A Scalable Framework for Learning From Implicit User Feedback to Improve Natural Language Understanding in Large-Scale Conversational AI Systems

Sunghyun Park*, Han Li*, Ameen Patel, Sidharth Mudgal, Sungjin Lee, Young-Bum Kim, Spyros Matsoukas, Ruhi Sarikaya
Amazon Alexa AI
{sunghyu, lahl, paameen, sidmsk, sungjinl, youngbum, matsouka, rsarikay}@amazon.com

Abstract

Natural Language Understanding (NLU) is an established component within a conversational AI or digital assistant system, and it is responsible for producing semantic understanding of a user request. We propose a scalable and automatic approach for improving NLU in a large-scale conversational AI system by leveraging implicit user feedback, with an insight that user interaction data and dialog context have rich information embedded from which user satisfaction and intention can be inferred. In particular, we propose a domain-agnostic framework for curating new supervision data for improving NLU from live production traffic. With an extensive set of experiments, we show the results of applying the framework and improving NLU for a large-scale production system across 10 domains.

1 Introduction

For a conversational AI or digital assistant system (Kepuska and Bohouta, 2018), Natural Language Understanding (NLU) is an established component that produces semantic interpretations of a user request, which typically involves analysis in terms of domain, intent, and slot (El-Kahky et al., 2014). For instance, the request *"Play a song by Taylor Swift"* can be interpreted as falling within the scope of *Music* domain with *Play Song* intent and *Taylor Swift* identified for *Artist* slot.

Without an accurate semantic understanding of the user request, a conversational AI system cannot fulfill the request with a satisfactory response or action. As one of the most upstream components in the runtime workflow (Sarikaya, 2017), NLU's errors also have a wider blast radius that propagate to all subsequent downstream components, such as dialog management, routing logic to back-end applications, and language generation.

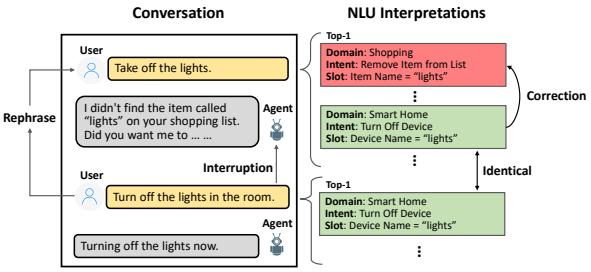

Figure 1: An example of implicit user feedback, specifically an indication of user dissatisfaction and user rephrase behavior, that can be used to create new supervision data to correct NLU errors. The left side shows the dialog history and the right side shows the ranked NLU interpretations for each user request.

A straight-forward way to improve NLU is through human annotations, but they are labor-intensive and expensive. Such annotations require at least multiple tiers of annotations (e.g., end user experience, error attribution, and semantic interpretation), and it is hard to consider all relevant contextual conditions. They are also limited by the existing annotation guidelines that may be outdated or that may not accurately reflect user expectations. Due to these limitations, leveraging user feedback, both implicit and explicit, from real production systems is emerging as a new area of research.

Our work makes three main contributions. First, this work is the first in the literature to introduce a scalable, automatic and domain-agnostic approach for leveraging implicit user feedback to continuously and directly improve the NLU component of a large-scale conversational AI system in production. This approach can be applied week over week to continuously and automatically improve NLU towards better end-to-end user experience, and given that no human annotation is required, the approach also raises minimal user privacy concerns. Our approach of using implicit feedback is based on our insight that user interaction data and dialog context have rich information embedded from which user

* Equal contribution.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6054–6063
November 7–11, 2021.

satisfaction and intention can be inferred (see Figure 1). Second, we propose a general framework for curating supervision data for improving NLU from live traffic that can be leveraged for various subtasks within NLU (e.g., domain/intent classification, slot tagging, or cross-domain ranking). Last, we show with an extensive set of experiments on live traffic the impact of the proposed framework on improving NLU in the production system across 10 widely used domains.

2 Background and Problem Definition

The NLU component typically has three main types of underlying models - domain classifiers, intent classifiers, and slot taggers (El-Kahky et al., 2014). The three modeling tasks can be treated independently (Gao et al., 2018) or as a joint optimization task (Liu and Lane, 2016; Hakkani-Tür et al., 2016), and some systems have a model to rank across all domains, intents and slots on a certain unit of semantic interpretation (Su et al., 2018).

Leveraging implicit feedback from the users has been widely studied in the context of recommendation systems (Hu et al., 2008; He et al., 2016; Liu et al., 2010; Loni et al., 2018; Rendle et al., 2012; He and McAuley, 2016; He et al., 2016; Wang et al., 2019) and search engines (Joachims, 2002; Sugiyama et al., 2004; Shen et al., 2005; Bi et al., 2019). In such systems, common types of implicit user feedback explored include a history of browsing, purchase, click-through behavior, as well as negative feedback. Leveraging implicit feedback in the context of conversational AI systems is relatively unexplored, but it has been applied for rewriting the request text internally within or post the Automatic Speech Recognition (ASR) component (Ponnusamy et al., 2019), improving the Natural Language Generation component (Zhang et al., 2018), and using user engagement signals for improving the entity labeling task specifically focused on Music domain (Muralidharan et al., 2019). We note that compared to explicit feedback (Petrushkov et al., 2018; Iyer et al., 2017), using implicit feedback is more scalable and does not introduce friction in user experience. But it comes with a challenge of the feedback being noisy, and leveraging the feedback is more difficult when there is no sufficient data such as for improving tail cases (Wang et al., 2021a,b).

