HERALD: An Annotation Efficient Method to Detect User Disengagement in Social Conversations

Weixin Liang[1]
Stanford University
wxliang@stanford.edu

Kai-Hui Liang[1]
Columbia University
kaihui.liang@columbia.edu

Zhou Yu
Columbia University
zy2461@columbia.edu

Abstract

Open-domain dialog systems have a user-centric goal: to provide humans with an engaging conversation experience. User engagement is one of the most important metrics for evaluating open-domain dialog systems, and could also be used as real-time feedback to benefit dialog policy learning. Existing work on detecting user disengagement typically requires hand-labeling many dialog samples. We propose HERALD, an efficient annotation framework that reframes the training data annotation process as a denoising problem. Specifically, instead of manually labeling training samples, we first use a set of labeling heuristics to label training samples automatically. We then denoise the weakly labeled data using the Shapley algorithm. Finally, we use the denoised data to train a user engagement detector. Our experiments show that HERALD improves annotation efficiency significantly and achieves 86% user disengagement detection accuracy in two dialog corpora. Our implementation is available at `https://github.com/Weixin-Liang/HERALD/`.

1 Introduction

Evaluation metrics heavily influence a field's research direction. The ultimate goal of open-domain dialog systems is to provide an enjoyable experience to users. Previous research mainly focuses on optimizing automatic dialog evaluation metrics such as BLEU, which models the distance between the system responses and a limited number of references available. However, it has been shown that these metrics correlate poorly with human judgments (Liu et al., 2016).

Open-domain dialog system evaluation has long been one of the most difficult challenges in the dialog community for several reasons: (1) The goal of dialog evaluation should be to evaluate users' conversational experience. Existing automatic evaluation metrics such as BLEU are mostly constrained to a static corpus, and do not capture the user experience in a realistic interactive setting. (2) Currently, self-reported user ratings are widely used to evaluate open-domain dialogs. However, self-reported ratings suffer from bias and variance among different users (Liang et al., 2020e). Although we could tell which dialog system is better by running statistical tests on a large number of noisy ratings, it is challenging to locate dialogs with bad performance reliably. Only by identifying these bad dialogs effectively can we correct errors in these samples to improve dialog system quality.

User engagement has been recognized as one of the essential metrics for open-domain dialog evaluation (Ram et al., 2018). Previous research also confirms that incorporating user engagement as real-time feedback benefits dialog policy learning (Yu et al., 2016). One of the most costly bottlenecks of learning to detect user disengagement is to annotate many turn-level user engagement labels (Ghazarian et al., 2020). In addition, the data annotation process becomes more expensive and challenging for privacy-sensitive dialog corpora, due to the privacy concerns in crowdsourcing (Xia and McKernan, 2020).

To improve annotation efficiency, we reframe the training data annotation process as a denoising problem. Specifically, instead of manually labeling each training datum, we automatically label the training samples with a set of labeling heuristics. The heuristic functions primarily consist of regular expressions (Regexes) and incorporate open-sourced natural language understanding (NLU) services. Since the automatically generated labels might contain noise, we then denoise the labeled data using the Shapley algorithm (Jia et al., 2019a,b). We use the Shapley algorithm to

---

[1]Equal Contribution.

3652

quantify the contribution of each training datum, so that we can identify the noisy data points with negative contribution and then correct their labels. Our experiments show that HERALD achieves 86% accuracy in user disengagement detection in two dialog corpora.

Our proposed framework HERALD is conceptually simple and suitable for a wide range of application scenarios: First, since our model could detect user engagement in real-time (i.e., after each user utterance), our model could be plugged into existing dialog systems as a real-time user experience monitor module. In this way, dialog systems could detect and react to user's disengagement in both open-domain dialogs (Yu et al., 2016) and task-oriented dialogs (Yu et al., 2017). During training, our model could also be used as real-time feedback to benefit dialog policy learning (Yi et al., 2019). Second, HERALD could quantify user engagement and be used as an automatic dialog evaluation metric. It could locate dialogs with poor user experience reliably to improve dialog system quality (Ghazarian et al., 2020; Choi et al., 2019). Third, user engagement is an essential objective of dialog systems, but few dialog datasets with user engagement ratings are available. Our heuristic functions, combined with the proposed workflow, can be readily deployed to annotate new dialog datasets.

2 Related Work

2.1 Open-Domain Dialog System Evaluation

Open-domain dialog system evaluation is a long-lasting challenge. It has been shown that existing automatic dialog evaluation metrics correlate poorly with human judgments (Liu et al., 2016; Lowe et al., 2017; Novikova et al., 2017). A well-known reason is that these automatic dialog evaluation metrics rely on modeling the distance between the generated response and a limited number of references available. The fundamental gap between the open-ended nature of the conversations and the limited references (Gupta et al., 2019) is not addressed in methods that are lexical-level based (Papineni et al., 2002; Lin, 2004; Banerjee and Lavie, 2005), embedding based (Rus and Lintean, 2012; Forgues et al., 2014), perplexity based (Adiwardana et al., 2020), or learning based (Tao et al., 2018; Lowe et al., 2017). Mehri and Eskénazi (2020) simulate user response using DialogGPT and evaluate the probability of user complaint.

Given the limitations above, self-reported user ratings are widely used to evaluate open-domain dialogs. However, self-reported ratings suffer from bias and variance among different users (Venkatesh et al., 2018). Denoising human ratings is still an open research problem (Liang et al., 2020e; Li et al., 2019).

2.2 User Engagement in Dialogs

User engagement is commonly defined as the user's willingness to continue conversing with the dialog system (Yu et al., 2016, 2017). Existing work on measuring user engagement primarily resorts to human rating (Yi et al., 2019; Hancock et al., 2019), or proxy metrics. Example proxy metrics include conversation length like number of dialog turns (Venkatesh et al., 2018; Ram et al., 2018), and conversational breadth like topical diversity (Guo et al., 2018). Sporadic attempts have been made to detecting user disengagement in dialogs (Yu et al., 2004; Ghazarian et al., 2020; Choi et al., 2019). A major bottleneck of these methods is that they require hand-labeling many dialog samples for individual datasets. Although Liang et al. (2020e) denoise user self-reported ratings with the Shapley algorithm for dialog system evaluation, their method cannot be directly applied to dialogs without user ratings as in our setting. Our work is focusing on the problem that it is expensive and difficult to obtain user ratings. The core insight of our work is to reframe the training data annotation process as a process of denoising labels created by heuristic functions pre-defined. To the best of our knowledge, we are the first to combine automatic data labeling with the Shapley algorithm to perform dialog evaluation. Our method could potentially generalize to other classification tasks if different weak labelers are provided.

