Data-Efficient Paraphrase Generation to Bootstrap Intent Classification and Slot Labeling for New Features in Task-Oriented Dialog Systems

Shailza Jolly[1,2,*], **Tobias Falke**[3], **Caglar Tirkaz**[3] and **Daniil Sorokin**[3]
[1]Technische Universität Kaiserslautern, Germany
[2]German Research Center for Artificial Intelligence (DFKI GmbH)
[3]Amazon Alexa AI, Aachen, Germany
shailza.jolly@dfki.de, {falket,caglart,dsorokin}@amazon.com

Abstract

Recent progress through advanced neural models pushed the performance of task-oriented dialog systems to almost perfect accuracy on existing benchmark datasets for intent classification and slot labeling. However, in evolving real-world dialog systems, where new functionality is regularly added, a major additional challenge is the lack of annotated training data for such new functionality, as the necessary data collection efforts are laborious and time-consuming. A potential solution to reduce the effort is to augment initial seed data by paraphrasing existing utterances automatically. In this paper, we propose a new, data-efficient approach following this idea. Using an interpretation-to-text model for paraphrase generation, we are able to rely on existing dialog system training data, and, in combination with shuffling-based sampling techniques, we can obtain diverse and novel paraphrases from small amounts of seed data. In experiments on a public dataset and with a real-world dialog system, we observe improvements for both intent classification and slot labeling, demonstrating the usefulness of our approach.

1 Introduction

Intent classification and slot labeling are two fundamental components in task-oriented dialog systems, producing a formal meaning representation for an utterance that the system can act upon to fulfill the user's request. As shown in Figure 1, it is typically modeled by classifying the utterance into a set of supported intents and labeling its sequence of tokens with expressed slots. We refer to the combined output of these two steps as the *interpretation* of the utterance. While the performance on these tasks has by now reached high accuracies on benchmarks like SNIPS (Coucke et al., 2018), the widespread use of real-world systems like Apple's Siri, Amazon's Alexa and Google's Assistant leads to new research challenges. As such systems are constantly expanding their functionality, bootstrapping new features is a common task that we focus on in this work.

In this paper, we define a *new feature* as a set of one or more intents and related slots that were not known to the system before. The main challenge when introducing a new feature is that typically, only very little seed training data is available, which makes it difficult to train good intent and slot models for the feature. However, manually collecting annotated data is expensive and time-consuming, slowing down the expansion of the system's functionality. We therefore aim to reduce that time and effort by automatically augmenting the seed training data and follow the idea of recent work to leverage paraphrase generation (Malandrakis et al., 2019; Cho et al., 2019). Through paraphrasing, we can automatically increase the amount of training data, but for the data to be useful, we have to ensure that the new utterances are diverse and different from what we already have. If not, the data augmentation would degenerate to simply upsampling the seed data. That can on its own already be helpful, but it does not expose any new input examples to the model during training which can help the model learn and generalize even better. On the other hand, we also have to ensure that paraphrases remain realistic and natural utterances that are representative of the real-world data distribution.

[*]The first author worked on this paper during an internship at Amazon.

This work is licensed under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/.

Proceedings of the 28th International Conference on Computational Linguistics: Industry Track, pages 10–20
Barcelona, Spain (Online), December 12, 2020

U:	find	movies	playing	at	the	closest	movie	theatre
I:	SearchScreeningEvent							
S:	O	B-Movie-Type	O	O	O	B-Spatial-Rel	B-Loc-Type	I-Loc-Type

Figure 1: Example utterance with intent and slot labels from SNIPS.

In contrast to previous work, we propose a data-efficient augmentation approach that can work with very small amounts of seed examples. First, we propose using an *interpretation-to-text* model that can leverage similarities to existing intents and shared slots when generating paraphrases for new features. In addition, instead of requiring pairs of paraphrases, it can be trained on single utterances annotated with intents and slots as they already exist for the downstream tasks. Second, we further propose new *paraphrase sampling strategies* that increase the amount and diversity of obtained paraphrases by using random sampling and shuffling the input representation. And third, we obtain token-level slot labels for the paraphrases via *alignment-based label projection*, instead of relying on self-labeling with a baseline model as in prior work. These differences make our approach very data-efficient, such that it can produce diverse and novel training examples for a new feature from just 100 seed utterances.

