Discovering Novel Features from User Utterances in Intelligent Virtual Assistants

Anonymous ACL submission

Abstract

Discovering new user intents is one of the key tasks in intelligent virtual assistants, and has been intensively studied in the prior work. However this does not cover all types of user features, due to diversity on their granularity levels. In this paper, we present DNF – the first approach for discovering novel user-requested features in virtual assistants. Our method is based on state-of-the-art language model BERT for utterance representation, with two novel contributions for distilling novel features. The first contribution is a feature-aware representation model, using a multi-stage fine-tuning process for transferring knowledge from feature-labeled data. The second contribution is a detection model for clustering and labelling the novelty of feature clusters. Extensive experiments on real-word datasets demonstrate the effective of our novel method both with respect to the quality of the utterance representation model and the novel features that are produced.

1 Introduction

Advances in Natural Language Understanding (NLU) have led to the emergence of intelligent virtual assistants such as Apple's Siri, Microsoft's Cortana, Google's Assistant and Amazon's Alexa. Over the last years, with the assistants becoming more intelligent, the adoption by end-users continued to increase. Standing at the core of a virtual assistant is a NLU model for parsing and understanding user utterances. Two of the key tasks of an NLU model are to determine the user's intent and slots given a single user utterance. Determining the intent guides the assistant to perform the proper actions as response to the user. For example, user utterances "I would like some music" and "play despacito" express the intent *PlayMusic* while "how is the weather?" and "is it raining today?" express the intent *GetWeather*. Intents can be further grouped into domains, e.g., intents *PlayMusic*, *RateSong* both can be assigned to domain *Music*. Furthermore, detecting slots and their corresponding values within an utterance gives information about objects upon which the action should be performed. For example, "play despacito" contain the slot *SongName: despacito*.

Over time, users start building up expectations about the features that the virtual assistant offers. Unsupported features might cause friction across the virtual assistant's components including the NLU model. In terms of NLU, a novel user feature could be mapped to a new combination of domain(s), intent(s), slot(s) and/or their values (where each is not necessarily novel). For example, while *PlayMusic* intent and *ServiceName* slot exist independently in the training data, potentially in different domains, their combination is never seen causing friction with the NLU model when parsing utterances like "play despacito on spotify". Consequently, given a large set of new user utterances, it becomes crucial to analyse and discover such features that are frequently requested by users but are still unsupported by the NLU model.

The task of novel intent/domain discovery has been intensively studied in the prior work, by proposing both totally unsupervised techniques (e.g., Liu et al., 2021) and (semi-)supervised methods for incorporating existing knowledge from labeled data (e.g., Vedula et al., 2020a; Lin et al., 2020; Zhang et al., 2021). Currently, most of the existing works study feature discovery for those covering a novel single intent and/or a single domain. However, this does not often cover all types of user-requested features that could occur on different granularity levels. Consider the two user utterances "play despacito", and "play despacito on spotify", while they belong to the same intent *PlayMusic*, the implementation for handling them must be different. The former requires the virtual assistant to play a song on the device, while the later asks for playing the same song on a specific service (spotify). For the second utterance, not

only does the virtual assistant need to recognize the slot *SongName*, it also needs to recognize the slot *ServiceName* to call the corresponding APIs from the third-party provider. Intuitively, an NLU model which does not support playing music on a specific service would not be trained on utterances with intent *PlayMusic* and containing slot *ServiceName* and consequently would have a high probability of mislabelling the utterance. Since the intent is not novel, applying standard intent/domain discovery would fail to discover those requests.

To allow for a more fine-grained discovery, and close the gap between user requests and the underlying models, we define a user ***feature*** to be any combination of domain(s), intent(s) and slot(s) and their values and move towards discovering clusters of utterances with novel feature definitions rather than only focusing on novel intent(s)/domain(s).

In this paper, we propose DNF – a semi-supervised method for discovering novel features from a given set of user utterances with respect to an underlying NLU model. DNF exploits knowledge from predefined existing features to guide the discovery process. Our method consists of a cascaded system with two steps: feature clustering and novelty detection. First, we employ state-of-the-art language model BERT (Reimers and Gurevych, 2019) coupled with multi-stage fine-tuning to produce feature-aware representations of the user utterances, transferring knowledge from existing feature-labeled data. Using the feature-aware representations, input user utterances could then be clustered into features. Second, we train a classifier to label each of the resulted feature clusters as novel or already supported by a given NLU model.

