification problem focused on extracting features that can characterize a breakdown (Schmitt et al., 2011; Meena et al., 2015; Walker et al., 2000). Textual features can be used to compute similarity between the system utterance and user utterance (Meena et al., 2015), or detecting emotions from utterances and using them as indicators of a dialog breakdown (Schmitt et al., 2011; Matsumoto et al., 2022). Alternatively, dialog manager-generated tabular features such as repetitions, negations, or utterance counts can be considered as well (Walker et al., 2000; Schmitt et al., 2011).

With advent of the DBDC challenge, novel approaches have been proposed, yet limited by the available dataset features. These approaches rely solely on the textual utterances and the number of turns. They explore both traditional (Kato and Sakai, 2017; Sugiyama, 2021) and deep learning-based models (Hendriksen et al., 2021; Wang et al., 2021; Park et al., 2017). Given the sequential nature of dialogs, many models exploit sequential architectures, such as RNN or LSTM (Hendriksen et al., 2021; Wang et al., 2021; Shin et al., 2019; Lee et al., 2020), or they use an attention mechanism to determine the utterance embeddings on which to focus the attention (Park et al., 2017). Considering the encoding of the text, different approaches have been investigated: from the use of static word embeddings such as Glove and Word2Vec (Hendriksen et al., 2021) to contextualized embeddings, e.g., BERT (Sugiyama, 2021; Shin et al., 2019).

3 A Privacy-Preserving Dataset for Breakdown Detection

3.1 Dataset Creation

The considered conversations are based on real conversations between a human and a task-oriented dialog system, with the goal of scheduling or canceling an appointment. The user interacts with the system over the phone. We considered four scenarios in which a phone call could end:

- **successful calls**: the caller hangs up after the caller's goal has been satisfied (e.g. the agent has successfully scheduled a booking);
- **agent-initiated forwarded calls**: the agent takes the decision to forward the call (e.g. technical problems with the system);
- **user-initiated forwarded calls**: the user explicitly requests to talk to an operator, identified by the *transfer_to_human* intent. (This may include NLU misclassifications);
- **user-initiated caller hangup**: all the remaining calls.

A conversational engineer strives to avoid both user-initiated forward calls and user-initiated hangups. However, observing the data, we can see that the caller's behavior changes depending on the number of turns. There is no way to prevent the caller from hanging up in the initial turns: This is the case when a user does not want to speak to a digital assistant at all. We report in Figure 1 the distribution of the different types of phone calls over the number of turns.

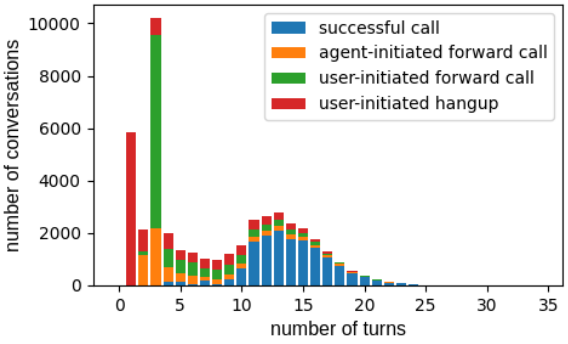

Figure 1: Distribution of the different types of phone calls over the number of turns on a sample of 45,385 conversations.

Given these considerations, we focus on user-initiated forward calls and hang-ups occurring late in the conversation. We will refer to these calls as to **LUHF**s (**L**ate **U**ser-initiated **H**ang-ups or **F**orward calls), a particular class of dialog breakdowns.

LUHFs are the types of calls that we aim to predict (positive examples). We consider as *late* conversations all the calls that reach at least the 8th turn. On the other hand, late successful calls are the negative examples of the dataset. In particular, we sample successful calls and then truncate the conversations at a random point (still, after the

Utterance	Intent	Entities
S: Are you a registered customer?	ask_if_current_client	
H: Uhm no	negate	
S: Can I book your service appointment under the phone number ending in <1234>?	confirm_phone_number	user_phone_suffix
H: Yeah that's correct	confirm	
S: What is the brand, model and year of your phone device?	new_user_profile_brand_model_year	
H: It's a <phonepink> why 100 <2022>	inform	brand_device, year
S: What is the model of phone device?	ask_device_model	
H: It is <y100>	inform	model_device
S: What is the battery health percentage of your phone device?	ask_for_battery_health	
H: <zero>	inform	numeric
S: What is your first name?	ask_first_name	ask_last_name
H: My name's <John>	inform	client_name
S: Great, what is your last name?	ask_last_name	
H: <Smith>	inform	client_name
S: What service does your phone device need?	ask_desired_service	
H: Uhm <battery replacement>	inform	type_of_repair

Table 2: Extract of a conversation between the system and a human. The entity values are enclosed by angular parentheses in the utterances.