We evaluate our technique by simulating the introduction of new features on SNIPS, a common benchmark for intent classification and slot labeling in English, and on German data from a real-world dialog system. In both settings, we see substantial improvements for new features without negative effects on existing features, demonstrating the usefulness of our approach despite little seed data being available.

2 Related Work

Intent classification and slot labeling have been studied for several decades as fundamental building blocks of task-oriented dialog systems, dating back at least to the introduction of the ATIS corpus (Price, 1990) and subsequent work, e.g. (Pieraccini et al., 1992). In recent years, much progress has been made by applying deep learning techniques, such as recurrent networks (Mesnil et al., 2013), and modeling both tasks jointly with a single model (Zhang and Wang, 2016; Hakkani-Tür et al., 2016). Approaches that further improve knowledge sharing between the tasks, such as slot-gates (Goo et al., 2018) or bi-directional task connections (E et al., 2019), continued this line of work. Yet, sufficiently large training datasets are required for such advanced machine learning approaches to be effective.

To overcome the lack of training data for new features and avoid costly data collections, several proposals have been made in the recent past. Machine translation can be used to obtain training examples if the feature already exists for other languages (Gaspers et al., 2018). Cross-lingual transfer learning is another technique to effectively use such existing data (Do and Gaspers, 2019). If a feature is already being actively used, feedback signals from users, such as paraphrases or interruptions, can be identified in user interactions to obtain additional training data (Muralidharan et al., 2019). If unlabeled utterances exist for the feature, pairs of paraphrases between labeled and unlabeled data might be found, which allows deducing labels for the unlabeled utterance from the paraphrase relationship (Qiu et al., 2019).

In our work, we make no assumption about the availability of labeled data in other languages, user feedback or unlabeled data, as none of them is necessarily available when bootstrapping a new feature. Therefore, the closest existing work to ours is (Cho et al., 2019) and (Malandrakis et al., 2019), who both deal with the setup where only seed examples are available. Malandrakis et al. (2019) propose using conditional variational auto-encoders to generate paraphrases for the seed data and show that the paraphrases increase intent classification performance in their experiment. In contrast to our work, they do not evaluate on slot labeling and do not suggest a technique to add slot labels to the paraphrases.

Cho et al. (2019) generate paraphrases for seed examples with a transformer network and self-label them with a baseline intent and slot model. When added as training data, this combination of paraphrasing with self-supervised learning improves both tasks in their experiment. Our approach differs mainly in terms of data efficiency, as we do not rely on a baseline model for self-labeling – which in turn re-

quires sufficient seed data for training – and in addition, we leverage data from existing features and use more advanced sampling strategies to obtain sufficient amounts of high-quality paraphrases from little seed data. We therefore demonstrate that our approach is applicable to earlier stages of bootstrapping by running our experiments on orders of magnitude less seed data than Cho et al. (2019). Furthermore, our paraphrase generation approach differs by training an interpretation-to-text model instead of a text-to-text model, such that no corpus of paraphrase pairs is needed. We rely exclusively on the already existing data for the downstream tasks of intent classification and slot labeling.

Beyond our specific use case, the task of paraphrase generation in general has gained much attention in recent years, driven partly by large-scale datasets such as Quora Questions Pairs[1] and the WikiAnswers[2] corpus, which enabled training complex models. Prakash et al. (2016) proposed one of the first neural models, relying on residual LSTMs. Further work explored auto-encoders (Gupta et al., 2018), generator-discriminator models trained with reinforcement learning (Li et al., 2018) or decompositions into multiple recurrent and transformer models that paraphrase at different levels (Li et al., 2019). All approaches in this line of work follow the text-to-text approach and assume training data in the form of paraphrase pairs. Because existing large datasets of this type focus on questions, we found direct application of models trained on them to the dialog domain to yield very unnatural paraphrases.

More closely related to the interpretation-to-text approach we use in this paper is work on generating natural language from structured data. Various versions of this task, differing mainly by the type of input, have been studied, e.g. generation from abstract meaning representations (Konstas et al., 2017) or from tabular data (Chen et al., 2019), as well as generation in unsupervised settings with denoising auto-encoders (Freitag and Roy, 2018). The E2E NLG challenge (Dušek et al., 2020), a recent competition carried out on a dataset from the restaurant domain with slot-based input representations, resembles our paraphrase generation scenario very closely. Among 62 competition entries, the organizers found that sequence-to-sequence models, as also used in our work, were most popular and also very competitive. However, a difference of that line of work is that paraphrases are evaluated only intrinsically, which does not necessarily reflect their usefulness for intent classification and slot labeling.