The salient contributions of our paper are as follows:

- We propose a new task: Feature Discovery, where, given a set of user utterances and a trained NLU model, we expect to extract clusters of user features (rather than intents/domains) that are unsupported by the NLU model
- We present DNF – the first approach for discovering novel features from user utterances, utilizing knowledge from predefined feature-labeled data.
- Our method includes a novel feature-aware representation model for encoding user utterances, constructed through a multi-stage fine-tuning process, and a novelty detection model for distilling novel features from utterance clusters.
- We implemented our approach and conducted extensive experiments on real-world data, including our own industrial internal dataset and a public dataset of user utterances augmented with feature labels. Experimental results demonstrate the effectiveness of our method. The feature-augmented public dataset will be released upon acceptance.

2 Related Work

Discovering novel intents from user utterances has been well addressed in the earlier works, with methods proposed in both unsupervised and (semi-)supervised settings for incorporating knowledge from labeled data. These include works for recognizing input texts with novel domains or intents separately, using various techniques such as constrained deep adaptive clustering (Lin et al., 2020), deep aligned clustering (Zhang et al., 2021), contrastive learning (Gao et al., 2021), capsule network (Liu et al., 2019; Xia et al., 2018), open intent extraction (Vedula et al., 2020b), and so on (e.g., Lin and Xu, 2019; Shivakumar et al., 2019; Yan et al., 2020). Intents and domains are also discovered jointly by Vedula et al., 2020a, using hierarchical linking to form an intent-domain taxonomy. The task is also performed jointly with slot filling in recent works (Wang et al., 2018; Goo et al., 2018; Kim et al., 2017; Castellucci et al., 2019). All of the above works require a small amount of labeled data as prior knowledge to guide the discovery process.

The most prominent work for detecting novel intents without any prior knowledge is from Liu et al. (2021), by employing a pre-trained network for generating sentence embeddings, and K-Means for intent clustering. However, as no labeled knowledge is involved, their method is limited, and inapplicable to different kinds of data.

None of the above mentioned works is ready for discovering novel features from user utterances as we are. Our method DNF builds up on state-of-the-art BERT language model for utterance representation, and makes contributions on transferring feature knowledge from labeled data.

3 Methodology

Figure 1 depicts the overview of our approach DNF. Given a set of user utterances $u_1, u_1, ..., u_n$, we aim at detecting utterance clusters of novel features from the set. Our method solves this problem into two steps. First, we assign each utter-

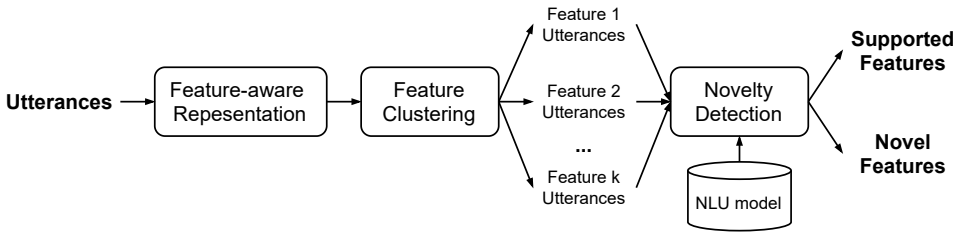

Figure 1: Approach Overview.

ance u_i a feature label and eventually produce a set of feature-labeled utterances $(u_1, f_1), ..., (u_n, f_k)$ where $f_1, f_2, ..., f_k$ is the list of k unique features. This is done by employing an utterance representation model specifically trained with feature-labeled data to embed the input utterances into vector space. Then, we use K-Means to cluster the feature-aware representation vectors into the k features. Second, a feature novelty detection model is built to classify each of the k feature clusters as novel or already supported by the virtual assistant. Our detection model exploits signals from the NLU model, which is trained for recognizing utterance intent and slots.

DNF is a semi-supervised technique, meaning that it exploits knowledge from existing data. We take into account two types of training data for DNF: *feature-labeled* and *feature-unlabeled* data.

- Feature-labeled training data ($Train_L$): is a set of utterances, where each utterance goes along with its feature label f. Moreover, we also have other information including intent label I and slot labels $S = \{s_1, s_2, ...\}$ where each s_i is a pair of slot name and its position/value in the utterance such as *SongName:despacito* or *Service:spotify*.
- Feature-unlabeled training data ($Train_U$): is a set of utterances along with their intent labels and slot labels, without feature labels. Even though having no feature label, this auxiliary training data is still useful for the feature discovering process as we can exploit information about utterances' intent and slot labels.

