Leveraging User Paraphrasing Behavior In Dialog Systems To Automatically Collect Annotations For Long-Tail Utterances

Tobias Falke, Markus Boese, Daniil Sorokin, Caglar Tirkaz and **Patrick Lehnen**
Amazon Alexa AI, Aachen, Germany
{falket,boesem,dsorokin,caglart,plehnen}@amazon.com

Abstract

In large-scale commercial dialog systems, users express the same request in a wide variety of alternative ways with a long tail of less frequent alternatives. Handling the full range of this distribution is challenging, in particular when relying on manual annotations. However, the same users also provide useful implicit feedback as they often paraphrase an utterance if the dialog system failed to understand it. We propose MARUPA, a method to leverage this type of feedback by creating annotated training examples from it. MARUPA creates new data in a fully automatic way, without manual intervention or effort from annotators, and specifically for currently failing utterances. By re-training the dialog system on this new data, accuracy and coverage for long-tail utterances can be improved. In experiments, we study the effectiveness of this approach in a commercial dialog system across various domains and three languages.

1 Introduction

A core component of voice- and text-based dialog systems is a language understanding component, responsible for producing a formal meaning representation of an utterance that the system can act upon to fulfill a user's request (Tur and De Mori, 2011). This component is often modeled as the combination of two tasks: intent classification (IC) – determining which intent from a set of known intents is expressed – and slot labeling (SL) – finding sequences of tokens that express slots relevant for the intent. As an example, consider the utterance *Play Blinding Lights*, which expresses the *PlayMusic* intent and the tokens *Blinding Lights* refer to the *Song* slot whereas *Play* expresses no slot.

As in many other language processing tasks, an important challenge arises from the fact that natural language allows one to express the same meaning in many different ways. In commercial systems at the scale of Apple's Siri, Amazon's Alexa or Google's Assistant, the variety of alternative utterances used to express the same request is immense, stemming from the scale of the system, the heterogenous language use among users and noise such as speech recognition errors (Muralidharan et al., 2019). In addition, the frequency distribution of alternative utterances follows a power law distribution, such that a system can handle a substantial part of the distribution by understanding just a few utterances, but needs to understand orders of magnitude more to also cover the long tail of the distribution. Although utterances from the tail occur rarely, humans have little difficulties understanding them, imposing the same expectation on a dialog system with true language understanding. However, the traditional way of building a language understanding component – supervised learning with manually annotated examples – is impractical to scale to that challenge as the necessary annotation effort becomes prohibitively expensive.

We propose to leverage implicit user feedback to reduce the need for manual annotation and thereby scale language understanding in dialog systems to more long-tail utterances. In this paper, we focus on cases where a user's utterance has not been correctly interpreted by the system, causing *friction* for the user, and the user gives implicit feedback by *paraphrasing* the utterance (see Figure 1). Such behavior is common in commercial systems, where many users make repeated attempts to receive a response. Predicted intents and slots of the eventually successful utterance can in these cases be used to infer labels

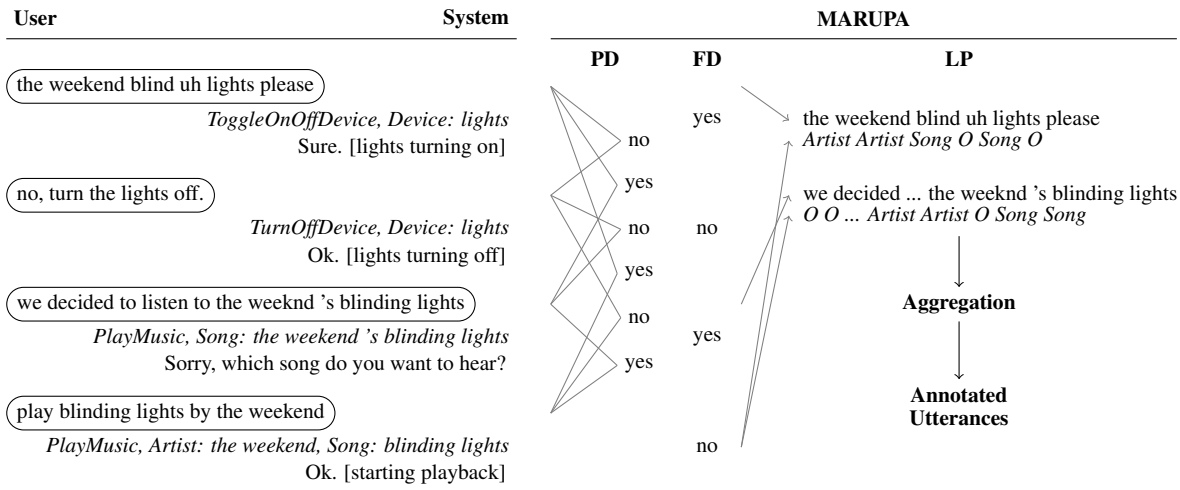

Figure 1: During the interaction (left), the first and third user utterances, exemplary for noisy and long-tail utterances, are mis-interpreted by the system. The final, more common paraphrase succeeds. MARUPA (right) uses PD and FD to detect two paraphrase pairs with a friction-causing and a successful utterance (1+4 and 3+4). For both, the successful interpretation is projected onto the failed utterance (LP). Utterance 2 is not used. After aggregation across interactions, the new data is used to re-train the model.

for the friction-causing utterances. Our proposed method, **MARUPA** (**M**ining **A**nnotations f**r**om **U**ser **Pa**raphrasing), combines *paraphrase detection (PD)*, *friction detection (FD)* and *label projection (LP)* models to automatically detect relevant paraphrasing behavior in interactions with a dialog system and to turn it into new, annotated utterances.