3 Data Augmentation Approach

We formalize the introduction of a new feature as having two sets of training data $\mathcal{D}_N$ and $\mathcal{D}_O$, where the former contains the seed examples for the new feature and the latter has all available examples for existing features. An example $(u, i, s) \in \mathcal{D}_O \cup \mathcal{D}_N$ consists of a tokenized utterance u, the true intent label i and true slot labels s in BIO-format. We refer to the label of a slot as the *slot name* and the covered tokens as its *slot value*. Let $\hat{s}$ be the mapping of slot names to their values, e.g. $\hat{s} = \{$*Movie-Type* : *movies*, *Spatial-Rel* : *closest*, *Loc-Type* : *movie theatre*$\}$ for the example in Figure 1.

In our scenario, $\mathcal{D}_N$ contains one or a few new intents not yet in $\mathcal{D}_O$[3] and is very small. We propose a paraphrase generation model $\mathcal{PG}$ that produces new examples $\mathcal{D}_P = \mathcal{PG}(\mathcal{D}_N)$ with the goal of improving intent classification and slot labeling performance when those models are trained on $\mathcal{D}_O \cup \mathcal{D}_N \cup \mathcal{D}_P$ instead of just $\mathcal{D}_O \cup \mathcal{D}_N$. In this section, we will describe how $\mathcal{PG}$ learns to paraphrase utterances, how we sample diverse paraphrases from it and how we project labels onto them to obtain examples $\mathcal{D}_P$.

3.1 Interpretation-to-Text Paraphrase Generation

In our interpretation-to-text approach, rather than learning to map utterances to paraphrases, we instead use examples (u, i, s) to model utterance sequences token by token conditioned on their interpretation:

$$p(u \mid i, \hat{s}) = \prod_{j=1}^{n} p(u_j \mid u_{1:j-1}, enc_j(i, \hat{s})) \tag{1}$$

where $enc_j(i, \hat{s})$ is an attention-based encoding of the interpretation $(i, \hat{s})$ at decoding step j. The idea behind this approach is that the notion of two utterances being paraphrases is, for our purposes, equivalent

[1] https://www.kaggle.com/c/quora-question-pairs

[2] http://knowitall.cs.washington.edu/paralex/

[3] Note that while all intents in $\mathcal{D}_N$ are new, slots might overlap with $\mathcal{D}_O$ since generic slots are used across many intents.

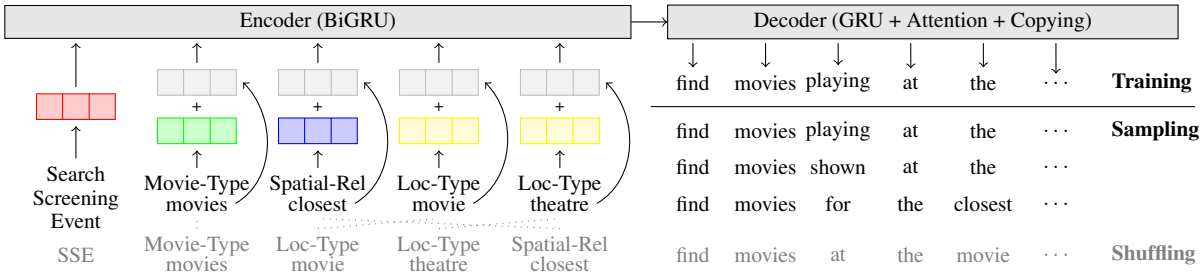

Figure 2: The interpretation-to-text model encodes an interpretation as a sequence of intent, slot name and slot value embeddings and is trained to predict the utterance. At inference time, we sample token by token and shuffle the input slot order (bottom) to obtain diverse paraphrases.

with them having the same interpretation. Thus, we model a mapping from that unique representation of a set of paraphrases – the shared interpretation – to all its realizations. At inference time, the conditional language model $p(u \mid i, \hat{s})$ can then, once conditioned on a specific interpretation, provide a distribution over all possible realizations from which paraphrases can be sampled.