3.1 Feature-Aware Utterance Representation

We encode each user utterance as a high-dimensional vector using an utterance representation model specifically geared for feature awareness. To this end, we employ the state-of-the-art language model SBERT – Sentence BERT (Reimers and Gurevych, 2019), which is a BERT model pretrained for generating sentence embeddings. Specifically, by feeding an utterance u into BERT, we get a list of token embeddings

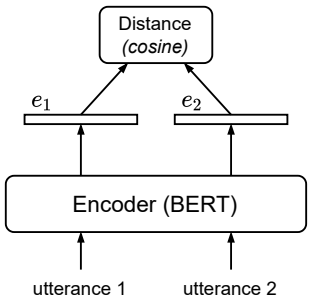

Figure 2: Fine-tuning with Utterance Similarity task.

$[CLS, t_1, t_2, ..., t_m]$ where *CLS* is the classification token. By applying mean-pooling, we achieve the representation vector of u:

$$e_u = \textit{mean-pooling}([CLS, t_1, t_2, ..., t_m])$$

We further transfer knowledge about features from training data into this model through a multi-stage fine-tuning process as described below.

3.1.1 Utterance Similarity

This fine-tuning stage is depicted in Figure 2, in which we use a Siamese Neural Network (SNN) training paradigm for transferring feature-based knowledge into the model. The intuition behind this fine-tuning stage is to directly optimize the utterance embeddings based on their similarity. In particular, pairs of utterances (u_1, u_2) are encoded into their embedding vectors e_1 and e_2. Their utterance similarity is computed as the cosine distance between the two vectors $cos(e_1, e_2)$, which is optimized towards the true distance label depending on if the two input utterances belong to the same feature or not. As our training data are both feature-labeled ($Train_L$) and feature-unlabeled ($Train_U$), we consider 3 kinds of training samples for this fine-tuning stage:

1. Both u_1 and u_2 come from the same $Train_L$ feature: (u_1, u_2) is a positive sample.
2. Both u_1 and u_2 come from $Train_L$, but different features: (u_1, u_2) is a negative sample.

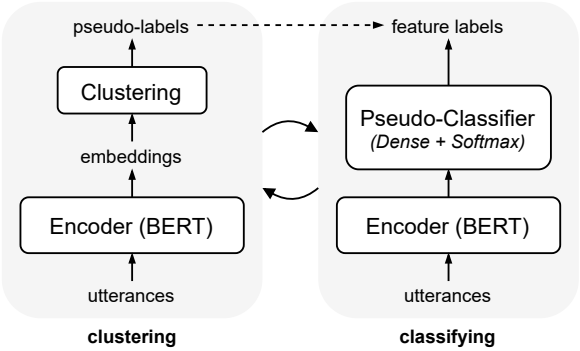

Figure 3: Fine-tuning with Pseudo-Classification task.

3. u_1 comes from $Train_L$ and u_2 comes from $Train_U$: (u_1, u_2) is a negative sample.

Note that for case 3, it is possible that the training sample is a false-negative, since the actual feature-label of u_2 could be the same as of u_1. However, we hypothesize that the number of such false-negative utterance pairs is much lower than the true-negative ones. And even if those false-negative u_2 are accidentally included in the training, u_2 will be pulled far away from u_1, but not as far as the true-negative ones, due to the inherent similarity between u_1 and u_2, which is learnt by the positive samples from case 1.

As the number of utterance pairs is quadratic, we do not include all of them into the training process. Instead, we use a simple way to build a random dataset for each training epoch as follows. For each utterance u_1 in $Train_L$ set, we randomly sample an utterance u_2 from its same feature as a positive sample, and 3 utterances either from $Train_L$, but different features, or from $Train_U$ as the negative samples; hence maintaining the rate of 1:3 between the number of positive and negative pairs.

3.1.2 Pseudo Classification

Utterance similarity stage only considers pairwise distances between pairs of utterances, without putting a global constraint on all utterances at the same time. Moreover, this stage only optimizes the distances among feature-labeled utterances in $Train_L$ set, and between $Train_L$ utterances and the feature-unlabeled ones from $Train_U$ set. However, it does not explicitly optimize the distances among $Train_U$ utterances, which will be additionally taken into account in pseudo-classification fine-tuning stage.

The overview of this stage is illustrated in Figure 3. Pseudo classification is an iterative process, alternating between the two steps: *clustering* and *classifying*, inspired by the DeepCluster method from the earlier work for image classification (Caron et al., 2018).

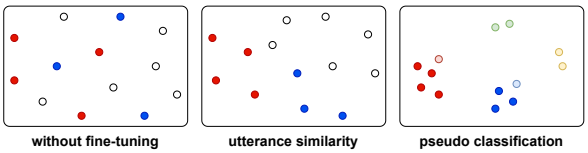

Figure 4: Utterance Similarity vs. Pseudo Classification. Solid red and blue dots are $Train_L$ utterances.

In the *clustering* step, we first encode the utterances from training data into their representation vectors. Then a clustering algorithm is used to group the representation vectors into clusters, which are the utterance pseudo-labels that will be used in the *classifying* step. We use COP-K-Means (Wagstaff et al., 2001) as the clustering algorithm – an extension of K-Means that allows putting constraints on the utterance vectors: which utterances must be grouped in the same clusters, which must not. In our case, the feature-labels of $Train_L$ is used as the constrained background knowledge for COP-K-Means.

In the *classifying* step, we fine-tune the BERT encoder in a classification task, using the pseudo-labels generated from the clustering step as the groundtruth feature labels. For doing this, we plug on top of the encoder a pseudo-classifier, consisting of a dense layer followed by softmax, and train the whole model using cross-entropy loss. In the original DeepCluster approach, the pseudo-classifier has to be reinitialized in each iteration, since the indices of the pseudo-labels are permuted randomly after the clustering step. This makes the training process become slower for convergence as the network parameters of the pseudo-classifier cannot be reusable. To alleviate this issue, we adopt the cluster centroid alignment technique proposed by Zhang et al. (2021), to re-assign the pseudo-label indices from the clustering step, aligning them with the pseudo-classifier trained from the previous iteration. Using the aligned pseudo-labels for classifying, the pseudo-classifier can be reusable, hence speeding up the training.

Comparing to the utterance similarity stage, pseudo classification additionally clusters $Train_U$ utterances, either into $Train_L$ clusters or into totally new ones. Figure 4 compares the expected effect between the two fine-tuning stages.

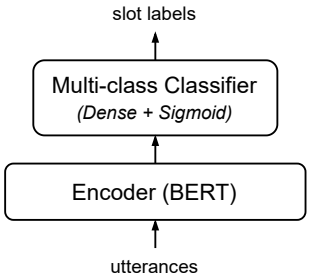

Figure 5: Fine-tuning with Slot Classification task.

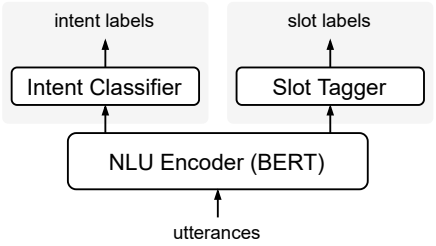

Figure 6: NLU model.

3.1.3 Slot Classification

Information about named entities appearing in user utterances is a good resource for feature-awareness, and is taken into account in slot classification stage. In contrast to utterance similarity and pseudo classification fine-tuning stages, slot classification does not directly pull/push the utterances close to or far away from each other in the embedding space. Instead, we aim to make the representation model aware of the presence of the named entity slots in the input utterances. This is modeled as a multi-class classification task by training the representation model to detect which entity slots appear in the input utterances (e.g., *SongName*, *Service*, etc.), as demonstrated in Figure 5. Specifically, we plug on top of the encoder (on top of the CLS token) a multi-class classifier, comprising of a dense layer followed by sigmoid activation, and fine-tune the whole model using binary-cross-entropy loss. After fine-tuning, we remove the classifier. Note that in this stage, we do not consider the exact position of the slots in the utterances, but only their presence.

3.1.4 Joint-training

We fine-tune the representation model in different stages to achieve different goals. The stages can be applied either sequentially, or jointly, to prevent the model from overfitting on a specific task. This is especially important for the slot classification stage, as we want the model to learn the named entity slots present in the input utterances, but at the same time also need to group such slot-based-similar utterances into feature clusters. For joint-training, we combine the losses from different fine-tuning stages using a weighted sum, and train the joint model with respect to the combined loss.

3.2 Feature Novelty Detection

3.2.1 Feature Clustering

The trained feature-aware utterance representation model is used for discovering novel features from live-traffic unlabeled user utterances. We encode the utterances into their representation vectors and use K-Means to cluster them into features. For automatically picking the optimal number of clusters k for K-Means, we employ two different techniques, namely the Elbow method (Thorndike, 1953) and the Silhouette score (Rousseeuw, 1987). As it is common that the unlabeled data is a mix of both novel and supported features with respect to the virtual assistant, we then classify whether each of the resulted feature cluster as novel or not. This is done through a feature novelty detection model, which is described next.