"late"-call threshold). The user-initiated forward calls are also truncated from the point when the caller asks to be transferred. The ratio successful calls/LUHFs is 2:1. As a result, the dataset contains 13,524 calls (4,508 LUHFs and 9,016 not LUHFs).

Let us notice that the dataset contains noisy data because it is automatically annotated. Conversations may lead to a user-initiated forward, for example, even if there was no indication of user frustration. Similarly, a user may become irritated with the conversation but still decide to end the call. Moreover, it is worth noticing that the provided annotation is not at the utterance-level. Instead, a LUHF/not LUHF annotation refers to the overall conversation. In other words, if a conversation is a LUHF, we are not aware of at which point of the conversation a breakdown occurred. These elements make the predictions more challenging.

3.2 Dataset Features

The dialog system is composed of different modular components, including an NLU and an NLG component. The NLU provides annotations to the user's utterances, i.e. the *intent* and the *entities*, recognized by an intent classifier and a named-entity recognition system respectively. The NLG also provides the name of the *intent* and the *entities*, uttered by the system. The NLG intents are always different from the NLU intents. On the other hand, the NLG and NLU may have some entities in common.

3.3 Dataset Anonymization for Privacy Preservation

We anonymize the original data to protect the privacy of the original conversation content. In particular, we remove natural-language text and entity values. Keeping only the intent and entity annotations guarantees the privacy of the data.

We report an example of a fictitious conversation in Table 2, reporting the utterances exchanged between the system and a human. The detected entities are enclosed by angular parentheses in the utterances. We can notice that the caller releases sensitive information, such as the name and phone number. The entities and the intents are a synthetic way to represent the utterances, and therefore to substitute the utterances with these annotations is a valid way to proceed. One may argue that, in order to not lose much information, it could be possible to keep the text and remove only the entity values. This approach would work fine only with a perfect NLU that is able to recognize all the entities in the text. As we can see from Table 2, the entity `model_device` is not detected the first time. Moreover, the caller may reveal other types of sensitive information that an NLU is not supposed to detect. Since the NLU is prone to these errors and may not detect an entity in the text, it is

indeed safer to remove all the textual information to preserve the privacy of the data.

# labels	# not LUHFs	9016
	# LUHFs	4508
	# LUHs	2477
	# LUFs	2765
# turns	min	8
	max	34
	avg	10
# NLG and NLU unique intents		91
# NLG and NLU unique entities		41

Table 3: BETOLD dataset statistics.

Table 3 reports the main dataset statistics. The resulting privacy-preserving dataset, named BETOLD (Breakdown Expectation for Task-Oriented Long Dialogues), is available at the following link: https://github.com/telepathylabsai/BETOLD_dataset.

4 An Attention-based Model for LUHF Detection

The task in BETOLD dataset is to classify if a conversation between a human and the system is a LUHF or not. In this setting, it is fundamental to keep track of what happened in the past, represented by the dialog history. We therefore investigate an attention-based architecture (Vaswani et al., 2017), that has proved to perform well in several dialog-related tasks (Qin et al., 2021; Colombo et al., 2020; Zhao and Kawahara, 2019; Hori et al., 2016), including dialog breakdown detection (Park et al., 2017).

4.1 Model Architecture

A conversation between a human and the system can be represented as a sequence x of n tuples, where each element of a tuple represents a different feature. Here, the features include the caller name (either NLG or NLU), the intents, and the entities. The sets of possible values (also called vocabularies) of each feature are represented by C, I, and E for the caller names, intents and entities respectively. The sequence x is then:

$$x = (c_1, i_1, e_1), (c_2, i_2, e_2), \ldots, (c_n, i_n, e_n) \quad (1)$$

where (c_j, i_j, e_j) is a tuple composed of a caller name $c_j \in C$, an intent $i_j \in I$ and an entity set $e_j \in \mathcal{P}(E)$ and $j = 1, \ldots, n$. We refer to the entities as *entity set*, because for each tuple, we can have zero, one or more entities. The vocabulary sets C, I and E have different dimensions, to which we add an *unknown* symbol for unseen elements and a *padding* symbol.