In terms of data-efficiency, an obvious advantage is that downstream training data $\mathcal{D}$ can be used to directly train the model. We can also include the large set $\mathcal{D}_O$ in addition to the seed data $\mathcal{D}_N$ to train a much more powerful paraphrase model than with $\mathcal{D}_N$ alone or with out-of-domain paraphrase data. Thereby, the model can learn more general properties of the language used with dialog systems and similarities of utterances across intents and slots. At inference time, when sampling paraphrases for interpretations from $\mathcal{D}_N$ token by token, the model can interpolate between utterances seen in $\mathcal{D}_N$ or $\mathcal{D}_O$ and thus is able to create novel utterances not seen during training but that reflect the interpretation.

We implement $p(u \mid i, \hat{s})$ following the sequence-to-sequence paradigm using a bi-directional GRU for encoding and a GRU equipped with attention and a pointer mechanism for decoding (See et al., 2017). Embeddings for utterance tokens, slot names and intents are learned. An interpretation is fed into the model as a sequence starting with the embedded intent, followed by vectors that are the sum of a slot name and slot value token embedding. See Figure 2 for an illustration. The same vocabulary and embeddings are used in encoder and decoder, but only tokens can be generated on the output side. We train minimizing cross-entropy for the utterances in $\mathcal{D}_O \cup \mathcal{D}_N$.

3.2 Paraphrase Sampling Strategies

At inference time, sequence-to-sequence models typically use greedy decoding or beam search to find the (approximately) best sequence given an input. In our use case of obtaining additional training data, however, we are not only interested in the most likely realization of an interpretation, but rather want multiple diverse and novel utterances. We therefore rely on *input shuffling* and *random sampling*.

3.2.1 Input Representation Shuffling

The fact that feeding a sequence to the model requires us to order the slots in a certain way provides a simple way for obtaining multiple paraphrases: shuffling the order. We observed that although we train the model using one order (corresponding to the utterance), decoding with alternative ones provides paraphrases that also mention the values in that alternative order. A problem is that decoded utterances sometimes miss a slot, which motivates our quality metric *partial slot carry-over (PSCO)*, measuring the fraction of slots $\hat{s}$ for which at least one token of the slot value $\hat{s}_j^v$ occurs in the decoded utterance u'.

$$PSCO(u', \hat{s}) = \frac{1}{|\hat{s}|} \sum_{j=1}^{|\hat{s}|} \min(1, |\{t \mid t \in u' \wedge t \in \hat{s}_j^v\}|) \tag{2}$$

Using that metric, we collect k paraphrases given a seed utterance u as follows: First, we create input sequences corresponding to all permutations of the slot values. Next, we decode the best utterance for each input with beam search, compute its PSCO and keep only candidates with a rate of 1. From the

remaining candidates, we sample k (if they are more) or upsample to k (if they are less). The last step is to ensure that each seed leads to exactly k paraphrases and the initial distribution is thereby preserved.

3.2.2 Random Sampling

As a second strategy, we replace beam search with random sampling, which samples an utterance token by token according to the probability distribution over the vocabulary rather than trying to find the most likely sequence. Since this decoding process is not deterministic, multiple rounds of sampling yield different utterances, which allows us to obtain multiple paraphrases per seed. When decoding token u_j, we scale the logits z_t over the vocabulary with a temperature α before applying softmax

$$p(u_j = t \mid u_{1:j-1}, h_j) = \frac{\exp(z_t/\alpha)}{\sum_{t' \in V} \exp(z_{t'}/\alpha)} \tag{3}$$

and then sample from the top-β tokens according to their probabilities. $\alpha < 1$ makes the distribution spikier, such that the most likely tokens are sampled more often, while $\alpha > 1$ promotes less likely ones, as does a larger β which defines the size of the distribution's head to consider. With higher values, a larger part of the distribution will be explored, leading to higher novelty and diversity of the utterances, but potentially also to more unnatural sequences.

3.3 Label Projection

Given a paraphrase u' for a seed example (u, i, s), we finally need to project labels i and s to turn u' into a useful example for the downstream tasks. While this is trivial for intent labels – we can simply use i – it is more intricate for per-token slot labels as u and u' differ. Nevertheless, many tokens typically overlap, which motivates using a 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 $sim(u_j, u'_k) \in [0, 1]$. We use the inverse of character-level Levenshtein edit distance normalized by length, which works well to identify identical tokens or slight morphological variations. Experiments with word embeddings to capture more semantic similarities made only small differences on our data. Based on the pairwise similarities, we then consider all alignments of source tokens in u to targets in u' and score them by their average similarity of the chosen alignments, allowing that a source is aligned to a virtual empty target token if no better target is found. Rather than exactly solving this optimization problem as done in (Fürstenau and Lapata, 2009), we find the best alignment greedily, choosing the most similar target for each source from left to right. Finally, we project slot labels s along that alignment, restore BIO prefixes, and use the result s' to form the new training example (u', i, s') for $\mathcal{D}_P$.

4 Experimental Setup

We experiment with our data augmentation approach by simulating the introduction of a new feature on a public dataset with English utterances and on internal data from a real-world dialog system in German.

4.1 Data

SNIPS The Snips NLU dataset (Coucke et al., 2018) is a commonly used benchmark for intent classification and slot labeling covering 7 intents and 39 slots. It is typically split into 13,084 train, 700 validation, and 700 test examples. We run 7 experiments, each time picking one of the intents to be the new feature. For each, we use the full training and validation data of the remaining 6 intents as $\mathcal{D}_O^{train}$ and $\mathcal{D}_O^{val}$ (together about 11,815 instances), while we sample 5% of the data for the new feature to be the seed data $\mathcal{D}_N^{train}$ and $\mathcal{D}_N^{val}$ (together 100 instances). We use the regular test split of SNIPS for evaluation. Since SNIPS is moderate in size and our experimental setup involves several randomized parts, we repeat the experiments multiple times and report averaged results. We use each of the 7 intents as the simulated new feature, sample three different 5% seed subsets for it and train models with 10 different seeds, resulting in 210 different experiment runs per approach.

Internal Data In addition to SNIPS, we use randomly sampled data from logs of a commercial dialog system in German to simulate the introduction of 5 different features that have been introduced in the past. For each, we use $|\mathcal{D}_N^{train}| = 450$ and $|\mathcal{D}_N^{val}| = 50$ seed utterances. The datasets $\mathcal{D}_O^{train}$ and $\mathcal{D}_O^{val}$ are significantly larger and cover several hundred other intents, not including examples for any of the 5 features we simulate to be new. For evaluation, we test against two disjoint test sets, one from the same distribution as $\mathcal{D}_O^{train}$ and one having examples for the new feature.

4.2 Intent and Slot Model

For intent classification and slot labeling, we use a neural model inspired by existing work. Tokens are embedded as 300-dimensional vectors and encoded with a bi-directional GRU with hidden size 512. For intent classification, the concatenated final representations of the GRUs are fed through a 300-d ReLU layer, followed by dropout and a final softmax layer over intent labels. For slot labeling, we use a similar two-layer network, but apply it to the GRU states from each timestep to predict slot labels per token. The model is trained with Adam, a batch size of 64 and dropout of 0.2 until the performance stops improving on the validation data. We train the model separately for intent classification and slot labeling, which departs from recent work that observed improvements through joint training (see Section 2). However, we choose this setup to be able to study the effect of our data augmentation on both tasks in isolation.

4.3 Compared Approaches

In our experiments, we compare the following variations of the techniques presented in Section 3:

- **Baseline** A model trained using just the seed data $\mathcal{D}_N$ and existing data $\mathcal{D}_O$. The goal for all data augmentation approaches is to improve upon this baseline.
- **Upsampling5** A variation of the baseline that repeats each seed example 5 times to match the amount of training data for the new feature obtained with the augmentation techniques below.
- **Beam1** A model trained on $\mathcal{D}_O$, $\mathcal{D}_N$ and $\mathcal{D}_P$, where the latter comes from decoding one paraphrase for each seed in $\mathcal{D}_N$ with beam search. We use a beam size of 5 and normalize scores by length.
- **Beam5** Same as Beam1, but using the top-5 hypotheses from the beam for each seed.
- **Shuffle+Beam5** Same as Beam5, but also using input shuffling. We select 5 paraphrases per seed based on PSCO from the beams across all shuffles as described in Section 3.2.1.
- **Rand1** Same as Beam1, but decoding with random sampling. We sample with $\alpha = 2$ and $\beta = 3$.[4]
- **Rand5** Same as Rand1, but sampling 5 paraphrases per seed.
- **Shuffle+Rand5** Same as Shuffle+Beam5, but using random sampling as in Rand1. We sample 3 paraphrases per shuffle and then select 5 across all shuffles by PSCO as in Section 3.2.1.