The proposed data collection approach has several advantages: First, it is fully automatic, thus reducing effort and cost for annotation and making data collection scalable. Second, the friction-causing utterances that are followed by paraphrases tend to be the less frequent ones from the long tail, such that the collected data helps in particular to increase coverage and improve accuracy on that part of the distribution. And third, using user feedback to annotate data integrates seamlessly with supervised learning and is agnostic to the type of model being used for IC and SL. It requires only user feedback and predictions of the underlying model. It can also be applied repeatedly with improving underlying models and is essentially an extension of self-learning, as we discuss in Section 2.

We present experiments with MARUPA on anonymized data from a commercial dialog system across many domains and three different languages. In addition, we also report component-wise evaluations and experiments on public datasets where possible. We observe improvements of IC and SL performance when using the collected data for training, demonstrating that the approach can effectively collect useful annotated utterances automatically from user paraphrasing behavior.

2 Related Work

Intent classification (IC) and slot labeling (SL) have been studied for several decades as fundamental building blocks of task-oriented dialog systems, dating back at least to the creation of the ATIS corpus (Price, 1990). In recent years, progress has been been made by applying deep learning, such as using recurrent networks (Mesnil et al., 2013), jointly learning both tasks (Zhang and Wang, 2016; Hakkani-Tür et al., 2016) and improving knowledge-sharing between them (Goo et al., 2018; E et al., 2019).

Methods to leverage user feedback for IC and SL have received less attention in the past, presumably because such work is difficult to perform without access to a deployed dialog system. Most similar to ours is the work of Muralidharan et al. (2019), which proposes to use positive and negative user feedback to automatically turn cheap-to-obtain coarse-grained annotations for SL into more fine-grained ones. Coarse and fine-grained annotations are used together in a multi-task learning setup. They apply that idea to playback requests for music, and rely on whether a user kept a song playing for more than 30

seconds as the feedback. The main differences to this work are that MARUPA is not music-specific but domain-agnostic, that it does not require a coarse-grained annotation to start with and that it integrates with existing IC/SL models more seamlessly, as it works in the same label space and requires no multi-task learning. We also report experiments across many domains and different languages.

Other related work is presented by Ponnusamy et al. (2020), who describe an online method to make use of paraphrasing behavior: They use absorbing Markov chains to distill common successful paraphrase pairs from paraphrasing behavior across many users and use them to re-write utterances at runtime; similar to query-rewriting in search engines. While focusing on the same user behavior, our methods are mostly orthogonal to that work, as it focuses on simplifying the utterance given a fixed IC/SL model while we aim to improve the IC/SL model. In less similar work, Qiu et al. (2019) propose a method to leverage paraphrase relationships between utterances to improve IC. Different to ours and the aforementioned methods, they try to find pairs of paraphrases among large, unstructured collections of unlabeled utterances rather than relying on the dialog flow and feedback given by users. More over, Yaghoub-Zadeh-Fard et al. (2019) study utterance paraphrases collected via crowdsourcing and develop methods to address common paraphrasing errors that occur in such crowdsourced data. Since our method relies on naturally occurring paraphrases, it does not face these challenges.

Beyond the IC and SL tasks of task-oriented dialog, incorporating user feedback is much more common for chit-chat dialog systems and for learning dialog management policies. For the latter, reinforcement learning based on simulated or real user feedback has been studied for a long time (Levin et al., 2000; Schatzmann et al., 2006; Liu et al., 2018). Similarly, chit-chat dialog systems are usually trained end-to-end with reinforcement learning and estimated user satisfaction has been part of proposed reward functions (Serban et al., 2018; Zhang et al., 2018). Hancock et al. (2019) propose a dialog agent that incorporates feedback via supervised multi-task learning and gathers the feedback by learning to explicitly ask a user for it. Through online learning, the feedback is incorporated while the system is actively used. Another line of work focuses on the detection of user feedback itself, rather than also trying to use it. Several models have been proposed to find positive and negative feedback utterances or predict a user's emotion (Wang et al., 2017; Hashimoto and Sassano, 2018; Eskenazi et al., 2019; Ghosal et al., 2019).