3.2.2 Novelty Detection

We detecting novel feature clusters by exploiting signals from a pseudo NLU model[1], which is trained for recognizing utterance intent and entity slots, in a joint paradigm, as demonstrated in Figure 6. The intent classifier (IC) head is plugged on top of the encoder CLS token, and optimized towards the true intent labels of the utterances. The slot tagger head for named entity recognition (NER) is instead plugged on top of utterance tokens for token classification, and optimized towards the slot labels encoded with BIO schema (Ramshaw and Marcus, 1995). Note that the NLU model considers the exact position of the named entities, which is different from slot classification in the utterance representation model. For inferencing in NER, we additionally add a constrained decoding layer (He et al., 2017), to extract the tag sequence with the highest probability that satisfies the BIO constraints: each I-token must follow a B or an I token of the same slot.

For novelty detection, we define the novelty confidence of a feature cluster $C = \{u_1, u_2, ...\}$ as the average novelty of its utterances:

$$conf_{novel}(C) = \frac{1}{|C|} \sum_{u \in C} conf_{novel}(u)$$

[1]We do not use our industrial NLU model in this work.

where the utterance novelty is computed by aggregating signals from the NLU model and the feature-aware utterance representation model:

- *NER confidence (ner)*: the confidence of named entity recognition, produced by the slot tagger of the NLU model. We tested with two variants: average over tokens *(ner_{avg})*, and minimum over tokens *(ner_{min})*.
- *IC confidence (ic)*: the confidence of the most probable intent class produced by the intent classifier of the NLU model.
- *PC confidence (pc)*: the confidence of the most probable pseudo cluster, produced by the pseudo classifier from the utterance representation model.

We sum up different kinds of signals for more robust detection:

$$conf_{novel}(u) = \frac{ner_{min}(u) + ic(u) + pc(u)}{3}$$

Feature clusters with novelty score within a predefined boundary decision threshold are labeled as novel.

4 Experiments

We conducted experiments to evaluate our approach DNF with respect to the two major components: *utterance feature-aware representation* and *feature novelty detection*.

Datasets. We evaluated our method with two datasets: (1) the SNIPS dataset (Coucke et al., 2018), consisting of 14K English user utterances spanning 7 intents: *GetWeather*, *AddToPlaylist*, *RateBook*, *BookRestaurant*, *PlayMusic*, *SearchCreativeWork* and *SearchScreeningEvent*; and (2) our own industrial internal dataset with user utterances from an European language.

We manually augment the SNIPS dataset with feature-labels and end up with total 43 user features from 7 intents, spanning over 3 feature types: combination of slots, combination of slots and values, and intents. The two intents are added as whole features are *RateBook* and *BookRestaurant*. This is used to evaluate the effectiveness of the method for discovering features at different granularities. From total 43 features, we select 19 into the feature-labeled training set $Train_L$, and additional 13 features into the feature-unlabeled training set $Train_U$, where we stripe out their groundtruth feature labels and only use them for validation purpose. These are considered as supported features by the NLU, and we only include 40% of their utterances into training. Hence, test data comprises utterances from all 43 features, the model should be able to distinguish between supported and novel ones. This augmented dataset will be released upon acceptance.

Our large industrial dataset has total 41 features predefined by our commercial virtual assistant, from which we include 31 features into our training data. The features span two feature types: single-domain and cross-domain.

4.1 Evaluation of the Representation Model

Setup. We compare our utterance representation model trained with different combination of fine-tuning stages:

- *NO fine-tuning*: We directly use the SBERT base model for utterance representation.
- *Utterance Similarity (US)*: The model is trained with utterance similarity stage only.
- *Pseudo Classification (PC)*: The model is trained with pseudo classification stage only.
- *US→PC*: The model is trained with both utterance similarity and pseudo classification stages sequentially.

We also evaluated the integration of information about named entity slots into the model through two pipelines:

- *Slot Classification (SC)→US→PC*: The model is trained with 3 stages slot classification, utterance similarity and pseudo classification sequentially.
- *(SC+US)→PC*: Slot classification and utterance similarity are jointly trained to prevent overfitting.

Metrics. For evaluating, we first encode test utterances using the trained representation model, then use K-Means to clusters them into features (with k is assumed to be known, which is the total number of features of the dataset). The resulted feature clusters are evaluated against the groundtruth feature labels using 3 metrics: Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Accuracy (ACC). ACC is computed by using the Hungarian algorithm, to find the maximum matching between the predicted utterance feature clusters and their true feature classes.

Table 1: Utterance representation performance.

Training Stages	SNIPS			Internal Dataset*		
	NMI	*ARI*	*ACC*	*NMI*	*ARI*	*ACC*
NO fine-tune	0.626	0.309	0.396	==== *baseline* ====		
US only	0.737	0.451	0.512	+0.096	+0.147	+0.149
PC only	0.728	0.374	0.471	+0.116	+0.161	+0.130
US→PC	0.749	0.475	0.537	+0.116	+0.178	**+0.166**
SC→US→PC	0.766	0.474	0.557	+0.104	+0.166	+0.129
(SC+US)→PC	**0.782**	**0.531**	**0.605**	**+0.137**	**+0.216**	+0.140

* Due to confidentiality, we report the relative gains/losses over baseline.

Table 2: Performance of different feature types.

Feature Type		SNIPS			Internal Dataset*		
		NMI	*ARI*	*ACC*	*NMI*	*ARI*	*ACC*
By Novelty	*supported*	0.836	0.698	0.746	=== *baseline* ===		
	novel	0.741	0.566	0.637	-0.177	-0.230	-0.187
By Granularity	*slot*	0.753	0.610	0.626	–	–	–
	slot+value	0.811	0.590	0.646	–	–	–
	intent	0.547	0.512	0.804	–	–	–
	single-dm.	–	–	–	=== *baseline* ===		
	cross-dm.	–	–	–	-0.053	+0.018	-0.028

* Due to confidentiality, we report the relative gains/losses over baseline.

Main results. Table 1 shows the performance of the utterance representation models fine-tuned with different pipelines. We observe that our models excel the baseline in all pipelines. The model trained with both tasks US and PC performs better than if being trained with only one of these tasks. The performance is further boosted with the integration of the SC task. Moreover, joint training of SC and US tasks gives better results than sequential training.

In subsequent experiments, we will stay with using the best observed training pipeline *(SC+US)→PC*.

Performance of different feature types. In Table 2, we report the performance of the representation model for each feature type, refined by their novelty and their granularity. We observe that for both datasets, our model clusters utterances from supported features better than from unseen ones. When dividing by granularity, on SNIPS dataset, we achieve better results on clustering at slot and slot+value levels than at intent level, which is reasonable as our training data contain only 2 features on this level, with only 1 is included in training. For our internal data, clustering for single-domain features is slightly better than for cross-domain ones.

Intent-agnostic vs. Intent-targeted discovery. The default evaluation setting is intent-agnostic, where models are trained with data across all intents. For deeper analysis, for SNIPS dataset, we additionally conducted experiments in intent-targeted setting, in which we only train/test with features from the same intent. These are used to compare if the models will work better if being trained with focused intent/domain data.

Table 3: Intent-agnostic vs. Intent-targeted Discovery.

Intent	Intent-agnostic			Intent-targeted		
	NMI	*ARI*	*ACC*	*NMI*	*ARI*	*ACC*
AddToPlaylist	0.544	0.402	0.595	**0.671**	**0.573**	**0.789**
GetWeather	**0.624**	**0.589**	**0.847**	0.377	0.143	0.487
PlayMusic	0.700	0.429	0.527	**0.747**	**0.625**	**0.672**
SearchCreat.Work	0.631	0.431	0.598	**0.766**	**0.602**	**0.707**
SearchScrn.Event	0.639	0.474	0.634	**0.740**	**0.541**	**0.678**

Table 4: Ablation study results.

Model	SNIPS			Internal Dataset*		
	NMI	*ARI*	*ACC*	*NMI*	*ARI*	*ACC*
Standard	0.782	0.531	0.605	=== *baseline* ===		
Ablation	0.769	0.444	0.523	-0.081	-0.132	-0.087
Upperbound	0.812	0.530	0.672	–	–	–

* Due to confidentiality, we report the relative gains/losses over baseline.

In Table 3, we report the results for this study. The models trained with focused-intent features perform generally better than when being trained with unfocused data, except for intent *GetWeather*. Our data has only 4 features from this intent, where only 2 are included in training, hence it is difficult for the model to learn feature-patterns from this data.

Ablation study. For understanding the impact of feature-labeled $Train_L$ and feature-unlabeled $Train_U$ training data on the model performance, we conduct ablation study by comparing our model trained with both kinds of data (standard model) against the model trained with only feature-labeled $Train_L$ (ablation model). Moreover, for SNIPS dataset, we additionally compare both of them to an *upperbound* model, in which we also include the true feature labels of $Train_U$ utterances during training. Hence, the upperbound model is also trained with only feature-labeled data, similar to the ablation model, but with more features. This is not a realistic model, and is only used to establish an upperbound by assuming having a large amount of feature-labeled data already available.

Table 4 shows the results for this study. The standard model performs consistently better than the ablation model on all metrics for both datasets, with NMI and ACC gains reaching 8-9%, and ARI gain of 13% on our internal dataset. This shows the importance of including feature-unlabeled data during training. Moreover, on SNIPS dataset, the upperbound model performs the best.