Given the number of seed examples available for the new feature – 100 on SNIPS and 500 on our internal dataset – the absolute number of examples added by the augmentation methods and baselines are 100 or 500 on SNIPS and 500 or 2,500 on the internal dataset. Beyond these two settings, it would be interesting to explore more seed-paraphrase-ratios and to determine an optimal one, but we leave this for future work to instead focus on the comparison of different sampling techniques in this work.

The paraphrasing model is trained on $\mathcal{D}_N \cup \mathcal{D}_O$ with a vocabulary of 30,000 tokens. Embeddings for tokens, intents and slot names are initialized with 300-dimensional GloVe vectors (Pennington et al., 2014) (or randomly if not found) and updated during training. The encoder and decoder GRUs have a size of 300 and two layers each. We train with a batch size of 64 and dropout of 0.3 with early stopping.

[4]We selected these parameters after manually inspecting paraphrases for a range of parameter settings and found them to represent a good trade-off between novelty and quality.

A	i'm looking for a table at a pasta restaurant in serbia	1 *BookRestaurant, Dish=pasta, Type=restaurant, Country=serbia* i'm looking to find a table at a pasta restaurant in serbia 2 *BookRestaurant, Dish=pasta, Country=serbia, Type=restaurant* book a pasta in serbia restaurant 3 *BookRestaurant, Type=restaurant, Dish=pasta, Country=serbia* i'm looking for a restaurant that serves pasta in serbia
B	please play the newest music by evil jared hasselhoff	1 *PlayMusic, Sort=newest, Artist=evil jared hasselhoff* play the newest music by evil jared 2 *PlayMusic, Artist=evil jared hasselhoff, Sort=newest* play music by evil jared hasselhoff on the newest
C	closest movie theatre with no time for sergeants	1 *SearchScreeningEvent, Sp.-Rel=closest, Movie=no time for ...* is the closest no time for sergeants at the movie theatre

Figure 3: Paraphrases of varying quality sampled for seeds from SNIPS using beam search and input shuffling (Shuffle+Beam5). Left side shows the seed, right the input and output of the paraphrase model. Note that the examples were manually chosen to demonstrate both good and bad examples.

4.4 Evaluation Metrics

Our main evaluation metric is the performance of the downstream tasks, measured by accuracy for intent classification and slot F1-score for slot labeling. We focus on the change in performance when using data augmentation compared to the baseline, as it shows whether the augmentation is helpful to bootstrap a new feature. Because bootstrapping a new feature can be detrimental to existing ones, we separately look at changes for the new feature (should be positive) and for existing features (should not be negative).
To be able to get additional insights into relative strengths of different paraphrase generation approaches, we also adopt the following metrics to compare paraphrases:

- **ESCO** Similar to PSCO defined in Eq. 2, we compute the *exact slot carry over (ESCO)*, which differs in counting only slots whose complete slot value has been carried over. Where we use PSCO=1 for candidate selection, this stricter version can still differentiate among the selections.
- **Novelty** In our use case, new examples need to be different from the seeds, otherwise, the augmentation degenerates to simply upsampling the data. We therefore use BLEU (Papineni et al., 2002) as a measure of similarity and compute for each paraphrase u' of seed u the score $1 - \text{BLEU4}(u, u')$. Higher scores indicate higher novelty. We report averages over all sampled paraphrases.
- **Diversity** In addition, we want paraphrases to be diverse instead of sampling the same paraphrase repeatedly. To measure diversity, we also compute $1 - \text{BLEU4}(u', u'')$, but among pairs of paraphrases u', u'' of the same seed. We report the average over all pairs, higher means more diversity.

Note that we report the three metrics introduced above primarily to better understand how paraphrases sampled with different methods differ. Whether such differences, e.g. higher novelty or diversity, are in fact beneficial is reflected by the downstream performance measured as intent accuracy and slot F1.

5 Results

5.1 Generated Paraphrases

Figure 3 shows several examples for paraphrases obtained with our approach. As the examples show, utterances range from slight variations of the original utterance, dropping only some tokens (B1, A1) or changing the order as encouraged by input shuffling (A2, A3), to more strongly deviating ones that can in the extreme become unnatural and ungrammatical (B2, C1).