Our work is also related to the semi-supervised learning approach known as self-learning or pseudo-labeling, in which models are trained on predictions that a previous version of the model made on unlabeled data. This idea has been successfully applied to a wide range of language tasks, e.g. named entity recognition (Collins and Singer, 1999), word sense disambiguation (Mihalcea, 2004) and parsing (McClosky et al., 2006). Successful applications are also known from computer vision, of which Xie et al. (2019) gives an overview. For task-oriented dialog systems, Cho et al. (2019) report substantial error reductions using self-learning to bootstrap new features. While the use of MARUPA-labeled data for training is at its core self-learning, our approach goes further by leveraging user feedback in the form of paraphrasing and friction as additional signals to guide the self-learning process.

3 MARUPA

Before describing our approach, we start defining important terminology. An *utterance* u is a sequence of tokens, directed from a user to a dialog system. Annotated utterances are triples (u, i, s) with an *intent* label i and *slots* s. Each slot s_i has a *slot name* and a *slot value*, the latter being a subset of tokens from the utterance. As an example, consider $u =$ *"play blinding lights by the weekend"* with intent $i =$ *PlayMusic*. Slots can be written as a mapping $s = \{$*Artist* : *"the weekend"*, *Song* : *"blinding lights"*$\}$ or as token-level labels *"O B-Song I-Song O B-Artist I-Artist"* using BIO-encoding.

MARUPA relies on paraphrase detection (PD), friction detection (FD) and label projection (LP) components as illustrated in Figure 1. Given an interaction between a user and the dialog system, consisting of a sequence of user utterances u along with intent and slots $(\hat{\imath}, \hat{s})$ predicted by the dialog system, we find all pairs (u_j, u_k), where u_j occurs before u_k, that satisfy the following condition:

$$\begin{aligned} &time(u_k) - time(u_j) \leq \delta \;\; \wedge \;\; u_j \neq u_k \;\; \wedge \;\; PD(u_j, u_k) = 1 \;\; \wedge \\ &FD(u_j, \hat{\imath}_j, \hat{s}_j) = 1 \;\; \wedge \;\; FD(u_k, \hat{\imath}_k, \hat{s}_k) = 0 \;\; \wedge \;\; LP(u_j, u_k) \geq \theta \end{aligned} \tag{1}$$

It captures pairs that occur within a maximum time distance δ, that are paraphrases according to a classifier $PD(u_j, u_k) \in \{0, 1\}$ and have the desired friction classifications $FD(\cdot) \in \{0, 1\}$, i.e. u_j causing friction but not u_k. The threshold θ is imposed on the label projection score $LP(u_j, u_k) \in [0, 1]$ defined later in Equation 5 to ensure that unsuccessful projections are discarded. Each pair that satisfies Equation 1 yields a new training example for IC and SL, which are finally aggregated.

In the following, we describe each of the involved components in detail, starting with PD (Section 3.1), then FD (Section 3.2), LP (Section 3.3) and aggregation (Section 3.4).

3.1 Paraphrase Detection

Sentential paraphrases are commonly defined as different sentences with the same, or very similar, meaning (Dolan and Brockett, 2005; Ganitkevitch et al., 2013). Paraphrase detection is the corresponding binary classification task on sentence pairs. In this work, we apply it to pairs of dialog system utterances.

Compared to paraphrase detection in general, our context has one advantage: The notion of meaning, on which the definition of paraphrase relies, can be formalized precisely with regard to the downstream task, i.e. by relying on intents and slots. If two utterances are annotated with the same intent and the same set of slots, we consider them to be paraphrases. On the other hand, the context also brings several challenges. While paraphrases can be very different in its surface form, non-paraphrases can be extremely similar, such as *set volume to 3* and *set volume to 6*. In addition, speech recognition and endpointing errors make the utterances noisy and the intended meaning often hard to recover. Crucially, in-domain paraphrase detection data is needed to address those challenges, but to the best of our knowledge, no such dataset exists. To overcome this issue, we propose an automatic corpus creation method.

Corpus Creation To overcome the lack of in-domain paraphrase datasets, we propose a corpus creation approach that automatically derives paraphrase pairs from annotated utterances as used for IC and SL. Such data is necessarily available in our context and using it ensures that the paraphrase pairs resemble the domain of interest. The core idea is that two utterances with the same signature and same slot values but different carrier phrases must be paraphrases. We define the *signature* of an annotated utterance to be the set of its intent and slot names, e.g. {*PlayMusic, Artist, Song*}. The *carrier phrase* of an utterance is the utterance without slot values, e.g. *"play $Song by $Artist"*.

Algorithm 1 shows the corpus creation procedure. Given annotated utterances and a target size, it repeatedly samples signatures occurring in the utterances. To create a positive paraphrase example, we sample two different carrier phrases for that signature and inject the same sampled slot values into them. As an example, consider the signature {*PlayMusic, Artist, Song*}. We obtain a positive pair by sampling slot values {*Artist: "the weekend", Song: "blinding lights"*} and the following two carrier phrases:

$$\textit{play \$Song by \$Artist} \quad \rightarrow \quad \textit{play blinding lights by the weekend} \tag{2a}$$

$$\textit{\$Artist \$Song please} \quad \rightarrow \quad \textit{the weekend blinding lights please} \tag{2b}$$