In this paper, we specifically focus on two types of implicit user feedback - *dissatisfaction* of experience (to understand what to fix, e.g., users prematurely interrupting a system's response) and *clarification* of intention through rephrase (to understand how to fix, e.g., users clarifying their requests by rephrasing the previous request in simpler terms). In this work, we assume that there are mechanisms already in place to automatically (1) infer user dissatisfaction (f_{defect} in Section 2.3) and also (2) detect whether a given request is a rephrase of a previous request ($f_{rephrase}$ in Section 3). There are many ways to build these two mechanisms, either rule-based or model-based. Due to space limitation, we leave more details of the two mechanisms outside the scope of this paper. For completeness and better context to the reader however, we briefly describe various ways to build them, which would be straight-forward to adapt and implement.

2.1 User Dissatisfaction Detection

Unless we specifically solicit users' feedback on satisfaction after an experience, user feedback is mostly implicit. There are many implicit user behavior signals that can help with detecting user dissatisfaction while interacting with a conversational AI system. They include termination (stopping or cancelling a conversation or experience), interruption (barging in while the system is still giving its response), abandonment (leaving a conversation without completing it), error-correcting language (preceding the follow-up turn with "no, ..." or "I said, ..."), negative sentiment language showing frustration, rephrase or request reformulation, and confirmation to execute on an action (Beaver and Mueen, 2020; Sarikaya, 2017).

Although not strictly from the user behavior, there are other signals from the system action and response that are also useful. They include generic error-handling system responses ("I don't know that one."), the templates executed for generating natural language error-handling responses (the song entity is not found for playing music), and the absence of a response (Beaver and Mueen, 2020; Sarikaya, 2017). There are also component-level signals such as latency or low confidence scores for the underlying models within each component such as ASR or NLU.

For more advanced approaches, we can combine the signals from the user behavior and the system together, try to model user interaction patterns, and use additional context from past interaction history beyond immediate turns (Jiang et al., 2015; Ultes

and Minker, 2014; Bodigutla et al., 2020). Furthermore, user satisfaction can depend on usage scenarios (Kiseleva et al., 2016), and for specific experiences like listening to music, we can adapt related concepts such as dwell time in the search and information retrieval fields to further fine-tune.

2.2 User Rephrase Detection

There are many lines of work in the literature that are closely related to this task under the topics of text/sentence semantic similarity detection and paraphrase detection. The approaches generally fall into lexical matching methods (Manning and Schutze, 1999), leveraging word meaning or concepts with a knowledge base such as WordNet (Mihalcea et al., 2006), latent semantic analysis methods (Landauer et al., 1998), and those based on word embeddings (Camacho-Collados and Pilehvar, 2018) and sentence embeddings (Reimers and Gurevych, 2019). In terms of modeling architecture, Siamese network is common and has been applied with CNN (Hu et al., 2014), LSTM (Mueller and Thyagarajan, 2016), and BERT (Reimers and Gurevych, 2019). The task is also related to the problems in community question-answering systems for finding semantically similar questions and answers (Srba and Bielikova, 2016).

2.3 Problem Definition

Denote $\mathcal{T} = (\Sigma, \Pi, N, A)$ to be the space of all user interactions with a conversational AI system with each request or turn $t_i = (u_i, p_i, c_i, a_i) \in \mathcal{T}$ consisting of four parts: $u_i \in \Sigma$ is the user request utterance, $p_i \in \Pi$ is the semantic interpretation for u_i from NLU, $c_i \in N$ is the contextual metadata (e.g., whether the device has a screen), and $a_i \in A$ is the system action or response. Here, we are proposing a general framework that allows a scalable and automatic curation of supervision data to improve NLU, and we keep the unit of the semantic interpretation abstract for generalizability, which can be for one or a combination of NLU subtasks of domain classification, intent classification, and slot tagging. For instance, one possible interpretation unit would be domain-intent-slots tuple, which is what we use in our experiments described in Section 4. Although we only focus on NLU in this paper, the approach here can be extended to improve other components in a conversational AI system such as skill routing (Li et al., 2021).

We define a *session* of user interaction $s = \{t_1, t_2, \ldots, t_q\} \subseteq \mathcal{T}$ which is a list of time-consecutive turns by the same user. Denote m_t to be the NLU component at timestamp t. We collect the interaction session data $\mathcal{S}_{live} = \{s_1, s_2, \ldots, s_n\}$ from live traffic for a certain period of time Δ (e.g., one week) starting at time t, from which we curate new supervision data to produce $m_{t+\Delta}$ with improved performance. Specifically, given a tool f_{defect} for automatic analysis of user dissatisfaction for each turn, we process $\mathcal{S}_{live}$ to identify all turns that indicate user dissatisfaction, $t_i \in \mathcal{D}_{defect}$, which we call a *defective turn* or simply a *defect*. The key challenges then are how to (1) identify *target defects* which are high-confidence defects that can be targeted by NLU (i.e., there is sufficient disambiguation power within NLU that it can learn to produce different results if given specific supervision) and that are likely causing repeated and systematic dissatisfaction of user experience, and (2) find a likely better interpretation for the *target defects* to change system action or response that leads to user satisfaction.