Table 1 compares the generated paraphrases with our quality metrics. As expected, paraphrases from random sampling are more novel and more diverse than from beam search, but at the expense of carrying

	Upsampling5	Beam1	Beam5	Shuffle +Beam5	Rand1	Rand5	Shuffle +Rand5
PSCO	1.000	0.988	0.984	**1.000**	0.911	0.910	**1.000**
ESCO	1.000	0.969	0.956	**0.972**	0.773	0.772	0.878
Novelty	0.000	0.486	0.574	0.742	0.767	0.766	**0.864**
Diversity	0.000	–	0.522	0.594	–	0.837	**0.881**

Table 1: Quality comparison of paraphrases generated on SNIPS with different approaches along our four metrics. Bold marks best technique per metric (excluding the edge case Upsampling5).

SNIPS	New Feature					Existing Features			
Method	#Train	IC	Δ	SL	Δ	IC	Δ	SL	Δ
Baseline	100	88.1		52.1		98.9		88.8	
Upsampling5	600	90.5	+2.46	63.6	+11.47	98.8	-0.10	88.4	-0.40
Beam1	200	89.7	+1.67	57.7	+5.58	98.8	-0.06	88.7	**-0.08**
Beam5	600	90.6	+2.50	62.9	+10.78	98.8	-0.10	88.5	-0.23
Shuffle+Beam5	600	88.7	+0.67	63.1	+11.02	98.7	-0.15	88.5	-0.23
Rand1	200	91.0	+2.97	57.5	+5.36	98.9	**-0.02**	88.6	-0.11
Rand5	600	92.0	**+3.95**	63.4	+11.32	98.8	-0.14	88.5	-0.27
Shuffle+Rand5	600	91.3	+3.26	64.7	**+12.66**	98.8	-0.14	88.4	-0.31

Table 2: Intent classification (IC) accuracy and slot labeling (SL) F1-scores on the SNIPS test set (English) after adding generated paraphrases as additional training data. Δ denotes the absolute change with regard to the baseline. Each result is an average over 210 runs of the experiments (see Section 4).

over less slots, indicating that too novel paraphrases might no longer fully represent the intended interpretation. Note that by design (see Section 3.2.1), PSCO is 1 for both input shuffling–based methods, but ESCO reveals that the slot carry over rate is in fact lower for random sampling. While we observe a trade-off between slot carry over and novelty/diversity in general, it is notable that input shuffling increases novelty and diversity while at the same time also achieving the highest slot carry over, for both beam search and random sampling. Hence, our proposed combination of shuffling and selection by PSCO appears to be an effective way to improve paraphrases among all dimensions.

5.2 Effect on Downstream Tasks

Table 2 shows the effect of adding the paraphrases as training data on SNIPS. Comparing the variations of our method, we make the following observations: Across both tasks and both sampling methods, generating more paraphrases per seed utterance (500 vs. 100) improves performance more. With regard to sampling methods, the results show that random sampling helps more than beam search, which is in line with the increased novelty and diversity observed in Table 1. However, that effect is less pronounced for slot labeling, as the lower slot carry over coming with higher novelty and diversity is more problematic for learning slot labeling, but almost irrelevant for intent classification. Input shuffling, on the other hand, seems beneficial for slot labeling but not for intent classification, which could be because word order is less relevant for intent classification in general. When evaluating on existing features, there is a small performance drop, but it is negligibly small, in particular when compared to the gain on the new feature. Finally, we note that the Upsampling baseline is strong, but, while it beats several of our ablations, its performance is not as good as our full method, indicating that the higher novelty and diversity of the sampled paraphrases (see Table 1) are beneficial for the downstream task.

On the internal data, we compared only our full method including input shuffling across the two sampling methods. The average improvement over all simulated new features is even bigger for intent

Internal Data	New Feature				Existing Features			
	ICΔ		SLΔ		ICΔ		SLΔ	
New Feature	S+B5	S+R5	S+B5	S+R5	S+B5	S+R5	S+B5	S+R5
WeatherForecast	+1.48	+4.81	+3.91	+4.98	-0.02	-0.04	-0.02	-0.01
SendMessage	+4.84	+7.73	+0.65	+0.47	-0.13	-0.11	-0.01	-0.02
PlayMusic	+8.79	+10.12	+1.35	-0.12	+0.11	+0.10	-0.05	-0.13
MovieListing	+9.50	+10.32	+1.28	+0.92	+0.05	+0.05	-0.09	-0.06
ApplianceOnOff	+0.91	+12.53	+1.25	+0.79	-0.02	-0.18	+0.04	-0.07
Average	+5.10	+9.10	+1.69	+1.41	0.00	-0.04	-0.03	-0.06