The more interesting part is how negative pairs are sampled, which are crucial to make the task non-trivial for paraphrase models. We sample a second, very similar signature from the k nearest signatures in lines 10 and 11 using Jaccard distance between signatures σ_a and σ_b:

$$d_{sig}(\sigma_a, \sigma_b) = 1 - \frac{|\ \sigma_a \cap \sigma_b\ |}{|\ \sigma_a \cup \sigma_b\ |} \tag{3}$$

To continue with the previous example, consider the second signature {*PlayMusic, Song, Room*} with a Jaccard distance of 0.33 that leads, when sampling carrier phrases, to a negative utterance pair:

$$\{\textit{PlayMusic, Artist, Song}\} \rightarrow \quad \textit{play \$S by \$A} \quad \rightarrow \quad \textit{play blinding lights by the weekend} \tag{4a}$$

$$\{\textit{PlayMusic, Song, Room}\} \rightarrow \quad \textit{play \$S in the \$R} \quad \rightarrow \quad \textit{play blinding lights in the kitchen} \tag{4b}$$

Note that slots shared by both signatures will be filled with the same values, making the utterances of negative examples very similar. Another important detail is that line 3 samples according to the frequency distribution in $\mathcal{D}$, to ensure intents and slots have a similar distribution in $\mathcal{P}$, but all other sampling steps choose uniformly from the possible unique values (carrier phrases, slot values, nearest signatures) to ensure diversity. Slot values are sampled from all values seen for the slot name and intent.

Algorithm 1 Paraphrase Corpus Creation

Input: Set $\mathcal{D}$ of triples (u, i, s), size n, nearest k
Output: Set $\mathcal{P}$ of triples (u, u', l) with $l \in \{0, 1\}$

$\mathcal{P} = \emptyset$
for $n/2$ **do**
  Sample a signature σ from all in $\mathcal{D}$.
  Sample a carrier phrase c for σ.
  // Create positive example
  Sample another $c_{pos} \neq c$ for σ.
  Sample slot values for slot names in σ and inject into c, c_{pos} to obtain u, u_{pos}.
  Add $(u, u_{pos}, 1)$ to $\mathcal{P}$.
  // Create negative example
  Compute $d_{sig}(\sigma, \sigma')$ for all $\sigma' \neq \sigma$.
  Sample σ_{neg} from k nearest signatures.
  Sample a carrier phrase c_{neg} for σ_{neg}.
  Inject values from 7, plus samples for new slots in c_{neg}, into c, c_{neg} to obtain u, u_{neg}.
  Add $(u, u_{neg}, 0)$ to $\mathcal{P}$.
end for
Return $\mathcal{P}$.

Model For paraphrase detection, we rely on BERT (Devlin et al., 2019), a pre-trained neural model that reached state-of-the-art performance on the common paraphrase detection benchmarks QQP and MRPC. We combine paraphrase pairs into sequences as in the BERT paper, e.g. *"[CLS] play blinding lights [SEP] blinding lights please [SEP]"*. The model encodes the sequence and feeds the output through a 256-d ReLU layer followed by a binary classification. The whole model is fine-tuned on our paraphrase corpus to ensure it addresses the unique challenges of dialog. To support different languages, we use a multilingually pre-trained BERT model (Devlin et al., 2019) and fine-tune it in the target language.

3.2 Friction Detection

Friction occurs for a user of a dialog system if their utterance is not correctly interpreted and the system does not show the desired reaction. It is thus closely related to work on user satisfaction and user feedback detection discussed in Section 2. While friction can in general be caused by many parts of a dialog system – speech recognition, entity resolution, request fulfillment – we are particularly interested in friction caused by IC and SL for our data collection. Therefore, given a small amount of hand-labeled friction examples, we train a model that can automatically detect such friction cases.

Given an utterance u and predictions $\hat{\imath}$ and $\hat{s}$, we model friction detection as a binary classification task. We use a linear SVM for classification. Our set of binary features (see Table 1) captures the utterance itself, the intent and slots predicted for it, utterance-level confidence scores as well as status codes received from downstream components that try to act upon the provided interpretation. Since these features capture only the utterance and how the dialog system reacts to it, but no additional feedback a user might give afterwards, this modeling approach relies on the friction detector to become aware of current limitations of the underlying model and where they are, rather than relying on explicit feedback. In fact, we found that confidences and status codes from fulfillment components are strong predictors for that. Although other work found it beneficial to include the user's next utterance in similar tasks (Eskenazi et al., 2019), we observed no improvements in classification performance when doing so.

3.3 Label Projection

Once we found a paraphrase pair (u, u') with predictions $\hat{\imath}, \hat{s}$ and $\hat{\imath}', \hat{s}'$, of which the first caused friction but not the second, we want to use the successful interpretation $\hat{\imath}', \hat{s}'$ with the utterance u as a new training example. While this is trivial for intents, it is more intricate for slot labels as they are per token, which differ between u and its paraphrase u'. See Figure 2 for an example. Nevertheless, many tokens typically overlap, which motivates our token alignment-based approach inspired by Fürstenau and Lapata (2009).