2.3 Learning from Weak Supervision

Learning from weak supervision reduces annotation costs by utilizing noisy but cost-efficient labels (Ratner et al., 2020, 2016; Liang et al., 2020e). One of the most popular forms of weak supervision is distant supervision, in which the records of an external knowledge base are heuristically aligned with data points to produce noisy labels for relationship extraction tasks (Bunescu and Mooney, 2007; Mintz et al., 2009; Hancock et al., 2018). Other applications of weak supervision to scene graph prediction (Krishna et al., 2019), intent classification (Mallinar et al., 2019), and medical imag-
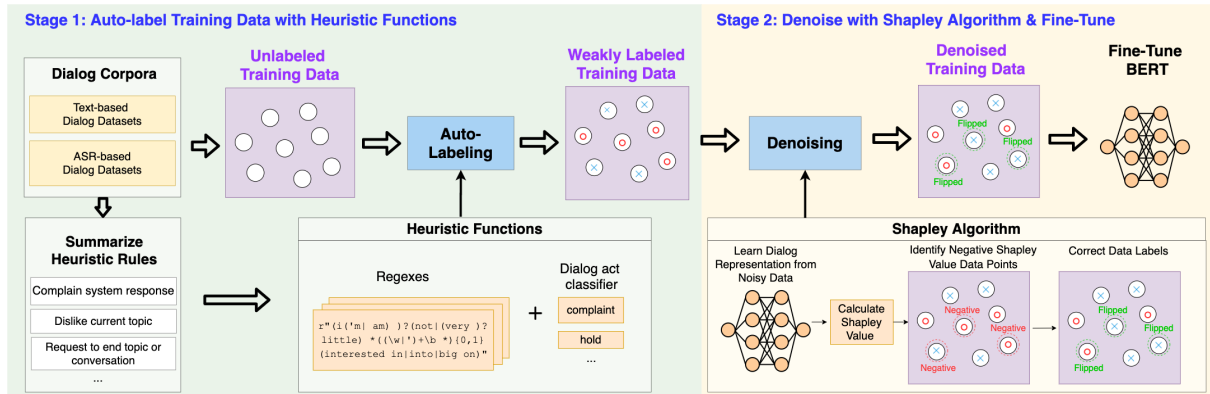
Figure 1: Schematic of the HERALD two-stage workflow. **Stage 1:** Auto-label training data with Heuristic Functions. We first design heuristics rules for detecting user disengagement by investigating multiple dialog corpora. The heuristics rules are implemented as heuristic functions based on regular expressions and dialog acts. Then, we use the heuristic function to label the training set automatically. **Stage 2:** Denoise weakly-labeled training data with Shapley Algorithm. We calculate the Shapley value for each data point and correct the noisy data points with negative Shapely values by flipping their labels. Finally, we fine-tune the model on the denoised training data.

ing (Varma et al., 2017) have observed similar benefits in annotation efficiency. Unlike the existing work, we leverage weak supervision to improve annotation efficiency for detecting user disengagement in social conversations.

3 Problem Formulation

We defined engagement as the degree to which users are willing to continue conversing with the dialog system Yu et al. (2016, 2017). We focus on identifying the dialog turns with "disengaged" user response, since they usually indicate poor conversation experience. We formulate the user engagement prediction as a binary classification problem: Our goal is to learn a parameterized user engagement predictor M_θ that, given a dialog turn (along with its dialog context) $x \in \mathcal{X}$, predicts the turn-level user engagement label $y \in \mathcal{Y} = \{0, 1\}$, where label $y = 1$ means "disengaged" and $y = 0$ means "engaged". We start from an unlabeled train set $D_{\text{train}} = \{x_i\}_1^{N_{\text{train}}}$ without any label y_i. The test set $D_{\text{test}} = \{(x_i, y_i)\}_1^{N_{\text{test}}}$ contains the ground-truth label y_i. The development set D_{dev} has a similar structure as the test set D_{test} but the development set can be much smaller than a train set (i.e., $N_{\text{dev}} \ll N_{\text{train}}$), making it economical to obtain. Following the general architecture of neural classifiers, we formulate our model $M_\theta = M(\phi, f) = f(\phi(x))$: Here BERT (Devlin et al., 2019)-based ϕ is a text encoder that maps each dialog turn x to a feature space $\phi(x) \in \mathbb{R}^d$. f is the final linear layer with softmax activation.

4 Data

To ensure our framework is generalized to various corpora, we investigate multiple open-domain dialog datasets ranging from ASR-based (Gunrock (Liang et al., 2020a)) to text-based (ConvAI2 (Dinan et al., 2019), Blender (Roller et al., 2020), and Meena (Adiwardana et al., 2020)) dialog systems.

Gunrock Movie Dataset Gunrock Movie dataset consists of dialog data collected from Gunrock, an ASR-based open-domain social chatbot originally designed for Amazon Alexa Prize (Liang et al., 2020a). The Gunrock dataset comes from a user study where in-lab users were recruited to carry on conversations. We have consent to use the data and we also removed any sensitive information in the conversation. Two dialog experts (co-authors of this paper) randomly annotated 134 dialogs and split them evenly into the test set and development set. In total, the experts labeled 519 turn-level disengaging user responses and 2,312 engaging user responses. They reached a high inter-annotator agreement score (Cohen, 1968) with kappa $\kappa = 0.78$. The training set contains 276 unlabeled dialogs, with 5644 dialog turns. In addition, we ensure that the data annotation is independent of the labeling heuristics collection, so there is no data leakage problem. A full example dialog can be found in Appendix A.4.