For each of these features, the model learns embedded representations of dimension m. These vector representations are then summed up, obtaining a m-dimensional vector, which is then passed through a positional encoding layer, to keep track of the order of each element of the sequence. The resulting sequence of m-dimensional vectors is used as input to the Transformer encoder. The Transformer encoder is a stack of t encoder layers of l dimensionality. The output sequence of the Transformer encoder is averaged and the resulting vector passes through a sequence of linear layers. Finally, we apply a sigmoid function to the last layer to get the predicted score (LUHF or not LUHF). We use a weighted Binary Cross Entropy (BCE) loss to optimize.

Figure 2 shows a sketch of the proposed architecture. Each type of feature is represented in a different color.

5 Experimental Setting

5.1 Text-only Baseline

Our goal is to demonstrate that a text-free model can achieve comparable performance to a text-based model. As a consequence, we consider the text-only model to be our baseline model. The available text is represented by the user and system utterances. We use Sentence-BERT (Reimers and Gurevych, 2019) to generate a contextualized sentence representation for each utterance.

We use the same type of architecture as the proposed model to ensure a fair comparison, except that the model's input is different, i.e. dense vector representations of text. We call this model `TEXT`.

5.2 Models

We compare the text-only baseline `TEXT` with different variants of the proposed model. To identify the single contribution of the entities and intents, we consider two variants of the model, namely `INT` and `ENT`, as the models that rely solely on intents or entities respectively. We then consider a model that combines all the features (intents, entities, and caller type (NLU or NLG)), referred to as `IEC`. The implementation of the models is available at

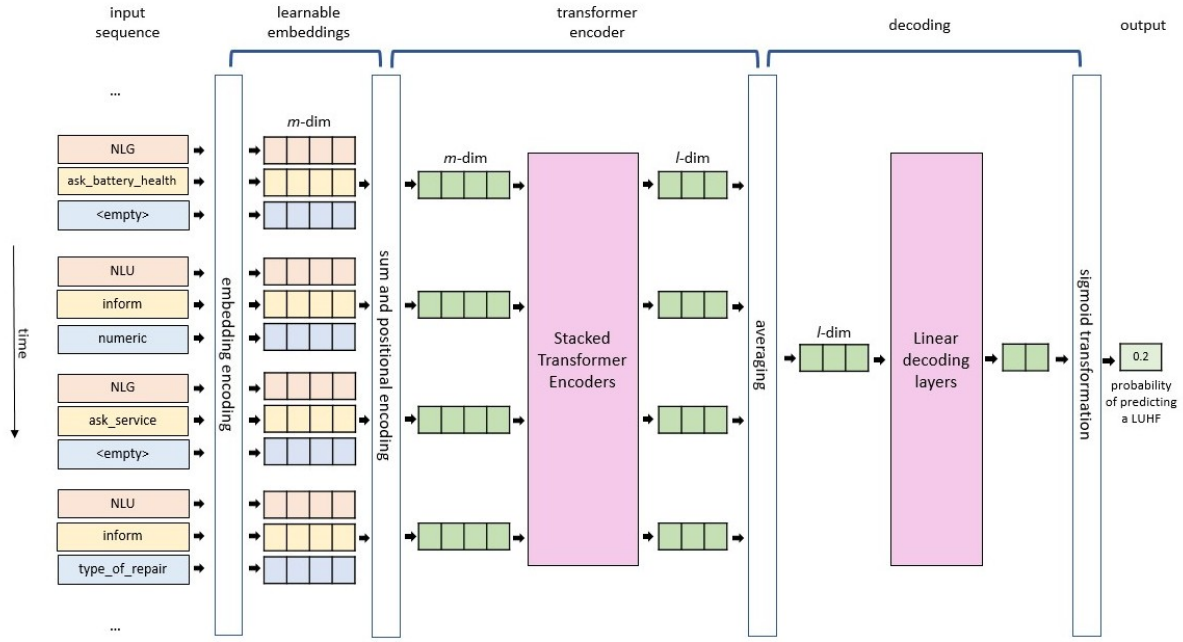

Figure 2: Sketch of the model architecture. To train the LUHF/not LUHF classifier multiple features are embedded and summed before a transformer encoder block followed by linear layers. The represented features are callers (orange, possible callers are "NLG" and "NLU"), intents (yellow, 91 intent names from NLU and NLG), and entities (blue, 41 entity names from NLU and NLG).

the following link: https://github.com/telepathylabsai/dialog_breakdown_detection.

5.3 Hyperparameter Setting

For running the experiments, we split the dataset into three parts: 80% for training, 10% for testing, and 10% for validation. We run all models for 50 epochs and then select the best model based on the validation set. We augment the training data by adding successful calls. In particular, we truncate the successful calls at random points and use them to expand the training data. We added 30% more successful calls to the overall training data. We do not perform data augmentation for the LUHFs because the LUHF annotation refers to the overall conversation and we have no hints about at which point in the conversation something went wrong.

We use grid search to determine the optimal hyperparameter configuration of the models. In particular, since we are more interested in the prediction of a LUHF rather than the prediction of a not LUHF, we select the optimal configuration based on the F1 score of the LUHF class. For the text-only model `TEXT`, we use the `all-MiniLM-L6-v2` pre-trained model to obtain the utterance representations.[1] Any document embedding model can be used to generate the utterances representations and feed the `TEXT` model. We run the `TEXT` model on the non-anonymized version of BETOLD, ensuring the same train/test/validation splits for a fair comparison. See Appendix A for further details on the hyperparameters.

6 Results

6.1 Quantitative Analysis

Table 4 reports the results of the models in terms of the F1 score for each class and the macro-averaged score. As a first remark, the models `ENT` and `INT` obtain a similar performance. This is probably due to the fact that some intents can be recognized by the entities of which they are composed. But since not all the intents are characterized by entities, the `ENT` model is not able to reach the same performance as the `INT` one.

Focusing on the `IEC` model, which combines the intents, entities, and caller type, we can notice that this model gets improved performance with re-

[1] See https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. In a preliminary investigation, we tried different pre-trained sentence embedding models made available by Hugging Face. Here, we report the results with the best performing model.

	LUHF F1	not LUHF F1	Macro avg F1
TEXT	0.798 ± 0.016	0.903 ± 0.005	0.850 ± 0.010
INT	0.727 ± 0.018	0.877 ± 0.002	0.802 ± 0.010
ENT	0.707 ± 0.018	0.867 ± 0.007	0.787 ± 0.011
IEC	0.744 ± 0.008	0.879 ± 0.003	0.812 ± 0.005

Table 4: Results of the considered models in terms of F1 score. We report the average and the deviation of 5 independent runs with the same hyperparameter configuration.

spect to the INT and ENT counterparts, as expected. IEC also obtains good performance if compared with the text-only baseline. It is worth noticing that the TEXT model is indeed a strong baseline: the intents and entity annotations provided by the dialog manager synthesize the meaning of an utterance, mapping them to a finite set of intents and entities. With this process, we inevitably lose some information about the dialog. Moreover, intent and entity annotations are prone to classification errors: the NLU may misclassify an intent or it may not detect an entity. Despite these difficulties, the IEC model can reach a comparable performance to the TEXT baseline, suggesting that it is possible to confidently identify a breakdown even without taking text into account.

6.2 Qualitative Analysis

In this section, we discuss some qualitative examples of the IEC model on the test set, one of a LUHF classification and one of a not LUHF misclassified as a LUHF.

Let us consider the conversation shown in Table 5. For each step of the conversation, we report the corresponding caller type (NLG or NLU), the intent, the entities, and the probability of identifying a LUHF. We filter out the first steps of the conversations due to space limitations. Let us recall that the LUHF annotation corresponds to the overall call. Therefore the model was not trained on each step of the conversation because there is no information about if a breakdown occurred at a given step. We will further discuss this issue in Section 7. Nevertheless, we can still compute the probability of identifying a LUHF at each step for a conversation, by generating a synthetic dataset composed of the same conversation but incrementally truncated.

Table 5 shows that the probability of detecting a LUHF increases as the conversation progresses. However, the probability often decreases after a user input, represented by an NLU annotation. We presume that this behavior happens because a user is less likely to hang up after replying and would wait until the next utterance before deciding to hang up. For example, at step 23, a user negates the proposed date of the system. This may be a signal of a breakdown and indeed the probability of a LUHF increases at step 24. The intent *time_asked_unavailable_propose_new* indicates an NLG intent where a time preference proposed by the user is unavailable, therefore the system proposes a new time. This can be an additional signal for a breakdown, which in fact increases the probability of detecting a LUHF.

In Table 6, we report an example of a not LUHF that has been classified as a LUHF at the final step of the conversation. As before, we eliminated the early steps of the call, where the conversation flowed nicely. At steps 13 and 16, the probability of predicting a LUHF increases, although there are no clear indications of a breakdown. It is worth noting that in the previous example the probability of a LUHF also increased after the intent *propose_date* (Table 5). Many LUHFs may happen in correspondence to this intent, therefore biasing the model to believe that this intent is an indication of a breakdown.

7 Limitations

As mentioned in Section 3.1, the labels in BETOLD are automatically assigned to each conversation. Therefore, it is possible that a conversation where everything goes smoothly but suddenly the user decides to hang up is classified as LUHF. Similarly, the user may decide to reach the end of the conversation even if they are extremely unsatisfied with the call. This case is not considered a dialog breakdown.

In addition to this issue, a LUHF annotation applies to the overall call and is not an indication of what happened during at each step of the conversation. A model that generalizes well should be able to predict whether a LUHF happened at each step of the call. The process of data augmentation described in Section 5.3 is an attempt to address this issue. This is, however, limited to not LUHF calls, given that we have no guarantees on when a breakdown happened in a conversation. Instead, we are quite confident that, if a call was successful, it was also successful in the previous steps.

Step	Caller	Intent	Entities	Probability of LUHF
9	NLG	ask_for_battery_health		0.002
10	NLU	inform	numeric	0.000
11	NLG	ask_first_name		0.109
12	NLU	inform	client_name	0.002
13	NLG	ask_last_name		0.296
14	NLU	inform	client_name	0.016
15	NLG	ask_desired_service		0.344
16	NLU	user_initial_request	type_of_repair	0.033
17	NLG	ask_additional_service		0.262
18	NLU	inconclusive		0.012
19	NLG	transportation_of_device	ask_to_schedule, ask_means_of_transportation	0.253
20	NLU	confirm		0.014
21	NLG	inform_schedule_inspection		0.012
22	NLG	propose_date	transportation_type_selection, available_slot_to_schedule	0.269
23	NLU	negate		0.033
24	NLG	ask_time_preference		0.378
25	NLU	user_proposed_date	time_range_indication	0.153
26	NLG	time_asked_unavailable_propose_new	transportation_type_selection, available_slot_to_schedule, user_request_start_time	0.922
27	NLU	negate		0.764
28	NLG	ask_time_preference		0.960

Table 5: Example of LUHF conversation correctly classified.

Figure 3 shows a density histogram of the probability by class of predicting a LUHF, averaged over all the conversation steps, for the test set conversations. As we can observe in the plot, the average

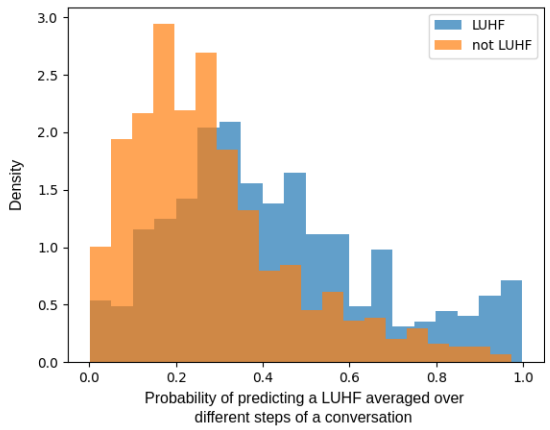

Figure 3: Histogram of the average probability of predicting a LUHF by class.

probability of predicting a LUHF in a not LUHF call is low in general. We recall that we have more data available for the successful calls (the not LUHFs are two times more than the LUHFs and, in addition, we add 30% more successful calls through augmentation). Therefore, it is not surprising to see that the not LUHF distribution is more skewed towards to 0 than the LUHF distribution. Moreover, the results shown in the plot correspond to the average score across many steps of the conversation. A breakdown may happen very late in the conversation, thus resulting in an overall low average score. However, this is hard to determine through an automatic investigation and would require a manual inspection.

8 Conclusions

In this paper, we proposed a simple way to automatically generate a breakdown detection dataset in task-oriented dialogs, where the breakdown labels are extracted by user-initiated events. This dataset guarantees the privacy of the data by only keeping the annotations from NLU and NLG components. We proposed an attention-based model which uses these types of annotations. As a result, we demonstrated that a model does not necessarily require textual utterances to predict a breakdown; yet, it can benefit from the NLG and NLU intents and entities, automatically provided by a classical dialog system.

Step	Caller	Intent	Entities	Probability of LUHF
11	NLG	transportation_of_device	ask_means_of_transportation, ask_to_schedule	0.164
12	NLU	confirm		0.005
13	NLG	ask_time_preference		0.391
14	NLU	confirm		0.011
15	NLG	inform_schedule_inspection		0.005
16	NLG	propose_date	transportation_type_selection, available_slot_to_schedule	0.678

Table 6: Example of not LUHF conversation misclassified as a LUHF.

We also discussed some possible limitations of the model and the dataset, connected to the annotation schema. An automatic annotation, as it often happens, results in noisy data. However, we do believe that the proposed dataset is a promising starting point, which can save resources and could be further improved through manual annotations.

As highlighted by the qualitative analysis, it is worth further investigating the results to understand which elements play a role in the detection of a LUHF. The implementation of explainability methods can be an important tool in this context (Lundberg and Lee, 2017). Given the transformer-based architecture of our proposed model, current explainability tools (Kokhlikyan et al., 2020; Attanasio et al., 2022) can enrich our investigation of the role of each attention head in the breakdown prediction. For that, gradient-based methods can give an overview of the importance of individual features as well as of the interactions.

References

Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. 2022. ferret: a Framework for Benchmarking Explainers on Transformers. *arXiv preprint*.

Dennis Benner, Edona Elshan, Sofia Schöbel, and Andreas Janson. 2021. What do you mean? A Review on Recovery Strategies to Overcome Conversational Breakdowns of Conversational Agents. In *International Conference on Information Systems (ICIS)*.

Petter Bae Brandtzæg and Asbjørn Følstad. 2018. Chatbots: changing user needs and motivations. *Interactions*, 25(5):38–43.

Pierre Colombo, Emile Chapuis, Matteo Manica, Emmanuel Vignon, Giovanna Varni, and Chloe Clavel. 2020. Guiding attention in sequence-to-sequence models for dialogue act prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7594–7601.

Allen L. Gorin, Giuseppe Riccardi, and Jeremy H. Wright. 1997. How may I help you? *Speech Commun.*, 23(1-2):113–127.

Mariya Hendriksen, Artuur Leeuwenberg, and Marie-Francine Moens. 2021. LSTM for dialogue breakdown detection: exploration of different model types and word embeddings. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction*, pages 443–453. Springer.

Ryuichiro Higashinaka, Kotaro Funakoshi, Masahiro Araki, Hiroshi Tsukahara, Yuka Kobayashi, and Masahiro Mizukami. 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*, pages 87–95.

Ryuichiro Higashinaka, Kotaro Funakoshi, Yuka Kobayashi, and Michimasa Inaba. 2016. The dialogue breakdown detection challenge: Task description, datasets, and evaluation metrics. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3146–3150.

Takaaki Hori, Hai Wang, Chiori Hori, Shinji Watanabe, Bret Harsham, Jonathan Le Roux, John R Hershey, Yusuke Koji, Yi Jing, Zhaocheng Zhu, et al. 2016. Dialog state tracking with attention-based sequence-to-sequence learning. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 552–558. IEEE.

Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems (TOIS)*, 38(3):1–32.

Sosuke Kato and Tetsuya Sakai. 2017. RSL17BD at DBDC3: computing utterance similarities based on term frequency and word embedding vectors. In *Proceedings of DSTC6.*, volume 34, pages 37–44.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch.

Seolhwa Lee, Dongyub Lee, Danial Hooshyar, Jaechoon Jo, and Heuiseok Lim. 2020. Integrating breakdown detection into dialogue systems to improve knowledge management: encoding temporal utterances with memory attention. *Information Technology and Management*, 21(1):51–59.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774.

Bilyana Martinovski and David Traum. 2003. Breakdown in human-machine interaction: the error is the clue. In *Proceedings of the ISCA tutorial and research workshop on Error handling in dialogue systems*, pages 11–16.