For each source-target token pair (u_j, u'_k), we first compute a similarity $sim(u_j, u'_k) \in [0, 1]$. Based on these pairwise similarities, we then consider all alignments of the n source tokens in u to the m targets

Group	Features
N-Grams	uni- and bi-grams of u
Predictions	intent $\hat{\imath}$, slot names in $\hat{s}$
Confidences	IC, SL, speech recognition
Fulfillment	status code

Table 1: Friction detection feature groups.

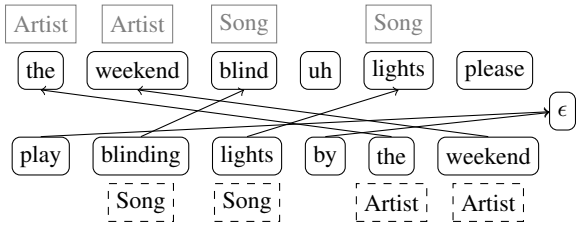

Figure 2: Alignment-based slot label projection.

in u' and score them by the average similarity of the chosen alignments

$$LP(u, u') = \frac{1}{n} \sum_{j=1}^{n} \sum_{k=1}^{m+1} sim(u_j, u'_k)\ x_{jk} \tag{5}$$

where x_{jk} is a binary decision variable representing the alignment. We enforce that all source tokens must have an alignment, only one source can align to one target, but allow alignment to a virtual empty target as a fallback. We experimented with different similarities and optimization methods. For MARUPA, we settled to using as similarity the inverse of character-level Levenshtein edit distance normalized by length, which works well to identify identical tokens or slight morphological variations. The optimization is done greedily, choosing the most similar target for each source from left to right. As we show in experiments in Section 5, this choice offers a good trade-off of alignment accuracy and runtime. Finally, given the alignment, we project slot labels $\hat{s}'$ to the target tokens accordingly (see Figure 2).

3.4 Aggregation

Once PD and FD identified relevant utterance pairs and LP projected slot labels, we obtain new annotated training examples. However, due to noise in the source data and noise introduced by the components of our method, they can be of varying quality. We therefore introduce a final aggregation step that takes examples collected from many user interactions and filters them by consistency. Only if we create an annotation for a specific utterance repeatedly and consistently, we keep the examples. In addition, we also test the effect of adding the training examples to the IC/SL model in terms of accuracy changes on a development set and filter out examples for intent/slot-combinations with accuracy degradations.

Note also that the conditions in Equation 1 are conjunctive, allowing us to enforce them in an arbitrary order on potential utterance pairs. In practice, this enables enforcing the least resource intense checks first, which will reduce the amount of pairs that have to be processed by more time-consuming methods such as paraphrase classification with the fine-tuned BERT model.

4 Experimental Results

In this section, we present experiments with MARUPA in the context of a commercial dialog system across multiple languages. They demonstrate that the newly collected training examples can help to improve the IC and SL performance, in particular on long-tail utterances.

4.1 Setup

We experiment with German, Italian and Hindi utterances. In each case, we use the production IC/SL model of the dialog system as the baseline and compare it against a model re-trained after adding our additional training examples. We report relative changes of a combined IC/SL error rate. In each language, the train and test data contains several hundred different intents and slots.

To collect new training examples, we apply MARUPA to a sample of anonymized user interactions with the dialog system. We set $\delta = 30s$ and $\theta = 0.4$ in Equation 1. For paraphrase detection, we use Algorithm 1 to derive a corpus of 300k paraphrase pairs with $k = 30$ from the training data of the baseline model, and split it into a train, validation and test split (70/10/20). As discussed in Section 3.1,

Language	Used Since	Added Data	Test Set (by utterance frequency)							Held-Out Data
			Total	[0,1)	[1,2)	[2,4)	[4,7)	[7,10)	$\geq$10	
German	3.0 y	70k	-0.01	+0.47	-2.91	-1.34	0.00	0.00	0.00	-79.51
Italian	1.0 y	71k	+0.08	+0.11	+0.42	-0.05	-0.96	0.00	0.00	-91.27
Hindi	0.5 y	2k	+0.53	+0.40	+1.39	+0.74	+0.85	0.00	0.00	-72.55

Table 2: Relative changes in error rates when adding MARUPA-collected data, evaluated on the main IC/SL test set and held-out collected instances. Frequency bin thresholds in $\log_2$ scale.

we fine-tune *BERT-Base Multilingual Cased*[1]. We use a batch size of 64 and dropout of 0.2. The model is trained using Adam with a learning rate of $5 \cdot 10^{-5}$ and with early stopping. For friction detection, we use the model described in Section 3.2 and tune the SVM's regularization constant via grid search. Labels are projected as described in Section 3.3. Finally, we aggregate the collected examples by keeping only utterances that occur at least 10 times and for which the same annotation was created in at least 80% of the cases. In addition, we filter out collected examples belonging to intent/slot-combinations for which the error rate on the IC/SL development set increases after retraining the model.