3 Solution Framework

The framework involves two deep learning models - *Defect Identification Model (DIM)* for addressing the first challenge of identifying *target defects* and *Defect Correction Model (DCM)* for the second challenge of correcting them by automatically labeling them with a likely better semantic interpretation (see Figure 2). It is straight-forward to apply DIM and DCM on the production traffic log to curate new supervision data for improving NLU.

Data Preparation: We collect the user interaction session data from the production log $\mathcal{S}_{live}$ for an arbitrary period of time (e.g., past one week). Given a user dissatisfaction analysis tool f_{defect} and a rephrase analysis tool $f_{rephrase}$, we tag $t_j \in s_i$ as a defect if f_{defect} detects user dissatisfaction for the turn and we tag $t_j \in s_i$ as a rephrase if there exists $t_i \in s_i$ where $j > i$ (i.e., temporally t_j occurred after t_i) and $f_{rephrase}$ detects t_j to be a rephrase of t_i. We then extract each turn in $\mathcal{S}_{live}$ to create turn-level data $\mathcal{D}_{live} = \{t_j \in s_i \mid s_i \in \mathcal{S}_{live}\}$ with t_j containing two binary labels of defect e_d and rephrase e_r.

3.1 Defect Identification Model (DIM)

We define DIM as $f_{dim} : \mathcal{T} \rightarrow \{0, 1\}$, which takes as input each turn $t_i \in \mathcal{D}_{live}$ and outputs whether t_i is a *target defect* or not. It uses the same contextual

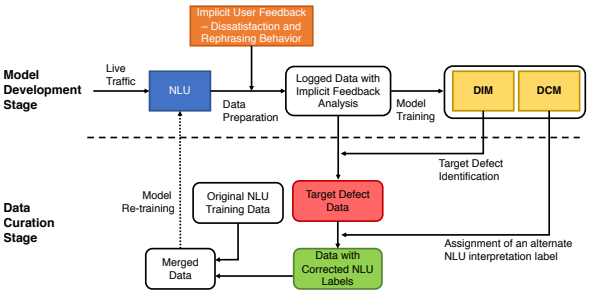

Figure 2: Our framework for leveraging implicit user feedback to automatically curate supervision data for improving NLU, consisting of *Defect Identification Model (DIM)* and *Defect Correction Model (DCM)*.

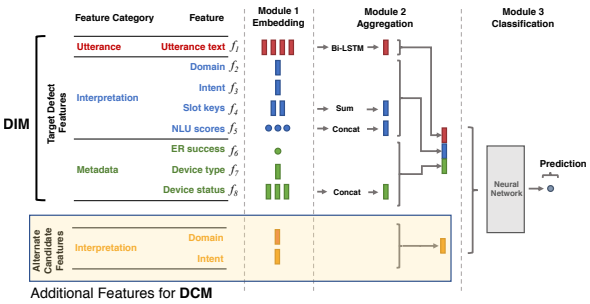

Figure 3: The model architectures for Defect Identification Model (DIM) and Defect Correction Model (DCM). For DIM, the prediction is for target defect probability, and for DCM, it is for correction probability (i.e., whether the alternate domain and intent is a good alternate ground-truth label).

features (and architecture) as the underlying individual NLU model we wish to improve and uses the results of f_{defect}, or e_d, as the ground-truth labels for training. This allows us to filter down the defects into those that can be targeted by the NLU model of interest (since the same features could predict the defects, suggesting enough disambiguation capacity). By tuning the probability threshold used for binary model prediction, we can further reduce noise in defects and focus on more high-confidence defects that are repeated and systematic failures impacting the general user population.

Figure 3 shows an example DIM architecture for a cross-domain interpretation re-ranking model (more detail in 4.1). The model architecture consists of three main modules: *embedding*, *aggregation*, and *classification*. Given each feature f_j extracted from t_i, the embedding module H_{emb} converts f_j into an embedding. For each sequential or categorical feature f_j, denoting $\mathbf{w}_{f_j,t_i}$ as the value of f_j with m tokens (where m=1 for categorical), we generate $\mathbf{v}_{f_j,t_i} = H_{emb}(\mathbf{w}_{f_j,t_i}) \in \mathbb{R}^{m \times d_{f_j}}$ with each token converted into the d_{f_j}-dimensional

Algorithm 1 DIM threshold determination.

procedure THRESSEARCH(f_{dim}, $\mathcal{D}_{valid}$, λ, ϵ)
  low, high $\leftarrow$ 0, 1
  while | low - high | $> \epsilon$ **do**
    $\tau \leftarrow$ (low + high) /2
    $\mathcal{P}_{valid} \leftarrow \{t_i \mid f_{dim}(t_i) > \tau, \forall t_i \in \mathcal{D}_{valid}\}$
    $\alpha \leftarrow$ PREDICTIONACCURACY($\mathcal{P}_{valid}$)
    if $\alpha < \lambda$ **then** low $\leftarrow \tau$
    else high $\leftarrow \tau$
  return τ

embedding. For each numerical feature, we have $\mathbf{v}_{f_j,t_i} = \mathbf{w}_{f_j,t_i}$ as each feature is already represented by numeric values. The aggregation module H_{agg} then converts $\mathbf{v}_{f_j,t_i}$ of each feature f_j to an aggregation vector $\mathbf{u}_{f_j,t_i}$ that summarizes the information of $\mathbf{v}_{f_j,t_i}$. Based on the feature type, H_{agg} applies different aggregation operations. For example, we apply a Bi-LSTM (Schuster and Paliwal, 1997) to the utterance text embeddings $\mathbf{v}_{f_1,t_i}$ to capture the word context information. Finally, the classification module H_{cls} takes as input all aggregation vectors to make a prediction whether t_i is a target defect or not. Specifically, we first concatenate all aggregation vectors to get a summarization vector $\mathbf{u}_{t_i} = \bigoplus_{f_j} \mathbf{u}_{f_j,t_i}$. Then, a two-layer highway network (Srivastava et al., 2015) is applied to $\mathbf{u}_{t_i}$ to make a binary prediction. The model is trained using binary cross-entropy loss.