Table 3: Intent classification (IC) and slot labeling (SL) performance change on internal data (German) across 5 (simulated) new features when adding paraphrases generated with Shuffle+Beam5 (S+B5) or Shuffle+Rand5 (S+R5). Numbers are absolute changes with regard to the baseline.

classification, as Table 3 shows, again with random sampling providing bigger improvements than beam search. Also, none of the improvements for the new feature changes the performance on existing features substantially, as desired. Compared to SNIPS, however, improvements are smaller for slots, and beam search outperforms random sampling. We attribute the smaller improvement to the fact that more (shared) slots are already known to the model since the set of existing features is larger. Thus, the baseline can already perform better, which makes it harder to improve. The breakdown by new feature in Table 3 illustrates that changes are different across features, which we attribute to the degree of variety within utterances for specific features and their similarity to already existing features.

5.3 Discussion and Future Work

Our results are promising and show that improvements are possible even with small seed datasets. That being said, we want to emphasize that finding the paraphrases that provide the combination of novelty, diversity and meaning-preservation that benefit the downstream task the most is challenging, as two different models with hyper-parameters and the sampling parameters are involved. Tuning them carefully is therefore very costly, and we leave exploring methods for that to future work. In addition, future work should study the optimal number of paraphrases for a new feature, whether sampling methods, in particular slot shuffling, work equally well across domains and how the paraphrase generation model could benefit from language model pretraining. Finally, we would like to point out that in practice, our approach can be combined with any other data augmentation technique discussed in Section 2, such as using machine translation or user feedback, to bootstrap new features even further.

6 Conclusion

We proposed a data augmentation approach for seed data of new features in dialog systems that relies on interpretation-to-text paraphrase models, shuffling and random sampling to generate paraphrases and alignment-based label projection. We demonstrated that using the resulting new training examples improves performance for intents and slots on an English benchmark and German dialog system data.

Acknowledgements

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

References

Zhiyu Chen, Harini Eavani, Yinyin Liu, and William Yang Wang. 2019. Few-shot NLG with Pre-trained Language Model. *arXiv*, 1904.09521.

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.

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.

Quynh Do and Judith Gaspers. 2019. Cross-lingual Transfer Learning with Data Selection for Large-Scale Spoken Language Understanding. In *Proceedings of the 2019 Conference on EMNLP and the 9th IJCNLP*, pages 1455–1460, Hong Kong, China.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of End-to-End Natural Language Generation: The E2E NLG challenge. *Computer Speech & Language*, 59:123–156.

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.

Markus Freitag and Scott Roy. 2018. Unsupervised Natural Language Generation with Denoising Autoencoders. In *Proceedings of the 2018 Conference on EMNLP*, pages 3922–3929, Brussels, Belgium.

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.

Judith Gaspers, Penny Karanasou, and Rajen Chatterjee. 2018. Selecting Machine-Translated Data for Quick Bootstrapping of a Natural Language Understanding System. In *Proceedings of the 2018 Conference of the NAACL-HLT*, pages 137–144, New Orleans, LA, USA.

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.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A Deep Generative Framework for Paraphrase Generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5149–5156, 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.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 146–157, Vancouver, Canada.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase Generation with Deep Reinforcement Learning. In *Proceedings of the 2018 Conference on EMNLP*, pages 3865–3878, Brussels, Belgium.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable Neural Paraphrase Generation. In *Proceedings of the 57th Annual Meeting of the ACL*, pages 3403–3414, Florence, Italy.

Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled Text Generation for Data Augmentation in Intelligent Artificial Agents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 90–98, Hong Kong, China.

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.

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.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the Annual Meeting of the ACL*, pages 311–318, Philadelphia, PA, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on EMNLP*, pages 1532–1543, Doha, Qatar.

Roberto Pieraccini, Evelyne Tzoukermann, Zakhar Gorelov, Jean-Luc Gauvain, Esther Levin, Chin-Hui Lee, and Jay G. Wilpon. 1992. A Speech Understanding System Based on Statistical Representation of Semantics. In *Proceedings of the 1992 IEEE ICASSP*, page 193–196, San Francisco, CA, USA.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural Paraphrase Generation with Stacked Residual LSTM Networks. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2923–2934, Osaka, Japan.

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.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the ACL*, pages 1073–1083, Vancouver, Canada.

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.