4.2 Results

Table 2 shows the results of our experiment for German, Italian and Hindi. We evaluate the model's error rate on its main test set, obtained via manual annotation, and on a set of held-out MARUPA-collected examples (20%). For the former, we break down results by utterance frequency.[2] While the overall change on the main test set (*Total*) is negligible, the break down reveals that for low-frequency utterances the new data leads to error rate reductions of up to 3%. As desired, this comes with no negative effects for high-frequency utterances. Even stronger error reductions can be seen for the held-out MARUPA examples, on which the baseline's error rate is by design high. We observed that many collected utterances are so rare that they are not captured even by the lowest-frequency bin of the main test set, making it difficult to assess the full impact of MARUPA. Across languages, we observe that for newer languages like Hindi, less useful examples can be collected and their quality is lower, which is because fewer user interactions are available and the underlying IC/SL is less mature. Overall, the results demonstrate that MARUPA can improve accuracy on long-tail utterances.

5 Analysis

In addition to the previous section, which presented our main experimental results, the following sections show further experiments to provide additional insights into our proposed method.

5.1 Application to Public Datasets

Since the core of our approach is to leverage user paraphrasing behavior, it is difficult to conduct experiments on academic datasets for IC and SL, which to the best of our knowledge contain neither examples for friction nor paraphrasing. However, we try to replicate our application scenario as closely as possible.

Setup We use SNIPS (Coucke et al., 2018), a common benchmark with 14,484 English utterances annotated for IC and SL. We sample a subset of 10% of the training set as labeled data and treat the remaining part as unlabeled. On the unlabeled part, we create paraphrase pairs re-using ideas from Section 3.1: We group all utterances by signature and compute the frequency of unique carrier phrases. Per signature, we use all utterances from the 70%-head of the carrier phrase distribution, and for each, create a paraphrase by sampling another carrier phrase from the remaining 30%-tail and inject the same slot values. This process turns the unlabeled data into pairs of paraphrases, each pair consisting of a

[1] https://github.com/google-research/bert/blob/master/multilingual.md

[2] Bins are chosen to contain roughly equal amounts of examples, except for [0,1) with unique utterances (40% of examples).

Setting		Training Data				Evaluation Metrics		
		Labeled	Unl.-Head	Unl.-Tail	Count	IC	SL	IC+SL
Baseline	1	All	—	—	1,309	97.54	76.11	56.27
No Selection	2.1	All	All	—	9,253	-0.13	+1.47	+1.90
	2.2	All	All	All, Self	17,197	+0.17	+1.37	+1.86
	2.3	All	All	All, LP	17,197	+0.51	+2.09	+3.21
Self-Labeled vs. Projection on Tail	3.1	All	—	FD, Self	2,807	-0.16	-0.77	-1.23
	3.2	All	—	FD, LP	2,807	**+0.99**	+2.63	+3.07
Confidence vs. Friction Detection	4.1	All	Conf	—	4,293	+0.23	+2.72	+4.74
	4.2	All	FD	—	6,208	+0.01	+3.42	+5.41
	4.3	All	Conf	Conf, LP	7,277	+0.74	+3.47	+5.01
MARUPA (FD+LP)	5	All	FD	FD, LP	7,706	+0.87	**+4.42**	**+6.57**

Table 3: Experiments with unlabeled head/tail–paraphrase pairs created for SNIPS (English), comparing label projection to self-labeling and selection by friction to selection by confidence. Metrics are accuracy (IC), slot-based F1 (SL) and full frame accuracy (IC+SL). All results are averages of 10 re-runs.

less common (tail) and a more common (head) utterance, similar to our real-world friction scenario. For SNIPS, the process yields 1,309 labeled utterances and 7,944 unlabeled paraphrase pairs.

Note that we can only apply FD and LP of MARUPA, but not PD, as paraphrase pairs are already given in this setup. For IC and SL, we train a joint neural model that encodes utterances with ELMO embeddings (Peters et al., 2018) and then feeds them through a BiLSTM. Slot labels are predicted from each hidden state and intent labels from the mean of them. We use 1024-d ELMO embeddings, a single-layer 200-d BiLSTM and dropout of 0.2 after both. The model is trained with Adam, batch size 16 and early stopping. For evaluation (and stopping) we use the original test (and validation) set of SNIPS.

Results Table 3 shows results for various ways to leverage the unlabeled paraphrase pairs, including using FD and LP of MARUPA (see row 5). The baseline (1) uses only labeled data, and all other methods make use of this model's predictions on the unlabeled pairs. The most simple approach of ingesting all self-labeled data back into training, for just head utterances (2.1) or head and tail (2.2), already boosts performance on both tasks. However, when using label projection from self-labeled head utterances to tail utterances (2.3), the performance is even better, demonstrating the effectiveness of label projection.

To apply FD in this setup, we train the IC/SL model with cross-validation on the labeled data and then train the detector on the out-of-fold predictions.[3] If we apply label projection only to paraphrase pairs selected by FD (3.2), i.e. pairs with friction for the tail but not for the head, we see that projection outperforms self-labeling (3.1) on these cases even more, as the tail's self-label is expected to cause friction. Finally, we compare selecting paraphrase pairs by FD to selecting based on the baseline model's confidence, a common approach for self-supervised learning (see Section 2). We tune a confidence threshold on the validation set.[4] Selecting with FD is superior both when using only head utterances (4.1 vs. 4.2) and when including tail utterances with label projection (4.3 vs. 5). Thus, we conclude that our self-learning approach based on friction detection is effective also beyond our exact application scenario.