Kazuyuki Matsumoto, Manabu Sasayama, Minoru Yoshida, Kenji Kita, and Fuji Ren. 2022. Emotion analysis and dialogue breakdown detection in dialogue of chat systems based on deep neural networks. *Electronics*, 11(5).

Raveesh Meena, José Lopes, Gabriel Skantze, and Joakim Gustafson. 2015. Automatic detection of miscommunication in spoken dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 354–363.

Chanyoung Park, Kyungduk Kim, and Songkuk Kim. 2017. Attention-based dialog embedding for dialog breakdown detection. In *Proceedings of the dialog system technology challenges workshop (DSTC6)*.

Libo Qin, Zhouyang Li, Wanxiang Che, Minheng Ni, and Ting Liu. 2021. Co-GAT: A Co-Interactive Graph Attention Network for Joint Dialog Act Recognition and Sentiment Classification. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 13709–13717. AAAI Press.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Alexander Schmitt, Benjamin Schatz, and Wolfgang Minker. 2011. Modeling and predicting quality in spoken human-computer interaction. In *Proceedings of the SIGDIAL 2011 Conference*, pages 173–184.

J Shin, Alireza Dirafzoon, and Aviral Anshu. 2019. Context-enriched attentive memory network with global and local encoding for dialogue breakdown detection. *Proceedings of the WOCHAT*.

Hiroaki Sugiyama. 2021. Dialogue breakdown detection using BERT with traditional dialogue features. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction*, pages 419–427. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008.

Mattias Wahde and Marco Virgolin. 2022. Conversational agents: Theory and applications. *arXiv preprint*.

Marilyn A. Walker, Irene Langkilde, Jeremy H. Wright, Allen L. Gorin, and Diane J. Litman. 2000. Learning to predict problematic situations in a spoken dialogue system: Experiments with how may I help you ? In *6th Applied Natural Language Processing Conference, ANLP 2000, Seattle, Washington, USA, April 29 - May 4, 2000*, pages 210–217. ACL.

Chih-Hao Wang, Sosuke Kato, and Tetsuya Sakai. 2021. RSL19BD at DBDC4: ensemble of decision tree-based and LSTM-based models. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction*, pages 429–441. Springer.

Jason D. Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan W. Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Mag.*, 35(4):121–124.

Runhua Xu, Nathalie Baracaldo, and James Joshi. 2021. Privacy-preserving machine learning: Methods, challenges and directions. *arXiv preprint*.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63(10):2011–2027.

Tianyu Zhao and Tatsuya Kawahara. 2019. Joint dialog act segmentation and recognition in human conversations using attention to dialog context. *Computer Speech & Language*, 57:108–127.

A Hyperparameter Search

We report the hyperparameter space in Table 8. Table 7 shows the optimal hyperparameters selected after the grid search approach. We train the models for 15 epochs and select the best result based on the F1 score for the LUHF class on the validation set.

B Computing Infrastructure

We ran the experiments on a machine equipped with AMD® Ryzen 9 5900hx CPU, NVIDIA GeForce RTX 3060 GPU with CUDA v11.4, Driver Version 470.141.03 and 32GB RAM.

Parameter Description	Value
Embedded representations dimension l	400
Transformer Encoder embeddings dimension m	128
# layers of the Transformer Encoder t	3
# attention heads of the Transformer Encoder	16
Dropout ratio applied in the Transformer Encoder	0.01
Learning Rate	0.0001
Size of layers of decoding stage	(256, 32)
Optimizer	Adam
Number of epochs	15
BCE weight for not LUHFs	1.0
BCE weight for LUHFs	3.0

Table 7: Hyperparameters used for training the models.

Parameter Description	Value
Embedded representations dimension l	[256, 400, 800]
Transformer Encoder embeddings dimension m	[64, 128, 256]
# layers of the Transformer Encoder t	[1, 3, 5]
# attention heads of the Transformer Encoder	[4, 8, 16]
Dropout ratio applied in the Transformer Encoder	[0.01, 0.1, 0.5]
Learning Rate	[0.0001, 0.0005, 0.001]
Size of layers of decoding stage	[(256, 32), (256, 128, 32), (256, 64), (256, 128, 64), (256, 128, 64, 32)]
Optimizer	Adam
Number of epochs	50
BCE weight for not LUHFs	1.0
BCE weight for LUHFs	[1.0, 3.0, 5.0, 10.0]

Table 8: Hyperparameter space used for grid search.