ConvAI2 Dataset ConvAI2 dataset contains text-based dialog collected from the second Conver-

Labeling Heuristics		Coverage (%)		Example Disengaged User Responses
Heuristics Group	Disengaged intents	Gunrock	ConvAI2	
(1) Complain system responses	Complain system repetition Complain system ignoring them Complain system misunderstanding Not understanding system Curse system Express frustration	1.93	1.95	{ You already asked me that. | I already told you. Remember? } { You're not listening. | You didn't answer my question. } { I never said I don't eat my favorite seafood. } { What are you talking about? } { You're dumb. } { Sigh. }
(2) Dislike current topic	Express negative opinion Show low interests	1.90	3.45	{ I don't like music. | It's boring. } { I don't care. }
(3) Request to end topic or conversation	Request topic change Request termination	5.20	2.92	{ Let's talk about something else. } { Stop. | Bye. }
(4) End with non-positive responses	End with negative answer End with unsure answer End with back-channeling End with hesitation	20.13	4.86	{ No. | I have not. } { I don't know. | I don't remember. | Well, maybe. } { Yeah. | Okay. } { Hmm... | That's a hard one, let me think. }

Table 1: Our labeling heuristics designed to capture user disengagement in dialogs. A dialog turn is considered disengaged if any of the heuristic rules apply to the user responses.

sational Intelligence (ConvAI) Challenge (Dinan et al., 2019). We select dialogs from the main eight participated chatbots (Bot 1, 2, 3, 4, 6, 9, 11) and exclude dialogs that are one-sided or shorter than three turns. The dialog experts annotated 207 dialogs in total. The dialogs are evenly distributed over all the eight bots to ensure system diversity, and are randomly sampled within each bot. The annotated data consist of 209 disengaging turns and 1684 non-disengaging turns. They reached a high inter-annotator agreement score (Cohen, 1968) with kappa $\kappa = 0.76$. We split the annotated dialogs evenly into the test set and develop set. The training set contains 2,226 dialogs, with 18,306 dialog turns.

Google Meena Dataset Meena (Adiwardana et al., 2020) is the largest end-to-end neural chatbot so far, trained on $867M$ public domain social media conversations. We study the 93 example Human-Menna conversations released by Google.

Facebook Blender Dataset The Blender bot (Roller et al., 2020) is an open-domain chatbot with several conversational skills: providing engaging talking points and listening to their partners, displaying knowledge, empathy, and personality appropriately while maintaining a consistent persona. We study the 108 example Human-Blender conversations released by Facebook.

5 Method

Our goal is to train a user engagement detector with minimum data annotation efforts. Traditional supervised learning paradigms require annotating many training samples. In addition, it requires additional data annotation to extend the model to a new dialog corpus. To reduce annotation work, we propose HERALD, a two-stage pipeline that annotates large-scale training data efficiently and accurately (Figure 1). Instead of hand-labeling training data points, we use heuristic functions to label each training datum automatically. The heuristic functions are built upon a set of user disengagement heuristics rules. Since the training data are automatically labeled, their labels would be noisy. We then clean the noisy training data with Shapley algorithm (Ghorbani and Zou, 2019) to improve the labeling accuracy. The Shapley algorithm denoises training data by identifying data with wrong labels and flip their labels. Finally, as we received clean training data, we use them to fine-tune a BERT-based model and obtain the final user disengagement detection model.

5.1 Stage 1: Auto-label Training Data with Heuristic Functions

Since labeling large-scale training data is time-consuming, we propose heuristic labeling functions to label training data automatically. The heuristic functions focus on detecting disengagement from user responses, as it directly indicates poor user experience. To build the heuristics functions, we first summarize the heuristic rules shared among users. We investigate the disengaged dialog turns from the four datasets mentioned above and identify four groups of user disengagement patterns: "complain system responses", "dislike current topics", "terminate or change topics", and "end with non-positive responses" (Table 1). We then discuss the implementation of heuristics functions.

5.1.1 Disengagement Heuristic Rules

Group 1: Complain system responses. Complaints are an evident sign of user disengagement. We identify six related disengaged intents. The first three intents ("complain system repetition", "complain system ignoring them" and "complain system misunderstanding") usually appear when the bot makes errors like repeating the same content, ignoring, forgetting, and misunderstanding the user's response. In these cases, users express their disengagement by indicating the bot's error (e.g. "You already told me that", "You're not listening"). Another intent "not understanding system" happens when users cannot understand the system's response (e.g. "I don't know what you're talking about."). In the last two intents, users reveal negative emotions by cursing the system (e.g. "you're dumb") or express frustration (e.g. "sigh") about the conversation.

Group 2: Dislike current topics. When discussing a given topic, users might show their disengagement by expressing negative opinions or low interest. For example, given the bot's response, "I write romantic novels under a pen name. ", for users who are not interested in reading, users might say "reading is boring", "I don't like to read", or "I'm not interested in this". We also make sure to handle the corner cases where the user utterance should be labeled as engaged but contains negative opinions. For instance, to respond to the bot's question, "do you want to not work?", a user might say, "Yes. my job is boring. I have to work with mail". Though the user mentions a negative feeling ("boring"), the user agrees with the bot and shares further information.

Group 3: Terminate or change topics Group 3 considers the cases where users express disengagement to the current topic in a more straightforward fashion. For example, if users are not interested in the current topic, instead of just expressing their dislike to it, they may request to switch topics with "Let's talk about something else". In some cases, users might show strong disengagement by requesting to end the conversation if the user is no longer interested in continuing the conversation.

Group 4: End with non-positive responses A more subtle but common clue of disengagement is when users end the response with non-positive content. For example, non-positive responses like "I don't know", "No", "Yeah", "uh", "Probably",

imply that users do not have much to talk about the current topic. To keep the precision of our heuristics high, we carefully consider the counterexamples. One case is that the user follows up with more responses such as questions (e.g., Bot: "Have you seen any movies lately? ", User: "No. Have you?"), and opinion (e.g. Bot: "What's your favorite animation movie?", User: "I don't know, but it might actually be frozen two. My sister loves it.") in the same dialog turn. These turns should not be labeled as disengaged since the user is still interested in sharing more content or asking follow-up questions. Therefore, we take a conservative approach: we label the dialog turn as disengaged only if no more responses follow the non-positive response.

5.1.2 Heuristic Functions Implementation

Next, we discuss how to use heuristic functions to auto-label disengaged user utterances. First, we split user responses into segments since user responses may consist of multiple units with different semantic meanings. We use NLTK Sentence Tokenizer for text-based system, and a segmentation model (Chen et al., 2018) for ASR (Automatic Speech Recognition)-based system as the segmentation tool. We then apply the heuristic functions on each segment to detect disengaged intents. For heuristic groups 1 to 3, if any segment contains a disengaged intent, the user response is auto-labeled as disengaged. For heuristic group 4 ("End with non-positive responses"), we assign disengaged labels only if the disengaged intents are detected in the last segment.

We detect disengaged intents with Regexes. The benefit of using Regexes is that they have minimum dependencies and are easy to modify. We design Regexes for each intent. Following common Regexes complexity metrics (Luo et al., 2018), our Regexes for each intent contains 43.9 Regexes groups and 87.7 *or* clauses on average.

Our framework also supports incorporating additional resources to improve the intent detection accuracy for automatic training data labeling. For example, we can enhance the recall of Regexes intent detection by incorporating existing deep learning-based NLU (Natural Language Understanding) models. Specifically, we re-purpose an open-sourced dialog act classification model (Yu and Yu, 2021) to enhance disengagement intent detection: we select 6 out of the 23 supported dialog act labels that are associated with disen-

gaged intents, and map each selected dialog act label to the heuristic groups. The dialog act "complaint" is mapped to the heuristic group "complain system repetition";"closing" is mapped to the disengaged intent "request termination"; "hold" to "hesitation";"other_answers" to "unsure answer"; "back-channeling" to "back-channeling", and "neg_answer" to 'negative answer'". If a user utterance is detected with disengaged intent by either Regexes or the deep learning model, then the utterance is auto-labeled as disengaged.

5.2 Stage 2: Denoise with Shapley Algorithm & Fine-tune

Overview   Next, we denoise the labeled data using Shapley algorithm (Ghorbani and Zou, 2019). Shapley algorithm has been studied in the co-operative game theory (Dubey, 1975) and economics (Gul, 1989) as a fair distribution method. Shapley algorithm computes a Shapley value for each training datum, which quantifies the contribution of each training datum to the prediction and performance of a deep network. Low Shapley value data capture outliers and corruptions. Therefore, we can identify and denoise the incorrectly labeled data by computing their Shapley values and fine-tune the model on the cleaned training set.

Shapley Algorithm   Shapley algorithm comes originally from cooperative game theory (Dubey, 1975). Consider a cooperative game with n players $D = \{1, ..., n\}$ and a utility function $v : 2^{[n]} \rightarrow \mathbb{R}$ which assigns a reward to each of 2^n subsets of players: $v(S)$ is the reward if the players in subset $S \subseteq D$ cooperate. Shapley value defines a unique scheme to distribute the total gains generated by the coalition of all players $v(D)$ with a set of appealing mathematical properties. In our setting, we can consider $D_{train} = \{(x_i, y_i)\}_1^{N_{train}}$ as N_{train} players. We define the utility function $v(S)$ as the performance on the development set D_{dev}. The Shapley value for player i is defined as the average marginal contribution of $\{(x_i, y_i)\}$ to all possible subsets that are formed by other players (Jia et al., 2019a,b):

$$s_i = \frac{1}{N} \sum_{S \subseteq D_{train} \setminus \{x_i\}} \frac{1}{\binom{N-1}{|S|}} [v(S \cup \{x_i\}) - v(S)]$$

As suggested by the definition of Shapley value, computing Shapley value requires an exponentially large number of computations to enumerate $O(2^{N_{train}})$ possible subsets and train the model M_θ on each subset, which is intractable. Inspired

by (Jia et al., 2019a,b), HERALD tackles this issue by reducing the deep model M_θ to a K-nearest neighbors (KNN) model and then apply the closed-form solution of Shapley value on KNN: We reduce our BERT-based classification model $M_\theta = M(\phi, f) = f(\phi(x))$ to a KNN by first fine-tuning M_θ on the auto-labeled training samples. We then use the feature extractor ϕ to map each training datum to the feature space $\{\phi(x_i)\}_1^{N_{train}}$. We construct a KNN classifier in the feature space to compute the closed-form Shapley value.

Next, we discuss the closed-form solution of Shapley value. We first consider a special case where the development set D_{dev} only contains one datum $D_{dev} = \{(x_{dev}, y_{dev})\}$. Given any nonempty subset $S \subseteq D_{train}$, we use the KNN classifier to classify x_{dev}. To do this, we sort the data points in the training set $\{x_i\}_1^{N_{train}}$ based on their euclidean distance in the feature space $\phi(x)$ to the datum in the development set x_{dev}, yielding $(x_{\alpha_1}, x_{\alpha_2}, ..., x_{\alpha_{|S|}})$ with $x_{\alpha_1}, ..., x_{\alpha_K}$ as the top-K most similar data points to x_{dev}. The KNN classifier outputs the probability of x_{dev} taking the label y_{dev} as $P[x_{dev} \rightarrow y_{dev}] = \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}[y_{\alpha_k} = y_{dev}]$, where α_k is the index of the kth nearest neighbor. We define the utility function as the likelihood of the correct label:

$$v(S) = \frac{1}{K} \sum_{k=1}^{\min\{K,|S|\}} \mathbb{1}[y_{\alpha_k(S)} = y_{dev}] \qquad (1)$$

Jia et al. (2019a,b) proves that the Shapley value of each training point s_{α_i} can be calculated recursively in $O(N \log N)$ time as follows:

$$s_{\alpha_N} = \frac{\mathbb{1}[y_{\alpha_N} = y_{dev}]}{N}$$

$$s_{\alpha_i} = s_{\alpha_{i+1}} + \frac{\min\{K, i\}}{i \times K}(\mathbb{1}[y_{\alpha_i}=y_{dev}] - \mathbb{1}[y_{\alpha_{i+1}}=y_{dev}])$$

The above result for a single point in D_{dev} could be readily extended to the multiple-point case, in which the utility function is defined by

$$v(S) = \frac{1}{N_{dev}} \sum_{j=1}^{N_{dev}} \frac{1}{K} \sum_{k=1}^{\min\{K,|S|\}} \mathbb{1}[y_{\alpha_k^{(j)}(S)} = y_{dev,j}]$$

where $\alpha_k^{(j)}(S)$ is the index of the kth nearest neighbor in S to $x_{dev,j}$. Jia et al. (2019a,b) also prove that the Shapley value in this case is the average of the Shapley value for every single dev point.

Denoising Procedure   Our denoising procedure works as follows: (1) We first fine-tune our BERT-based classification model $M_\theta = M(\phi, f) = f(\phi(x))$

3657

No.	Method	Gunrock Movie		ConvAI2	
		bACC	F_2Score	bACC	F_2Score
(1)	Heuristics	78.32	65.09	76.58	58.16
(2)	Heuristics (regex only)	62.81	35.46	72.04	49.90
(3)	Heuristics (NLU only)	72.68	56.32	63.62	32.86
(4)	Heuristics w/o Group 1	78.21	64.88	71.20	48.44
(5)	Heuristics w/o Group 2	77.96	64.49	75.45	56.22
(6)	Heuristics w/o Group 3	71.52	55.36	71.96	49.80
(7)	Heuristics w/o Group 4	58.34	23.97	68.32	42.68
(8)	BERT(dev)	73.98	60.74	74.97	55.40
(9)	BERT(Auto)	80.55	71.77	78.76	63.13
(10)	BERT(Auto+dev)	80.73	72.16	80.46	64.54
(11)	**HERALD**	**86.17***	**80.01***	**86.22***	**70.49***

Table 2: Evaluation results comparison among variants of HERALD. * indicates that the model is statistically significantly better than baseline models. All numbers in the table are in percentage.

on the auto-labeled training samples. This step injects the knowledge in the labeling heuristic into the model M_θ. (2) We then map each auto-labeled training datum to the feature space $\{\phi(x_i)\}_1^{N_{train}}$, since we want to apply the closed-form KNN formula of Shapley value in the feature space. (3) Next, for a binary classification problem, we duplicate each training datum 2 times with labels [0, 1]. This generates a large training set D_{large} with $2 \times N_{train}$ data points, and we note that the origin training set D_{train} is a subset of D_{large}, since D_{large} enumerates all C possible labels for each each training datum. (4) We then calculate Shapley value for the $2 \times N_{train}$ data points in D_{large} using the closed-form KNN formula. (5) We remove the data with negative Shapley value in D_{large}, and get a cleaned training set D_{clean}. The duplicate-and-remove procedure "flips" the labels of the noisy data points with low Shapley value. (6) Finally, we fine-tune the classification model M_θ on D_{clean} to get the final user disengagement detection model.

To sum up, the Shapley value quantifies the contribution of each training datum. Low Shapley value data capture outliers and corruptions that are not consistent with the distribution of other data points. We identify and correct these outliers and corruptions to provide a clean training set.

6 Experiments

Model Setup We use $K = 10$ for the KNN Classifier. We use BERT (Devlin et al., 2019) as the text encoder ϕ of our classification model $M_\theta = M(\phi, f) = f(\phi(x))$. Additional implementa-

tion details are included in Appendix.

Model Comparisons and Ablations We compare HERALD to its several ablations (Table 2) and evaluate the performance on the test set. We report balanced accuracy (bACC) and F_β Score with $\beta = 2$ (Baeza-Yates et al., 1999). (1) *Heuristics* uses the labeling heuristic function with both Regex and dialog act to predict the test set. (2) *Heuristics (Regex only)* uses the labeling heuristic function only with Regex to predict on the test set. (3) *Heuristics (NLU only)* uses the labeling heuristic function only with NLU. (4-7) show the ablation of the heuristics function prediction baseline by excluding each heuristic group. (8) *BERT(dev)* fine-tunes BERT on the expert-annotated development set. (9) *BERT(Auto)* fine-tunes BERT on the auto-labeled training samples. (10) *BERT(Auto+dev)* fine-tunes BERT on both the auto-labeled training samples and the development set. (11) *HERALD* reports the performance of the final model trained on D_{clean}.

Results Our first takeaway is that our labeling heuristics produce decent predictions and generalize to different datasets. As shown in Table 2, Heuristics prediction (Heuristic, 78.32%, 76.58%) is better than the BERT-based model with limited training samples (BERT(dev), 73.98%, 74.94%) on both datasets. It also shows that our labeling heuristics are generalizable to different corpora.

Our second takeaway is that learning from a large number of noisy labels works better than learning from a limited number of clean labels. As shown in Table 2, BERT fine-tuned on the auto-labeled training set (BERT(Auto), 80.55, 78.76) outperforms BERT fine-tuned on clean but small development set (BERT(dev), 73.98, 74.94) by a large margin. In addition, we also observe that the BERT model fine-tuned on the auto labeled training data (BERT(Auto), 80.55%, 78.76%) generalizes beyond the labeling heuristics (Heuristics, 78.32%, 76.58%).

Our third takeaway is that using the expert-annotated development set for denoising is more efficient than using the development set as additional training data. After fine-tuning BERT on the weakly labeled training data (BERT(Auto), 80.55%, 78.76%), having an additional fine-tuning step using the development set slightly improves the model's performance (BERT(Auto+dev), 80.73%, 80.46%). In contrast,

using the development set for the Shapley denoising algorithm gives a significant performance gain (HERALD, 86.17%, 86.22%).
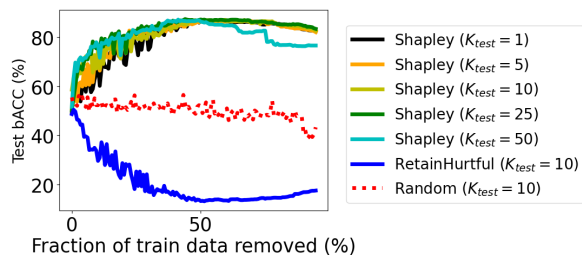

Figure 2: Removing data with low Shapley values (Shapley with K_{test} = 1, 5, 10, 25, 50) improves the performance of the KNN in Gunrock Movie Dataset while removing data with high Shapley values and retain data with low Shapley values ("RetainHurtful") leads to worse performance.

Annotation Cost   The cost of annotating the DEV set is small for the Shapley algorithm. For Gunrock Movie Dataset, we used 67 annotated dialogs as the DEV set. For ConvAI2, we used 52 annotated dialogs as the DEV set. The annotation takes less than 1 hour in both cases, which is negligible compared to the cost of annotating all training data.

Heuristics Group Analysis   We perform ablation studies to analyze the importance of each of the four heuristics groups in Table 1. As shown in Table 2, excluding heuristics group 4 leads to the most significant performance drop in both datasets (Heuristics w/o Group 4, 58.34%, 68.32%), indicating that "end with non-positive response" is the most prevalent form of user disengagement.

In addition, each heuristics group has different importance in different datasets. For example, dropping heuristics group 1 ("complain system responses") only leads to a marginal performance drop on the Gunrock Movie dataset but incurs a significant performance drop on the ConvAI2 dataset. We also notice that heuristic group 4 ("End with non-positive responses") plays a more critical role in the Gunrock Movie dataset than in the ConvAI2 dataset. This might be mainly due to the difference between ASR-based (Gunrock Movie) and text-based (ConvAI2) systems. When asked an open-ended question in ASR-based systems, since users have less time to think, they are more likely to reply with responses such as "I'm not sure", "let me think". While in text-based systems (ConvAI2), users have more time to think and formulate

their responses. Hence, heuristics group 4 covering these responses happen more in Gunrock Movie than ConvAI2.

Generalizability of Heuristic Functions   The results show that our heuristic functions are generalized to both ASR-based and text-based systems. As indicated in Table 2, our Regexes reach a decent accuracy of 62.81% and 72.04% on the expert annotated test set respectively on Gunrock Movie and ConvAI2 dataset, and thus can serve as a relatively reliable source for auto-labeling. In addition, although the dialog act model (MIDAS) is initially designed for ASR-based systems and thus has a better performance on the Gunrock Movie data, it should be generalizable to other ASR-based systems, as the six selected dialog acts are general and independent of topics. Therefore, the combination of dialog acts and Regexes should be sufficient to be applied to various corpora.

Figure 3: An example dialog turn from the Gunrock Movie dataset with an incorrect auto label "non-disengaged" identified by data Shapley. In this case, the user actually says "I don't wanna talk about movies anymore," but an ASR error happens, and thus the labeling heuristics fail to capture this dialog turn.

Figure 4: An example dialog turn from Gunrock Movie dataset that is incorrectly auto-labeled as "disengaged" because the labeling heuristics see the negative word "disagree". This data point is also identified and corrected by data Shapley.

Shapley Value Analysis   We also present an analysis to show how Shapley denoising works, as shown in Figure 2. We examine the Shapley value for each training datum in Stage 2. We first show two example dialog turns from the Gunrock Movie dataset with a negative Shapley value in Figure 3 and Figure 4. In Figure 3, the dialog turn is incorrectly auto-labeled as "non-disengaged". This is because an ASR error happens, and the user utterance "I don't wanna talk about movies anymore"

is transcribed as "I wanna talk about movies anymore". In Figure 4, the user says, "Oh I disagree. I think the movie was fantastic!". The labeling heuristics see the negative word "disagree" and auto-label this turn as "disengaged". Both data points are with negative Shapley values and are corrected in Stage 3.

Next, we present a quantitative analysis of Shapley value. According to the Shapley value, we remove data points one by one, starting from the least valuable (low Shapley values) to the most valuable (high Shapley values). Each time, after removing the data point, we create new KNN classifier models on the remaining dialog turns and labels and evaluate them on the test set with expert annotations. As shown in Figure 2, removing training data with low Shapley values increases the performance to a certain point before convergence for K of all choices. We observe a similar trend when re-training a model on the remaining data. In contrast, removing data randomly or removing data starting from high Shapley values decreases the performance on the test set ("Random" and "Retain-Hurtful" in Figure 2). This shows that low Shapley value data effectively capture outliers and corruptions, which further justifies our design choice of denoising with Shapley value.

Alternative Data Valuation Methods We also explored alternative methods to data Shapley like influence function (Koh and Liang, 2017) and TracIn (Pruthi et al., 2020): on Gunrock Movie, Influence Functions and TracIn achieve 82.96% and 83.15% accuracy, respectively. Both methods outperform BERT(Auto+dev) (80.73%) significantly but perform slightly worse than HERALD (86.17%). Overall, results show that our data annotation workflow also works well with other data valuation methods.

> **System:** Ok, yeah. Well personally, I am more of a movie-lover. Are you interested in movies too?
>
> **User:** I have a question for you social bot.

Figure 5: An error case where the low engagement dialog turn that is not captured by HERALD.

Error Analysis Figure 5 shows an error example of HERALD, where both the labeling heuristics and the Shapley algorithm fail to identify this turn as low engagement. In this example, the chatbot system asks whether the user is interested in movies, but the user does not directly answer the question. Instead, the user says "I have a question for you social bot", indicating that the user does not like the current topic and wants to talk about something else. HERALD fails to identify this dialog turn as low engagement, partly because the Regexes in the "request topic change" heuristic rule does not cover this example. One way to fix this error is to upgrade the Regexes. A more general solution is to consider the chatbot system's expectations on user responses conditioned on the chatbot's question. If the chatbot receives an "unexpected" user response, then the user is probably not interested in discussing the current topic.

7 Conclusion

The ultimate chatbot evaluation metric should be user-centric, as chatbots are there to provide humans with enjoyable experiences. Previously detecting user disengagement typically requires annotating many dialog samples for each individual dataset. We propose a two-stage pipeline HERALD to automatically label and denoise training data and, at the same time, build a user disengagement detector. Our experiment shows that HERALD significantly reduces the annotation cost of a new corpus. HERALD's disengagement detection results highly correlate with expert judgments on user disengagement in both datasets (86.17% bACC in Gunrock Movie, 86.22% in ConvAI2).

References

Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*, pages 65–72. Association for Computational Linguistics.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *ACL*. The Association for Computational Linguistics.

Chun-Yen Chen, Dian Yu, Weiming Wen, Yi Mang Yang, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, et al. 2018. Gunrock: Building a human-like social bot by leveraging large scale real user data. *Alexa Prize Proceedings*.

Jason Ingyu Choi, Ali Ahmadvand, and Eugene Agichtein. 2019. Offline and online satisfaction prediction in open-domain conversational systems. In *CIKM*, pages 1281–1290. ACM.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019. The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098*.

Pradeep Dubey. 1975. On the uniqueness of the shapley value. *International Journal of Game Theory*, 4(3):131–139.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2.

Sarik Ghazarian, Ralph M. Weischedel, Aram Galstyan, and Nanyun Peng. 2020. Predictive engagement: An efficient metric for automatic evaluation of open-domain dialogue systems. In *AAAI*, pages 7789–7796. AAAI Press.

Amirata Ghorbani and James Y. Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR.

Faruk Gul. 1989. Bargaining foundations of shapley value. *Econometrica: Journal of the Econometric Society*, pages 81–95.

Fenfei Guo, Angeliki Metallinou, Chandra Khatri, Anirudh Raju, Anu Venkatesh, and Ashwin Ram. 2018. Topic-based evaluation for conversational bots. *CoRR*, abs/1801.03622.

Prakhar Gupta, Shikib Mehri, Tiancheng Zhao, Amy Pavel, Maxine Eskénazi, and Jeffrey P. Bigham. 2019. Investigating evaluation of open-domain dialogue systems with human generated multiple references. *CoRR*, abs/1907.10568.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazaré, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *ACL (1)*, pages 3667–3684. Association for Computational Linguistics.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *ACL (1)*, pages 1884–1895. Association for Computational Linguistics.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gürel, Bo Li, Ce Zhang, Costas J. Spanos, and Dawn Song. 2019a. Efficient task-specific data valuation for nearest neighbor algorithms. *PVLDB*, 12(11):1610–1623.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. 2019b. Towards efficient data valuation based on the shapley value. In *AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

Ranjay Krishna, Vincent S. Chen, Paroma Varma, Michael Bernstein, Christopher Ré, and Fei-Fei Li. 2019. Scene graph prediction with limited labels. In *ICCV*, pages 2580–2590. IEEE.

Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: improved dialogue evaluation with optimized questions and multi-turn comparisons. *CoRR*, abs/1909.03087.

Kaihui Liang, Austin Chau, Yu Li, Xueyuan Lu, Dian Yu, Mingyang Zhou, Ishan Jain, Sam Davidson, Josh Arnold, Minh Nguyen, et al. 2020a. Gunrock 2.0: A user adaptive social conversational system. *arXiv preprint arXiv:2011.08906*.

Weixin Liang, Yanhao Jiang, and Zixuan Liu. 2021. GraghVQA: Language-guided graph neural networks for graph-based visual question answering. In *MAI@NAACL-HLT*. Association for Computational Linguistics.

Weixin Liang, Feiyang Niu, Aishwarya N. Reganti, Govind Thattai, and Gökhan Tür. 2020b. LRTA: A transparent neural-symbolic reasoning framework with modular supervision for visual question answering. *CoRR*, abs/2011.10731.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020c. MOSS: end-to-end dialog system framework with modular supervision. In *AAAI*, pages 8327–8335. AAAI Press.

Weixin Liang and James Zou. 2021. Neural group testing to accelerate deep learning. In *IEEE International Symposium on Information Theory, ISIT 2021*. IEEE.

Weixin Liang, James Zou, and Zhou Yu. 2020d. AL-ICE: active learning with contrastive natural language explanations. In *EMNLP (1)*, pages 4380–4391. Association for Computational Linguistics.

Weixin Liang, James Zou, and Zhou Yu. 2020e. Beyond user self-reported likert scale ratings: A comparison model for automatic dialog evaluation. In *ACL*, pages 1363–1374. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, pages 2122–2132. The Association for Computational Linguistics.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ACL (1)*, pages 1116–1126. Association for Computational Linguistics.

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying up regular expressions with neural networks: A case study for spoken language understanding. *arXiv preprint arXiv:1805.05588*.

Neil Mallinar, Abhishek Shah, Rajendra Ugrani, Ayush Gupta, Manikandan Gurusankar, Tin Kam Ho, Q. Vera Liao, Yunfeng Zhang, Rachel K. E. Bellamy, Robert Yates, Chris Desmarais, and Blake McGregor. 2019. Bootstrapping conversational agents with weak supervision. In *AAAI*, pages 9528–9533. AAAI Press.

Shikib Mehri and Maxine Eskénazi. 2020. Unsupervised evaluation of interactive dialog with dialogpt. In *SIGdial*, pages 225–235. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*, pages 1003–1011. The Association for Computer Linguistics.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *EMNLP*, pages 2241–2252. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *NeurIPS*.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2018. Conversational AI: the science behind the alexa prize. *CoRR*, abs/1801.03604.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason A. Fries, Sen Wu, and Christopher Ré. 2020. Snorkel: rapid training data creation with weak supervision. *VLDB J.*, 29(2-3):709–730.

Alexander J. Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *NIPS*, pages 3567–3575.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2020. Recipes for building an open-domain chatbot. *CoRR*, abs/2004.13637.

Vasile Rus and Mihai C. Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *BEA@NAACL-HLT*, pages 157–162. The Association for Computer Linguistics.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. RUBER: an unsupervised method for automatic evaluation of open-domain dialog systems. In *AAAI*, pages 722–729. AAAI Press.

Paroma Varma, Bryan D. He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel L. Rubin, and Christopher Ré. 2017. Inferring generative model structure with static analysis. In *NIPS*, pages 240–250.

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, Rahul Goel, Shaohua Yang, and Anirudh Raju. 2018. On evaluating and comparing conversational agents. *CoRR*, abs/1801.03625.

Huichuan Xia and Brian McKernan. 2020. Privacy in crowdsourcing: a review of the threats and challenges. *Comput. Support. Cooperative Work.*, 29(3):263–301.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. *CoRR*, abs/1904.13015.

Chen Yu, Paul M. Aoki, and Allison Woodruff. 2004. Detecting user engagement in everyday conversations. In *INTERSPEECH*. ISCA.

Dian Yu and Zhou Yu. 2021. Midas: A dialog act annotation scheme for open domain human machine spoken conversations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, page 1103–1120.

Zhou Yu, Leah Nicolich-Henkin, Alan W. Black, and Alexander I. Rudnicky. 2016. A wizard-of-oz study on A non-task-oriented dialog systems that reacts to user engagement. In *SIGDIAL Conference*, pages 55–63. The Association for Computer Linguistics.

Zhou Yu, Vikram Ramanarayanan, Patrick L. Lange, and David Suendermann-Oeft. 2017. An open-source dialog system with real-time engagement tracking for job interview training applications. In *IWSDS*, volume 510 of *Lecture Notes in Electrical Engineering*, pages 199–207. Springer.

A Appendix

A.1 Implementation Details of HERALD

We use $K = 10$ for the KNN Regressor. We load and fine-tune pre-trained BERT as the feature extractor ϕ. The details of extending BERT to encode multi-turn dialogs are as follows. Each dialog turn (along with its dialog context) is represented as a sequence of tokens in the following input format (Liang et al., 2020c): Starting with a special starting token $[CLS]$, we concatenate tokenized user and system utterances in chronological order with $[SEP]$ as the separators for adjacent utterance. In other words, we represent each dialog as a sequence: $[CLS]$, $S_{1,1}$, $S_{1,2}$, ..., $[SEP]$, $U_{1,1}$, $U_{1,2}$, ..., $[SEP]$, $S_{2,1}$, $S_{2,2}$, ..., $[SEP]$ where $S_{i,j}$ and $U_{i,j}$ are the jth token of the system and user utterance in the ith turn. Following BERT, we also add a learned embedding to every token indicating whether it comes from user utterances or system utterances . In addition, since the disengaging class and the non-disengaging class are imbalanced, we up-sample the disengaging dialog turns for both the training set and the development set. Though it is also possible to handle the imbalanced classes by adding weights for two classes, we did not take this approach because we do not have a closed-form solution for calculating the shapley value for weighted KNN in $O(N \log N)$ time. Improving the architecture of HERALD and extending HERALD to other machine learning tasks (Liang and Zou, 2021; Liang et al., 2020d,b, 2021) are interesting directions of future work.

A.2 Reproducibility

The source code of HERALD can be found in the supplementary materials. We run experiments on a server of eight GTX 1080 GPUs. The average runtime for all stages of HERALD is less than 10 minutes. The number of parameters is similar to BERT. We use the default hyperparameters of BERT. The public examples of Google Meena Dataset can be downloaded from `https://github.com/google-research/google-research/blob/master/meena/meena.txt` The public examples of Facebook Blender Dataset can be downloaded from `https://parl.ai/projects/recipes/chatlog_2.7B_render.html` The public examples of ConvAI2 Dataset can be downloaded from `http://convai.io/data/data_volunteers.json` and `http://convai.io/data/summer_wild_evaluation_dialogs.json`
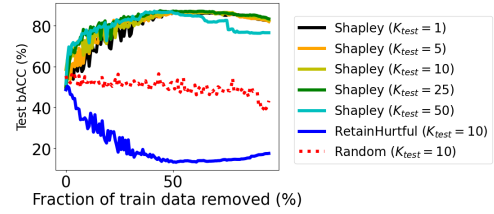
(a) Denoising with Shapley Value in Gunrock Movie Dataset
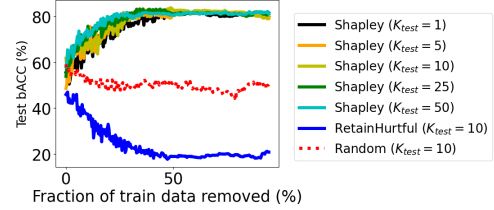
(b) Denoising with Shapley Value in ConvAI2 Dataset

Figure 6: Removing data points with low Shapley value improves the performance of the KNN classifier.

Additional Shapley Value Analysis We also present addition analysis to show how Shapley denoising works as shown in Figure 6. We present the experiments on both Gunrock Movie Dataset and ConvAI2 Dataset. Figure 6 presents a quantitative analysis of Shapley value. According to the Shapley value, we remove data points one by one starting from the least valuable to the most valuable. Each time, after the data point is removed, we create new KNN classifier models on the remaining dialog turns and labels and evaluate them on the test set with expert annotations. As shown in Figure 6, removing training data with low Shapley values increases the performance to a certain point before convergence for K of all choices. We observe a similar trend when re-training a model on the remaining data. In contrast, removing data randomly or removing data from the most to least valuable data decreases the performance on the test set. This shows that low Shapley value data effectively capture outliers and corruptions, which further justifies our design choice of denoising with Shapely value.

A.3 Addition Dialog Examples

We show additional dialog examples. Figure 7 shows a full dialog example from ConvAI dataset. Figure 8 shows a full dialog example from Gunrock Movie dataset.

System: Hi. Nice to meet you! Do you want to tell me your name?

User: Mike

System: Hi, Mike. How are you doing today?

User: drink a beer and having a good time

System: Nice! I'm glad you're having a good day! I'm curious, What do you like to do in your free time?

User: say that again

System: I forgot to tell you! I am a huge fan of movies. Do you enjoy watching movies?

User: yes i do

System: I'm thrilled to talk about movies. Have you seen any films recently ?

User: yeah i saw the new

System: Nice. What movie did you see?

User: of the one with the vin diesel

System: Actors can make or break a movie. Do you want to talk more about Vin Diesel?

User: yeah

System: Ok, let's talk about Vin Diesel. I've heard someone describe acting as behaving truthfully under imaginary circumstances. How do you feel about Vin Diesel's acting?

User: you can't really agree you know

System: Okay. For me, I usually enjoy Vin Diesel's acting. So, I read this fact. Vin Diesel broke into a Theater in New York with his brother when he was 7 with the intent of vandalizing it. When they were caught, instead of calling the police, the theater's artistic director offered them roles in an upcoming show, thus starting Diesel's acting career. Any thoughts?

User: he's a lucky son of a gun

System: I think that when people are really famous, it's interesting how they behave. What do you think about Vin Diesel as a person?

User: i don't know

System: Ok. So, I heard this fact. Vin Diesel has played Dungeons and Dragons for 20 years, and even had a fake tattoo of his character's name on his stomach while filming. Can you believe that?

User: yeah can

System: I don't know much more about Vin Diesel. So, what movie with Vin Diesel do you like best?

User: the uhhh fast and the furious movies

System: Oh, I know about The Fate of the Furious! hmm Did you like the movie?

User: yes i like action movies

System: Cool! I enjoyed watching the movie too. I heard this fact about the movie. When the crew is shown the garage of all of the cars including the army tank, you can see Paul Walker's GTR that his character drove in Furious 7 2015 up on top of one of the three-tier car racks. A subtle tribute to the late actor. What do you think?

User: it's cool

System: Glad you like it! The whole time I was on the edge of my seat waiting to see what would happen next. What else do you like about the movie?

User: computer we're done shut down

Figure 8: A full example from Gunrock Movie Dataset.

User: Hey!

System: hello there how are you? do you like dogs?

User: Good ! I like dogs, I grew up in a farm

System: that is cool. i've a german shepard.

User: what is your dog's name ?

System: his name is pepper. do you have any pets?

User: I used to have a dog but he died....

System: that is so sad! do you watch nhl hockey?

User: no... not a big fan of sport ?

System: no, i love watching it. i love the nhl.

Figure 7: A full example from ConvAI Dataset.