5.2 Quality of Derived Paraphrase Corpora

Next, we give more insight into the effectiveness of the proposed paraphrase corpus creation. We include public datasets, as paraphrase detection was not included in the previous experiment.

[3]We omit the fulfillment and speech recognition features described in Table 6, as they are not available in this setup.

[4]We tried thresholds 0.8, 0.85, 0.9, 0.95, 0.97, 0.99, 0.999 based on plotting the distribution, finding 0.95 to perform best.

Method	German Internal	Italian Internal	Hindi Internal	English SNIPS	English ATIS
Edit-Distance	59.75	65.29	64.65	65.95	61.00
Jaccard Similarity	61.02	66.04	67.82	68.80	63.10
N-Gram SVM	80.36	83.22	83.79	86.85	93.40
Fine-tuned mBERT	**92.33**	**92.99**	**94.61**	**97.75**	**97.55**

Table 4: Paraphrase detection accuracy of fine-tuned mBERT and baselines across paraphrase corpora.

Language	Maturity	Prec.	Rec.	F1
English	5.0	76.9	55.2	64.3
German	3.0	76.3	60.0	67.2
French	1.5	82.6	69.7	75.6
Italian	1.0	85.5	72.5	78.4
Hindi	0.5	**92.0**	**81.6**	**86.5**

Table 5: Friction detection performance across languages with IC/SL model maturity in years.

Features	F1	Delta
All	67.2	
- N-Grams	65.2	-2.0
- Predictions	65.3	-1.9
- Confidences	62.4	-4.8
- Fulfillment	56.2	-11.0

Table 6: Feature ablation for friction detection on German utterances.

Setup We use our internal IC/SL training data for German, Italian and Hindi as well as the commonly used SNIPS and ATIS (Price, 1990) corpora (both English) to derive paraphrase pairs and fine-tune mBERT models as outlined in Section 4.1 on them. For each, we report accuracy for paraphrase detection evaluated on the 20% test split of the paraphrase pairs. For ATIS and SNIPS, due to their small size, we derive corpora with 10k pairs and $k = 10$. As baselines, we include simple threshold-based classifiers using character-level edit-distance or token-level Jaccard similarity and a linear SVM using binary uni- and bigram features, and- and xor-combined between the two utterances.

Results Table 4 shows paraphrase detection results. We observe that the simple baselines relying on surface-level similarities perform much worse than BERT, which demonstrates that our approach of selecting negative examples with high similarity (see Section 3.1) is effective for creating a challenging corpus. We saw in preliminary experiments that sampling negative pairs fully randomly makes the baselines perform much better. Furthermore, we observe that the trained SVM performs much better than the simple baselines, but using a fine-tuned BERT model substantially improves upon it. Finally, this experiment also shows that our approach can be applied across different languages and domains.

5.3 Analysis of Friction Detection

For friction detection, we report classification performance for the task in Table 5, evaluated on a held-out set of the same data the models are trained on for MARUPA. Performance, and in particular precision, is generally high across languages, showing that the model can reliable find many friction cases. That also confirms that the approach is largely language independent and works for any of the tested ones. An interesting trend is that friction detection, using the set of features proposed in this work, tends to become more difficult the more mature the underlying IC/SL model for the language is, indicating that the causes of friction most easy to detect are also being resolved the earliest.

In Table 6, we show ablation results for the friction detector trained on German utterances. Removing any of the features decreases F1-scores by at least 2 points, showing that they all contribute to the task. Among them, the status codes received from the fulfillment component are by far the most useful features. This underlines the point made earlier that the dialog system is already aware of many friction cases and does not necessarily require explicit or implicit feedback from the user to detect friction – but, we do require feedback in terms of paraphrasing to go further and avoid the friction in the future.

Optimization	Similarity	Exact Match	Token F1	Runtime
Greedy	Word Identity	82.65	92.89	**3.8s**
	Norm. Character Edit-Distance	85.46	94.17	8.8s
	Word2Vec Cosine Similarity	81.47	92.64	51.8s
Exact	Word Identity	84.87	94.26	61.8s
	Norm. Character Edit-Distance	**87.03**	**95.52**	75.1s
	Word2Vec Cosine Similarity	86.41	95.23	120.5s

Table 7: Label projection accuracy and runtime across similarity measures and optimization strategies.

5.4 Quality vs. Runtime Trade-Offs in Label Projection

Finally, Table 7 shows different instantiations of our label projection approach, tested on a dataset of 11k German utterance pairs for which reference projections have been derived from gold IC/SL annotations. We report whether the projection matches the reference completely (Exact Match) and F1-scores over token-level matches (Token F1). We observed that using word embeddings to compute similarities provided no improvements, as cases where that would be needed, i.e. aligning synonyms, are rare in the data. Exact optimization using linear programming, on the other hand, does improve the projection regardless of similarity being used, but comes at a large increase in runtime. Trading off quality and runtime, we therefore rely on greedy optimization with character-level edit-distance in MARUPA.