Table 5: Performance of automatically choosing k.

Method	SNIPS				Internal Dataset*			
	k	*NMI*	*ARI*	*ACC*	k	*NMI*	*ARI*	*ACC*
Gold_k	43	0.782	0.531	0.605	41	=== *baseline* ===		
Silhouet.	24	0.761	0.511	0.593	35	-0.030	-0.063	-0.013
Elbow	30	0.790	0.573	0.632	39	+0.002	+0.003	+0.020

* Due to confidentiality, we report the relative gains/losses over baseline.

Table 6: Performance of the NLU model.

Signal	SNIPS				Internal Dataset*			
	AUC	*Prec.*	*Rec.*	*F1*	*AUC*	*Prec.*	*Rec.*	*F1*
ner_{avg}	0.603	0.407	1.000	0.579	====== *baseline* ======			
ner_{min}	0.693	**0.750**	0.545	0.632	+0.077	+0.076	0.000	+0.043
ic	0.517	0.500	0.636	0.560	-0.181	-0.083	-0.100	-0.091
pc	0.692	0.615	0.727	**0.667**	-0.042	**+0.167**	-0.200	-0.020
$conf_{novel}$	**0.700**	**0.750**	0.545	0.632	**+0.087**	**+0.167**	0.000	**+0.091**

* Due to confidentiality, we report the relative gains/losses over baseline.

Automatically choosing the number of clusters. In the above experiments, we use the number of groundtruth features k from test data for evaluating the representation model. However, for predicting novel features, this number is usually unknown in reality. In Table 5, we report the model performance when choosing k automatically using 2 different methods: the Elbow method (Thorndike, 1953) and the Silhouette score (Rousseeuw, 1987). We observe that for both datasets, the Elbow method works slightly better than Silhouette score, with the predicted k closer to the groundtruth value. In terms of metrics, Elbow method even produces slightly better feature clusters than the baseline.

We use the feature clustering output from Elbow method as the input for feature novelty detection, as described in the next section.

4.2 Evaluation for Feature Novelty Detection

The evaluation for feature novelty detection has two experiments. First, we evaluate the performance of our trained NLU model for detecting novel features with respect to the groundtruth feature clusters, using different signals as described in Section 3.2.2. Second, as an end-to-end experiment, we evaluate the performance of novelty detection on the predicted feature clusters, with the number of clusters k predicted by the Elbow method.

4.2.1 NLU Model Evaluation

For evaluating the NLU model, we use different signals as described in Section 3.2.2 as the measures for predicting a groundtruth feature cluster as *novel* or *supported*, including: ner_{avg}, ner_{min}, *ic*, *pc* and the combined signal $conf_{novel}$. For each of them, we report 4 metrics: the precision-recall AUC (with *"novel"* is positive class), and the precision, recall, F1 at the confidence-threshold with the optimal F1.

The results are shown in Table 6. We observe that the combined signal $conf_{novel}$ performs the best in all metrics for our internal dataset. On the SNIPS dataset, $conf_{novel}$ still excels against other signals in terms of AUC and precision, and places the second best in F1.

Table 7: Novelty detection for predicted clusters.

Method	SNIPS					Internal Dataset*				
	#	*Prec.*	*Rec.*	*F1*	*Avg. Nov.*	#	*Prec.*	*Rec.*	*F1*	*Avg. Nov.*
all	30	0.233	0.636	0.341	0.175	39	====== *baseline* ======			
ner_{avg}	23	0.238	0.455	0.312	0.176	16	+0.295	0.000	+0.288	+0.285
ner_{min}	5	**0.600**	0.273	0.375	**0.499**	12	+0.462	0.000	+0.400	+0.446
ic	21	0.238	0.455	0.312	0.176	8	+0.545	-0.200	+0.340	**+0.533**
pc	10	0.400	0.364	**0.381**	0.325	9	+0.462	-0.200	+0.305	+0.353
$conf_{novel}$	5	**0.600**	0.273	0.375	**0.499**	9	**+0.573**	-0.100	**+0.410**	+0.504

* Due to confidentiality, we report the relative gains/losses over baseline.

4.2.2 Novelty Detection of Predicted Clusters

We perform novelty detection on the predicted clusters generated in the feature clustering step. Again, we use the signals mentioned earlier at their best confidence-thresholds for novelty boundary decision.

For evaluation, we assign a feature label l_C for each predicted cluster C from its main feature (i.e., the feature with the most number of utterances). We report 4 metrics: (1) precision: the fraction of unique feature labels that are actually novel over the total number of clusters predicted as novel; (2) recall: the fraction of unique feature labels that are actually novel over the total number of groundtruth novel features; (3) F1; and (4) the *average novelty* of predicted novel clusters, where the *novelty* of a cluster C is defined as:

$$novelty(C) = \begin{cases} H(\frac{|u \in C : f_u = l_C|}{|C|}, \frac{|u \in C : f_u = l_C|}{|u : f_u = l_C|}), & \text{if } l_C \text{ is novel} \\ 0, & \text{otherwise} \end{cases}$$

In other words, the cluster novelty is computed as the harmonic mean between the rate of the number of utterances from its label to the size of the cluster, and to the total number of utterances from that label over all clusters, similar to the principle of F1, but on utterance level.

Table 7 reports the number of novel clusters predicted by each signal and their quality metrics. *all* is the baseline, where we consider all feature clusters as novel. We observed that $conf_{novel}$ signal still performs the best in terms of precision and average novelty on SNIPS data, together with ner_{min}. On

our internal dataset, $conf_{novel}$ also achieves highest precision and F1, places the second best in terms of average novelty, while discovering almost novel features in the data (losing only 10% recall, comparing to the *all* baseline).

5 Conclusion

We have presented DNF – the first method for extracting novel features from user utterances in intelligent virtual assistants, utilizing minimal feature knowledge from labeled data. We evaluated DNF with various configurations on real-world datasets and observed significant improvements over baselines, showing the effectiveness of our method.

For future work, we plan to improve our multi-stage fine-tuning process for better utterance representation, conduct more experiments with our different internal datasets, and finally integrate the approach into our commercial product.

References

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, volume 11218 of *Lecture Notes in Computer Science*, pages 139–156. Springer.

Giuseppe Castellucci, Valentina Bellomaria, Andrea Favalli, and Raniero Romagnoli. 2019. Multi-lingual intent detection and slot filling in a joint bert-based model. *CoRR*, abs/1907.02884.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Xibin Gao, Radhika Arava, Qian Hu, Thahir Mohamed, Wei Xiao, Zheng Gao, and Mohamed AbdelHady. 2021. Graphire: Novel intent discovery with pretraining on prior knowledge using contrastive learning.

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 North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 753–757. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 473–483. Association for Computational Linguistics.

Young-Bum Kim, Sungjin Lee, and Karl Stratos. 2017. ONENET: joint domain, intent, slot prediction for spoken language understanding. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017, Okinawa, Japan, December 16-20, 2017*, pages 547–553. IEEE.

Ting-En Lin and Hua Xu. 2019. Deep unknown intent detection with margin loss. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5491–5496. Association for Computational Linguistics.

Ting-En Lin, Hua Xu, and Hanlei Zhang. 2020. Discovering new intents via constrained deep adaptive clustering with cluster refinement. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8360–8367. AAAI Press.

Han Liu, Xiaotong Zhang, Lu Fan, Xuandi Fu, Qimai Li, Xiao-Ming Wu, and Albert Y. S. Lam. 2019. Reconstructing capsule networks for zero-shot intent classification. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 4798–4808. Association for Computational Linguistics.

Pengfei Liu, Youzhang Ning, King Keung Wu, Kun Li, and Helen Meng. 2021. Open intent discovery through unsupervised semantic clustering and dependency parsing. *CoRR*, abs/2104.12114.

Lance A. Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora, VLC@ACL 1995, Cambridge, Massachusetts, USA, June 30, 1995*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.

Journal of computational and applied mathematics, 20:53–65.

Prashanth Gurunath Shivakumar, Mu Yang, and Panayiotis G. Georgiou. 2019. Spoken language intent detection using confusion2vec. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 819–823. ISCA.

Robert L Thorndike. 1953. Who belongs in the family? *Psychometrika*, 18(4):267–276.

Nikhita Vedula, Rahul Gupta, Aman Alok, and Mukund Sridhar. 2020a. Automatic discovery of novel intents & domains from text utterances. *CoRR*, abs/2006.01208.

Nikhita Vedula, Nedim Lipka, Pranav Maneriker, and Srinivasan Parthasarathy. 2020b. Open intent extraction from natural language interactions. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2009–2020. ACM / IW3C2.

Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 577–584. Morgan Kaufmann.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 309–314. Association for Computational Linguistics.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3090–3099. Association for Computational Linguistics.

Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert Y. S. Lam. 2020. Unknown intent detection using gaussian mixture model with an application to zero-shot intent classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1050–1060. Association for Computational Linguistics.

Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. 2021. Discovering new intents with deep aligned clustering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14365–14373. AAAI Press.