When developing DIM, we split $\mathcal{D}_{live}$ into the training set $\mathcal{D}_{train}$ and the validation set $\mathcal{D}_{valid}$ with a ratio of 9:1. Once we have DIM trained with $\mathcal{D}_{train}$, we use $\mathcal{D}_{valid}$ to further tune the prediction probability threshold used to extract target defects from all defects tagged by f_{defect}. Specifically, for each turn $t_i \in \mathcal{D}_{defect}$, we pass it to f_{dim} to get the confidence score $o_i = f_{dim}(t_i)$ of being a defect. Then, we generate the target defect set $\mathcal{D}_{target} = \{t_i \mid o_i > \tau\}$, i.e., we collect all turns satisfying the defect prediction confidence being greater than a threshold τ. In order to select the value for τ, we perform a binary search on $\mathcal{D}_{valid}$ as shown in Algorithm 1, which takes as inputs two additional parameters λ (to set the minimum prediction accuracy we want) and ϵ.

3.2 Defect Correction Model (DCM)

We define DCM as $f_{dcm} : \mathcal{T} \times \Pi \rightarrow \{0, 1\}$, which takes as input a pair (t_i, p_j) with $t_i \in \mathcal{D}_{live}$ and $p_j \in \Pi$ to make a prediction whether p_j is a proper semantic interpretation for t_i. As the space of the semantic interpretation Π is too large, we can make the process more efficient by restricting to

find a better interpretation in the k-best predictions $P_i^k \subseteq \Pi$ (i.e., k interpretations with the highest prediction confidence) by the NLU model of interest. Note that it is not difficult to force more diversity into the k-best predictions by only allowing top predictions from each domain or intent. For training, we leverage rephrase information from the logged data to automatically assign a corrected semantic interpretation as the new ground-truth label for the defects, with the following assumption: Given a pair of turns t_i and t_j, if (a) the utterance of t_j rephrases the utterance of t_i in the same session and (b) t_j is non-defective, then the semantic interpretation of t_j is also the correct interpretation for t_i.

Following the example DIM architecture for the cross-domain interpretation re-ranking model in Figure 3, the DCM architecture extends that of DIM with the main difference that we can generate other features based on domain, intent and slot information from p_j. To obtain the training data, we first examine all turns in $\mathcal{D}_{live}$ to generate the high value set $\mathcal{D}_h \subseteq \mathcal{T} \times \mathcal{T}$. Each instance $(t_i, r_i) \in \mathcal{D}_h$ is a pair of turns satisfying (a) $t_i \in \mathcal{D}_{live}$ is a defect and (b) $r_i \in \mathcal{D}_{live}$ is a non-defective rephrase of t_i in the same session (defects and rephrases are described in Section 2.3 and Section 3:Data Preparation). We then generate the training data $\mathcal{D}_{train}$ using the high value set $\mathcal{D}_h$. Specifically, for each pair $(t_i, r_i) \in \mathcal{D}_h$, we generate k training instances as follows. First, we get the k-best interpretations $P_{r_i}^k$ of r_i. Then, we pair t_i with each candidate $p_j \in P_{r_i}^k$ to get a list of tuples $(t_i, p_1), (t_i, p_2), \ldots, (t_i, p_k)$. Next, we expand each tuple (t_i, p_j) by assigning a label c indicating whether p_j can be a proper interpretation for t_i. Denote $p^* \in P_{r_i}^k$ as the correct interpretation for r_i, assumed since it is executed without a defect (note that the top-1 interpretation is not necessarily the executed and correct one, although it is most of the time). We generate one positive instance $(t_i, p^*, c = 1)$, and $k-1$ negative instances $\{(t_i, p_j, c = 0) \mid p_j \in P_{r_i}^k \wedge p_j \neq p^*)\}$. Only using the k-best interpretations from r_i to generate $\mathcal{D}_{train}$ may not be sufficient, as in practice the value k is small and many interpretations observed in real traffic does not appear in the training data. To make the model generalize better, we augment the training data by injecting random noise. For each pair $(t_i, r_i) \in \mathcal{D}_h$, in addition to the $k-1$ generated negative instances, we randomly draw q interpretations $P_{noise}^q = \{p_1^n, p_2^n, \ldots, p_q^n\} \subseteq \Pi$ that are not in $P_{r_i}^k$, and we generate q new negative instances $\{(t_i, p_j^n, c = 0) \mid p_j^n \in P_{noise}^q\}$. In short, DCM's role is to find the most promising alternate interpretation in t_i's k-best interpretation list given that t_i is a defect.

Domain	Total	W	L	T	O	Δ_1	Δ_2(%)
Overall*	2,000	367	196	1,412	25	171	8.5
Knowledge*	200	77	25	98	0	52	26.0
MyTasks*	200	82	36	73	9	46	23.0
Multimedia-2*	200	59	39	100	2	20	10.0
Help*	200	42	22	134	2	20	10.0
Multimedia-3*	200	34	19	146	1	15	7.5
ChitChat	200	29	22	149	0	7	3.5
DeviceControl	200	8	4	187	1	4	2.0
SmartHome	200	14	10	172	4	4	2.0
Shopping	200	9	7	183	1	2	1.0
Multimedia-1	200	13	12	170	5	1	0.5

Table 1: Overall side-by-side win-loss evaluation results across 10 domains, comparing the top interpretation prediction between the baseline NLU and the updated NLU improved with our framework. "W," "L," "T" and "O" represent "Win," "Loss," "Tie" and "Others" respectively. A win means that the updated NLU produced a better top interpretation than the baseline (* denotes statistical significance at p<.05).

New Supervision Data Curation: Once we have f_{dcm} trained, the last step of the framework is to curate new supervision data by applying f_{dcm} to each turn $t_i \in \mathcal{D}_{target}$ identified by f_{dim} and automatically assigning a better semantic interpretation for correction. Specifically, we pair each turn $t_i \in \mathcal{D}_{target}$ with every interpretation candidate $p_j \in P_i^k$ as the input to f_{dcm}. The interpretation with the highest score $p^* = \arg\max_{p_j \in P_i^k} f_{dcm}(t_i, p_j)$ is used as the corrected interpretation for t_i.

4 Experiment Results and Discussion

4.1 Experiment Methodology

Dataset and Experiment Settings: Given a baseline NLU in production, m_{base}, which produces a ranked list of interpretations with each interpretation comprising domain-intent-slots tuple, we inject a re-ranking subtask at the very last layer of the NLU workflow to build an improved NLU, m_{new}. We call the subtask re-ranking because it takes in an already ranked list (i.e., the output of m_{base}) and makes a final adjustment. We leverage the new supervision data obtained through our framework to train the re-ranking model for improv-

Domain	ASR Error		NLU Error		Bad Response		Others		NLU Error Δ
	DEF	DIM	DEF	DIM	DEF	DIM	DEF	DIM	
Overall*	29.8	29.1	14.3	39.0	24.8	11.3	31.1	20.6	24.7
SmartHome*	29.0	22.0	0.0	34.0	47.0	13.0	24.0	31.0	34.0
DeviceControl*	13.0	36.0	7.0	41.0	20.0	12.0	60.0	11.0	34.0
Multimedia-3*	41.0	30.0	21.0	52.0	20.0	10.0	18.0	8.0	31.0
Knowledge*	36.0	23.0	16.0	45.0	13.0	5.0	35.0	27.0	29.0
Multimedia-1*	37.0	34.0	6.0	33.0	38.0	20.0	19.0	13.0	27.0
MyTasks*	24.0	46.0	16.0	42.0	6.0	3.0	54.0	9.0	26.0
ChitChat*	32.0	13.0	3.0	29.0	7.0	15.0	58.0	43.0	26.0
Multimedia-2*	24.0	16.0	43.0	67.0	22.0	8.0	11.0	9.0	24.0
Shopping	31.0	44.0	11.0	19.0	35.0	7.0	23.0	30.0	8.0
Help	31.0	27.0	20.0	28.0	40.0	20.0	9.0	25.0	8.0

(a)

Domain	Total	W	L	T	U	Δ_1	Δ_2(%)
Overall*	1,000	399	77	365	159	322	32.2
Multimedia-2*	100	82	3	12	3	79	79.0
Multimedia-3*	100	61	0	31	8	61	61.0
Knowledge*	100	56	7	23	14	49	49.0
Multimedia-1*	100	40	3	48	9	37	37.0
MyTasks*	100	41	9	26	24	32	32.0
ChitChat*	100	34	12	46	8	22	22.0
Help*	100	41	20	25	14	21	21.0
SmartHome*	100	25	7	56	12	18	18.0
Shopping*	100	10	2	53	35	8	8.0
DeviceControl	100	9	14	45	32	-5	-5.0

(b)

Table 2: (a) The analysis of DIM through error attribution annotations between the defects in the production traffic vs. the target defects identified by DIM. The numbers are in percentage. (b) The analysis of DCM through win-loss annotations between the top interpretation produced by the baseline NLU and the new interpretation label assigned by DCM. Statistical significance at p<.05 is noted with *, specifically on the NLU errors in (a).

ing the overall NLU performance. Figure 4 shows the model architecture of the re-ranker, which is a simple extension of the DIM architecture, and it learns from the new supervision data when to top-rank a better interpretation that is not at the top of the list (trained with sigmoid activation functions at the output layer and binary cross-entropy loss). We note here that the specific model architecture is not as important as the new supervision data obtained through our framework that is the key for bringing NLU improvements. This experiment setup is appealing in that it is straightforward and simple, especially in the production setting. First, NLU consists of many domain-specific models that are spread out to multiple teams, making it difficult to coordinate leveraging the new supervision data for improvement across multiple domains. Second, working with the final re-ranking model allows us to improve NLU performance domain-agnostically without needing to know the implementation details of each domain. Third, it is easier to control the influence of the new supervision data since we need to manage only one re-ranking component.

Given sampled and de-identified production traffic data from one time period $\mathcal{D}_{period1}$, which have been analyzed by f_{defect} and $f_{rephrase}$[1], we first train DIM according to Section 3.1, with over 100MM training instances from $\mathcal{D}_{period1}$ and over 10MM defects identified by f_{defect}. Then, we extract over 8MM high-value rephrase pairs (a defective turn and non-defective rephrase in the same session) from $\mathcal{D}_{period1}$ to train DCM according to Section 3.2. To train the re-ranker, we randomly sample over 10MM instances $\mathcal{D}_s \subseteq \mathcal{D}_{period1}$ and over 1MM defects identified by f_{defect}. We apply

[1]In today's production system, f_{defect} and $f_{rephrase}$ show F_1 scores over 0.70.

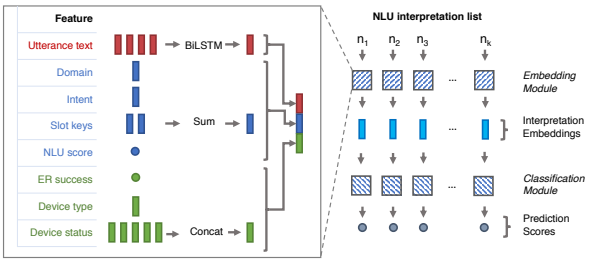

Figure 4: The model architecture for the re-ranker, which is a subtask we put at the last layer of the NLU to produce a better ranked list of interpretations.

the trained DIM to the sampled defects $\mathcal{F}_{def}$ that filters them down from over 1MM defects to over 300K target defects $\mathcal{F}_{dim}$ that the NLU re-ranker has sufficient features to target and produce different results. Then, all target defects $\mathcal{F}_{dim}$ are assigned a new ground-truth interpretation label by the trained DCM (note that not all defects have corresponding non-defect rephrases, hence the value of DCM for finding the most promising alternate interpretation from the ranked list), which serve as the new curated supervision for building m_{new}, while the rest of the non-defective instances keep the top-ranked interpretation as the ground-truth label. In other words, most of the instances in $\mathcal{D}_s$ are used to replicate the m_{base} results (a pass-through where the same input ranked list is outputted without any change), except for over 300K (over 3% of the total training data) that are used to revert the ranking and put a better interpretation at the top.

Overall Side-by-Side Evaluation: The overall performance between m_{base} and m_{new} was compared on another sampled production traffic from non-overlapping time period $\mathcal{D}_{period2}$ in a shadow evaluation setting, in which the traffic flowing through m_{base} was duplicated and simultaneously

sent to m_{new} that is deployed to the same production setting as m_{base} but without end-user impact. Both m_{base} and m_{new} produced the same ranked list of interpretations for over 99% of the time. Note that this is by design since incremental improvements are preferred in production systems without drastically changing the system behavior and that our approach can be applied continuously, week over week (changing the proportion of the new supervision data will have an impact on the replication rate). Furthermore, even 1% change in the overall system behavior has a huge impact at the scale of tens of million of requests per week in a large-scale production system. We performed win-loss annotations on the deltas (when m_{base} and m_{new} produced different results) with in-house expert annotators who follow an established NLU annotation guideline to make a side-by-side evaluation whether m_{new} produced a better interpretation (i.e., win) on the top compared to m_{base} or not (N = 12, agreement = 80.3%, Cohen's kappa = 0.60 indicating moderate agreement; note that the annotators are trained to reach agreement level that is practical given the high complexity of the NLU ontology). We randomly sampled 200 such requests per domain that produced different results[2].

DIM Analysis: We randomly sampled 100 defects per domain from $\mathcal{F}_{def}$ and $\mathcal{F}_{dim}$ respectively and performed error attribution annotations (i.e., *ASR error* for mis-transcribing *"play old town road"* to *"put hotel road"*, *NLU error* for misinterpreting *"how do I find a good Italian restaurant around here"* to *Question Answering* intent instead of *Find Restaurant* intent, *Bad Response* for having a correct interpretation that still failed to deliver a satisfactory response or action, and *Others* for those that the annotators could not determine due to lack of context or additional information; N = 12, agreement = 71.3%, Cohen's kappa = 0.63 indicating substantial agreement).

DCM Analysis: We perform the same win-loss annotations as described in overall shadow evaluation on 100 random samples per domain, specifically on the curated supervision data $\mathcal{F}_{dim}$ with new ground-truth assigned by DCM.

Training Setup: All the models were implemented in PyTorch (Paszke et al., 2019) and trained and evaluated on AWS p3.8xlarge instances with Intel Xeon E5-2686 CPUs, 244GB memory, and 4 NVIDIA Tesla V100 GPUs. We used Adam (Kingma and Ba, 2014) for training optimization, and all the models were trained for 10 epochs with a 4096 batch size. All three models have around 12MM trainable parameters and took around 5 hours to train.

[2] A/B testing results on around 20 intents with over 100MM live utterances showed improvement in reducing defect ratio (i.e., the ratio of utterances tagged by f_{defect}) end-to-end from 72.9% to 42.2% on the deltas (statistically significant at p<.05).

4.2 Results and Discussions

Overall Side-by-Side Evaluation: Table 1 shows the overall shadow evaluation results, making NLU-level comparison between m_{base} and m_{new}. The column *Total* shows the number of requests annotated per domain. The columns *Win*, *Loss*, and *Tie* show the number of requests where m_{new} produced better, worse, and comparable NLU interpretations than m_{base} respectively. The column *Others* shows the number of requests where the annotators could not make the decision due to lack of context. The column Δ_1 shows the difference in the number of win and loss cases, and Δ_2 shows the relative improvement (i.e., Δ_1 / *Total* in percentage). First, we note that m_{new} overall produced a better NLU interpretation on 367 cases while making 196 losses, resulting in 171 absolute gains or 8.5% relative improvement over m_{base}. This indicates that applying our framework can bring a net overall improvement to existing NLU. Second, analyzing per-domain results shows that m_{new} outperforms m_{base} (7.5-26.0% relative improvements) on 5 domains, while making marginal improvements (0.5-3.5% improvements) on the other 5 domains.

Analysis on DIM: Table 2.(a) summarizes the results of error attribution annotations between the defects in the production traffic (denoted as *DEF*) and target defects identified by DIM (denoted as *DIM*). The results show that the target defects identified by DIM help us focus more on the defects that are caused by ASR or NLU (the ones that can be targeted and potentially fixed, specifically *NLU Error* which is at 39.0% of total for *DIM* compared to 14.3% for *DEF*) and filter out others (*Bad Responses* and *Others*). Per-domain results show that the target defects identified by DIM consistently have a higher NLU error ratio than that of original defects for all domains.

Analysis on DCM: Table 2.(b) summarizes the win-loss annotation results on the new supervision data that take target defects from DIM and assign

Example	*Dialog*	*Interpretation by* m_{base}	*Interpretation by* m_{new}
Win_1	U_1: ***Show me d. c. universe.*** A_1: *I don't see a title named d. c. universe.* U_2: *I want to watch d. c. universe.* A_2: *Here is what I found ...*	Domain: Multimedia-3 Intent: Play Slots: TitleName → d. c. universe	Domain: Multimedia-2 Intent: Play Slots: TitleName → d. c. universe
Win_2	U_1: ***What's the weather at Mission Beach today?*** A_1: *There aren't any missions near Long Beach, California.* U_2: *What's the weather in San Diego?* A_2: *Currently, 69 degrees with clear skies and sun.*	Domain: Knowledge Intent: QA Slots: Question	Domain: Weather Intent: WeatherDetails Slots: Location → San Diego, CA
Loss_1	U_1: ***Play Wings of Fire.*** A_1: *I don't have a free content for that.* U_2: *Play Me.* A_2: *Me, by Taylor Swift.*	Domain: Multimedia-3 Intent: Play Slots: TitleName → Wings of Fire	Domain: Multimedia-1 Intent: Play Slots: TitleName → Wings of Fire
Loss_2	U_1: ***Is the Collard Festival going to happen today?*** A_1: *Sorry, I'm not sure.*	Domain: Knowledge Intent: QA Slots: Question	Domain: Calendar Intent: CheckCalendar Slots: Event → Collard Festival
Tie	U_1: ***Can you play the Baltimore Ravens?*** A_1: *Here is what I found ...*	Domain: Knowledge Intent: QA Slots: Question	Domain: Multimedia-1 Intent: Play Slots: TitleName → Baltimore Ravens

Table 3: Qualitative analysis comparing m_{base} and m_{new} in the overall side-by-side evaluation. For each example, the user request in bold is the turn for which the evaluation was performed. We show subsequent interaction dialog for context (U_* for user requests, A_* for system answers). The first two examples are "wins" (i.e., m_{new} better than m_{base}), followed by two "losses" (i.e., m_{new} worse than m_{base}), and a "tie" (i.e., m_{new} comparable to m_{base}).

new interpretation labels for correction with DCM. The results show that overall DCM correctly assigns a better, corrected NLU interpretation on 399 cases and fails on 77 cases, resulting in 322 absolute gains or 32.2% relative improvement. Per-domain results show that DCM consistently assigns a comparable or better interpretation on the target defects on almost all domains with a large margin (with 8.0%-79.0% relative improvements on 9 domains).

4.3 Qualitative Analysis

The first two examples in Table 3 are wins where m_{new} produced a better top interpretation than m_{base}. In *Win 1*, m_{base} produced an interpretation related to playing a title for a specific type of multimedia, while the user wanted to play the corresponding title in another multimedia type (e.g., music, video, or audio book). The updated NLU model m_{new} produced the correct interpretation, most likely having learned to favor a multimedia type depending on the context, such as device status (e.g., music or video currently playing or screen is on). Similarly in *Win 2*, m_{base} mis-interpreted the request as a general question due to not understanding the location "Mission Beach," which is corrected by m_{new}.

The next two examples are losses where m_{new} top-ranked incorrect interpretations such that they produced worse results than m_{base}. In *Loss 1*, the user is in the middle of trying out a free content experience for a specific multimedia type, and we suspect the reason m_{new} produced the incorrect interpretation is that there are similar requests in live traffic to "Play Wings of Fire" with another multimedia type, such that the model learns to aggressively top-rank the interpretations associated with a more dominant multimedia type. In *Loss 2*, the request is for a general event query in the area, and although the Q&A still failed to correctly answer, it was determined that it would be worse to fail in Calendar domain.

The last example is a "tie" where m_{new} and m_{base} both produced incorrect top interpretations that are equally bad in terms of user experience. Specifically, m_{base} mis-interpreted the request as a Q&A, while m_{new} mis-interpreted the meaning of "play" for playing multimedia instead of sports. As in *Loss 1*, We suspect many live utterances with the word "play" tend to be multimedia-related and biases DCM towards selecting multimedia-related interpretations.

From the qualitative analysis, especially losses, we observe that we can make our framework and new supervision data more precise if we consider more interaction history context spanning a longer period of time when we train DCM, use more signals such as personalization or subscription signals (for multimedia content types such as music or

audio book). Furthermore, for truly ambiguous requests, instead of aggressively trying to correct through a new interpretation, we could offer a better experience by asking a clarifying question.

5 Conclusion

We proposed a domain-agnostic and scalable framework for leveraging implicit user feedback, particularly user dissatisfaction and rephrase behavior, to automatically curate new supervision data to continuously improve NLU in a large-scale conversational AI system. We showed how the framework can be applied to improve NLU and analyzed its performance across 10 popular domains on a real production system, with component-level and qualitative analysis of our framework for more in-depth validation of its performance.

Acknowledgments

We thank Sergei Dobroshinsky, Nathan Eversole, Alex Go, Kerry Hammil, Archit Jain, Shubham Katiyar, Siddharth Mohan Misra, Joe Pemberton, and Steve Saunders for their active involvement and support for this work in the industry production system.

References

Ian Beaver and Abdullah Mueen. 2020. Automated conversation review to surface virtual assistant misunderstandings: Reducing cost and increasing privacy. In *AAAI Conference on Artificial Intelligence*.

Keping Bi, Choon Hui Teo, Yesh Dattatreya, et al. 2019. Leverage implicit feedback for context-aware product search. *arXiv preprint arXiv:1909.02065*.

Praveen Kumar Bodigutla, Aditya Tiwari, Spyros Matsoukas, et al. 2020. Joint turn and dialogue level user satisfaction estimation on multi-domain conversations. In *Conference on Empirical Methods in Natural Language Processing*.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Ali El-Kahky, Xiaohu Liu, Ruhi Sarikaya, et al. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *International Conference on Acoustics, Speech and Signal Processing*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational AI. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, et al. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Annual Conference of the International Speech Communication Association*.

Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI Conference on Artificial Intelligence*.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, et al. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Baotian Hu, Zhengdong Lu, Hang Li, et al. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *International Conference on Data Mining*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, et al. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Jiepu Jiang, Ahmed Hassan Awadallah, Rosie Jones, et al. 2015. Automatic online evaluation of intelligent assistants. In *International Conference on World Wide Web*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Veton Kepuska and Gamal Bohouta. 2018. Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home). In *Annual Computing and Communication Workshop and Conference*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Julia Kiseleva, Kyle Williams, Jiepu Jiang, et al. 2016. Understanding user satisfaction with intelligent assistants. In *ACM on Conference on Human Information Interaction and Retrieval*.

Thomas Landauer, Peter Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Han Li, Sunghyun Park, Aswarth Dara, et al. 2021. Neural model robustness for skill routing in large-scale conversational AI systems: A design choice exploration. *arXiv preprint arXiv:2103.03373*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *International Conference on Intelligent User Interfaces*.

Babak Loni, Martha Larson, and Alan Hanjalic. 2018. Factorization machines for data with implicit feedback. *arXiv preprint arXiv:1812.08254*.

Christopher Manning and Hinrich Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT press.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI Conference on Artificial Intelligence*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI Conference on Artificial Intelligence*.

Deepak Muralidharan, Justine Kao, Xiao Yang, et al. 2019. Leveraging user engagement signals for entity labeling in a virtual assistant. *arXiv preprint arXiv:1909.09143*.

Adam Paszke, Sam Gross, Francisco Massa, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*.

Pavel Petrushkov, Shahram Khadivi, and Evgeny Matusov. 2018. Learning from chunk-based feedback in neural machine translation. *arXiv preprint arXiv:1806.07169*.

Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, et al. 2019. Feedback-based self-learning in large-scale conversational AI agents. *arXiv preprint arXiv:1911.02557*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Conference on Empirical Methods in Natural Language Processing*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, et al. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34:67–81.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2005. Context-sensitive information retrieval using implicit feedback. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ivan Srba and Maria Bielikova. 2016. A comprehensive survey and classification of approaches for community question answering. *Transactions on the Web*, 10:1–63.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, et al. 2018. A re-ranker scheme for integrating large scale NLU models. In *Spoken Language Technology Workshop*.

Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. 2004. Adaptive web search based on user profile constructed without any effort from users. In *International Conference on World Wide Web*.

Stefan Ultes and Wolfgang Minker. 2014. Interaction quality estimation in spoken dialogue systems using hybrid-HMMs. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Cheng Wang, Sun Kim, Taiwoo Park, et al. 2021a. Handling long-tail queries with slice-aware conversational systems. *arXiv preprint arXiv:2104.13216*.

Cheng Wang, Sungjin Lee, Sunghyun Park, et al. 2021b. Learning slice-aware representations with mixture of attentions. *arXiv preprint arXiv:2106.02363*.

Haoyu Wang, Nan Shao, and Defu Lian. 2019. Adversarial binary collaborative filtering for implicit feedback. In *AAAI Conference on Artificial Intelligence*.

Wei-Nan Zhang, Lingzhi Li, Dongyan Cao, et al. 2018. Exploring implicit feedback for open domain conversation generation. In *AAAI Conference on Artificial Intelligence*.