6 Conclusion

We proposed MARUPA, an approach to collect annotations for friction-causing long-tail utterances automatically from user feedback by using paraphrase detection, friction detection and label projection. We demonstrated that this form of user feedback-driven self-learning can effectively improve intent classification and slot labeling in dialog systems across several languages and on SNIPS. In the future, we plan to integrate more advanced aggregation methods to reduce noise and to more closely study the effect of the feedback loops that emerge when our data collection and re-training are applied repeatedly.

Acknowledgements

We would like to thank Judith Gaspers, Fabian Triefenbach and our anonymous reviewers for their thoughtful comments and suggestions that improved this paper.

References

Eunah Cho, He Xie, and William M. Campbell. 2019. Paraphrase Generation for Semi-Supervised Learning in NLU. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating NLG*, pages 45–54, Minneapolis, MN, USA.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, College Park, MD, USA.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips Voice Platform: An Embedded Spoken Language Understanding System for Private-By-Design Voice Interfaces. *arXiv*, 1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, MN, USA.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 9–16, Jeju Island, Korea.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A Novel Bi-directional Interrelated Model for Joint Intent Detection and Slot Filling. In *Proceedings of the 57th Annual Meeting of the ACL*, pages 5467–5471, Florence, Italy.

Maxine Eskenazi, Shikib Mehri, Evgeniia Razumovskaia, and Tiancheng Zhao. 2019. Beyond turing: Intelligent agents centered on the user. *arXiv*, 1901.06613.

Hagen Fürstenau and Mirella Lapata. 2009. Semi-Supervised Semantic Role Labeling. In *Proceedings of the 12th Conference of the EACL*, pages 220–228, Athens, Greece.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, GA, USA.

Deepanway Ghosal, Navonil Majumder, Soujanya Poria, Niyati Chhaya, and Alexander Gelbukh. 2019. DialogueGCN: A graph convolutional neural network for emotion recognition in conversation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 154–164, Hong Kong, China.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-Gated Modeling for Joint Slot Filling and Intent Prediction. In *Proceedings of the 2018 Conference of the NAACL-HLT*, pages 753–757, New Orleans, LA, USA.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM. In *Interspeech*, pages 715–719, San Francisco, CA, USA.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy.

Chikara Hashimoto and Manabu Sassano. 2018. Detecting absurd conversations from intelligent assistant logs by exploiting user feedback utterances. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 147–156, Lyon, France.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2060–2069, New Orleans, LA, USA.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 152–159, New York, NY, USA.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. In *Interspeech*, pages 3771–3775, Lyon, France.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 33–40, Boston, MA, USA.

Deepak Muralidharan, Justine Kao, Xiao Yang, Lin Li, Lavanya Viswanathan, Mubarak Seyed Ibrahim, Kevin Luikens, Stephen Pulman, Ashish Garg, Atish Kothari, and Jason Williams. 2019. Leveraging User Engagement Signals For Entity Labeling in a Virtual Assistant. *arXiv*, 1909.09143.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, LA, USA.

Pragaash Ponnusamy, Alireza Roshan-Ghias, Chenlei Guo, and Ruhi Sarikaya. 2020. Feedback-based self-learning in large-scale conversational ai agents. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, New York, NY, USA.

P. J. Price. 1990. Evaluation of Spoken Language Systems: the ATIS Domain. In *Speech and Natural Language: Proceedings of a Workshop*, Hidden Valley, PA, USA.

Zimeng Qiu, Eunah Cho, Xiaochun Ma, and William Campbell. 2019. Graph-Based Semi-Supervised Learning for Natural Language Understanding. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for NLP*, pages 151–158, Hong Kong, China.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21(2):97–126.

Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeswar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. 2018. A Deep Reinforcement Learning Chatbot (Short Version). *arXiv*, 1801.06700.

Gokhan Tur and Renato De Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. John Wiley and Sons.

Xin Wang, Jianan Wang, Yuanchao Liu, Xiaolong Wang, Zhuoran Wang, and Baoxun Wang. 2017. Predicting users' negative feedbacks in multi-turn human-computer dialogues. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 713–722, Taipei, Taiwan.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. 2019. Self-training with noisy student improves imagenet classification. *arXiv*, 1911.04252.

Mohammad-Ali Yaghoub-Zadeh-Fard, Boualem Benatallah, Moshe Chai Barukh, and Shayan Zamanirad. 2019. A study of incorrect paraphrases in crowdsourced user utterances. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 295–306, Minneapolis, MN, USA.

Xiaodong Zhang and Houfeng Wang. 2016. A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding. In *Proceedings of the Twenty-Fifth IJCAI*, page 2993–2999, New York, NY, USA.

Wei-Nan Zhang, Lingzhi Li, Dongyan Cao, and Ting Liu. 2018. Exploring implicit feedback for open domain conversation generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA.