Identifying Domain Independent Update Intents in Task Based Dialogs

Prakhar Biyani [*]
Twitter Inc.
1355 Market St #900
San Francisco, CA 94103
pbiyani@twitter.com

Cem Akkaya
Yahoo Research, Oath Inc.
701 First Ave
Sunnyvale, CA 94089
cakkaya@oath.com

Kostas Tsioutsiouliklis
Yahoo Research, Oath Inc.
701 First Ave
Sunnyvale, CA 94089
kostas@oath.com

Abstract

One important problem in task-based conversations is that of effectively updating the belief estimates of user-mentioned slot-value pairs. Given a user utterance, the intent of a slot-value pair is captured using dialog acts (DA) expressed in that utterance. However, in certain cases, DA's fail to capture the actual *update intent* of the user. In this paper, we describe such cases and propose a new type of semantic class for user intents. This new type, Update Intents (UI), is directly related to the type of update a user intends to perform for a slot-value pair. We define five types of UI's, which are independent of the domain of the conversation. We build a multi-class classification model using LSTM's to identify the type of UI in user utterances in the Restaurant and Shopping domains. Experimental results show that our models achieve strong classification performance in terms of F-1 score.

1 Introduction

An important part of dialog management in dialog systems is to detect the type of update to be performed for a slot after every turn in order to keep track of the dialog state. (The dialog state reflects the user goals specified as slot-value pairs.) User dialog acts (Young, 2007) express the user's intents towards slots mentioned in the conversation. They are extracted in the spoken language understanding (SLU) module and are utilized by the downstream state tracking systems to update belief estimates (Williams et al., 2013; Lee and Stent, 2016; Henderson et al., 2014c). However, currently used dialog acts do not capture the update intended by the user in the following cases:

1. **Implicit denials:** User denials for slot-values are expressed using the "deny" and "negate" dialog acts. However, these acts only address explicit negations/denials such as "no", "I do not want ⟨slot-value⟩'. But a user may express denial for a value implicitly. Consider utterances 8 and 9 in Table 1 where a user *adds* and *removes* people from a slot, PNAMES, which contains names of people going to an event. Current SLU systems would detect the "inform" dialog act in both utterances and, hence, would miss the (implicit denial) "remove" update.
2. **Updates to numeric slots**: Numeric slots are the slots whose values can be increased and decreased in addition to getting set/replaced. Since dialog acts do not capture the "increase" and "decrease" intents towards a numeric value, such updates cannot be handled using dialog acts alone. For example, consider utterances 4, 5 and 6 in Table 1 where the value of a numeric slot, NGUEST (number of guests in an invite), is set, increased and decreased respectively. The dialog act expressed in these utterances is "inform" which does not convey the update type.
3. **Preference for slot values:** The "inform" dialog act specifies values for slots but does not take into account the preferences for any particular slot value(s). Consider utterances 1, 2 and 3 in Table 1 where the location slot (LOC) is referred. In utterance 2, the user is equally interested in the three locations ("Ross", "Napa" and "San Jose"). However, in utterance 3 the user prefers "Gilroy" over other values and intends to *replace* the old values with "Gilroy". Clearly, the SLU output does not capture this change in the user intent.

We posit that identifying the above intents in user utterances as a part of SLU would improve estimation of user goals in task based dialogs. To ad-

[*]The work was done when the author was at Yahoo Research, Oath Inc.

Id	**User utterance**	**Expected SLU output**	**SLU output with update intents**
		Task: Restaurant search	
1	Find French restaurants in Ross and Napa.	inform(LOC=Ross|Napa)	inform(**append**(LOC=Ross|Napa))
2	Show some in San Jose too.	inform(LOC=San Jose)	inform(**append**(LOC=San Jose))
3	Show me in Gilroy instead.	inform(LOC=Gilroy)	inform(**replace**(LOC=Gilroy))
		Task: Restaurant reservation	
4	Book a table for 4 at Olive Gardens.	inform(NGUEST=4)	inform(**replace**(NGUEST=4))
5	Add 4 more seats.	inform(NGUEST=4)	inform(**increaseby**(NGUEST=4))
6	Can you remove 2 seats.	inform(NGUEST=2)	inform(**decreaseby**(NGUEST=2))
7	Actually make it for 5.	inform(NGUEST=5)	inform(**replace**(NGUEST=5))
		Task: Restaurant reservation	
8	Invite Joe, Mike and John for drinks at SoMar today.	inform(PNAMES=[John &Mike& Joe])	inform(**append**(PNAMES=Joe&Mike &John))
9	Take Joe off the list.	inform(PNAME=Joe)	inform(**remove**(PNAMES=Joe))

Table 1: Example user-bot conversations with only user utterances. For illustration, only the relevant slots are shown in the SLU output.

dress the above issues, we propose five generic ***update intents*** (UI's) which are directly related to the type of update expressed by the user: Append, Remove, Replace, IncreaseBy and DecreaseBy, and build a model to identify them in a user utterance. Table 2 defines the five UI's. We model the problem of identifying UI's as a multi-class classification. For a user utterance, we classify UI's for all the slot-values present in the utterance into one of the five classes. We treat an utterance as a sequence of tokens and slot-values, and perform sequence labeling using LSTM's for the classification. It should be noted that the focus of this work is on identifying the UI's in user utterance and not on investigating the mechanisms of using them for belief tracking, which is part of our larger goal.

UI's are generic in nature and independent of the dialog domain. Given a slot type (such as numeric), they can be applied to any slot of that type. This enables transfer learning across similar slots in different domains. To demonstrate this, we experiment with two domains (shopping and restaurants) and define three types of slots: 1. Numeric slots, 2. Conjunctive multi-value (CMV) slots, and 3. Disjunctive multi-value (DMV) slots (explained in Section 3.1.1). We then delexicalize slot-values in user utterances with the corresponding slot type (not slot name) and conduct cross-domain training and testing experiments. Experimental results demonstrate strong classification performance in individual domains as well as across domains.

Contributions: 1) We propose a new semantic class of slot-specific user intents (UI's) which are directly related to the update a user intends to perform for a slot. 2) The proposed UI's enable effective updates to slots. 3) Our models predict UI's with high accuracy. 4) We present a novel delexicalization approach which enables transfer learning of UI's across domains.

2 Related Work

Although we are not aware of any prior work on identifying update intents, our current work is related to dialog act identification and dialog state tracking. Here, we review works in these two areas and contrast them against ours.

Dialog act identification: Dialog acts (DA) in an utterance express the intention of their speaker/writer. Identifying DA types has been found to be useful in many natural language processing tasks such as question answering, summarization, and spoken language understanding (SLU) in dialog systems. A variety of DA's have been proposed for specific application tasks and domains, such as email conversations (Cohen et al., 2004), online forum discussions (Bhatia et al., 2012; Kim et al., 2010), and dialog systems (Young, 2007). The latter is relevant to this work. In dialog systems, DA's are used to infer a user's intention towards either the slots or the conversation in general. Some of the DA's used in dialog systems are inform, confirm, deny, and negate. Previous works on DA identification in dialog systems have used a range of approaches like n-grams based utterance level SVM classifier (Mairesse et al., 2009), SVM classifier built on weighted n-grams using word confusion networks incorporating ASR uncertainties and dialog context (Henderson et al., 2012), log linear models (Chen et al., 2013), and recurrent neural networks (Hori et al., 2015, 2016; Ushio et al., 2016). This work is similar to DA identification in the sense that both the UI's and the DA's express certain semantics in the utterance and are independent of the dialog domain. However, there are important differences: 1) DA's mainly reflect the intent towards the conversation; however, UI's convey the type of update a user wants to a particular slot. 2) DA's can be slot-independent (such as *hello*, *negate*, etc.) whereas UI's are always defined with respect to a slot.

Dialog State Tracking: Dialog state tracking (DST) entails updating the conversation state (also known as belief state) after every dialog turn. A conversation state is a probability distribution over competing user goals which are expressed in the form of slot-value pairs. For a user utterance, DST relies on SLU to get a list of k-best hypotheses of DA's and slot-value pairs expressed in the utterance. To update the belief state, DST approaches utilize DA's by using their SLU confidence scores as features (Ren et al., 2013; Kim and Banchs, 2014), encoding the DA's using n-gram vectors weighted by the SLU confidence scores (Henderson et al., 2014c; Mrkšić et al., 2015), and using rule-based updates (Lee and Stent, 2016). Recently, efforts have been made to bypass the SLU output and learn update mechanisms directly from user utterance (Mrksic et al., 2017). Though DA's are important for updating belief state, as explained in Section 1, certain updates like implicit denials, numeric updates, and slot preferences are not handled by the DA's used in the dialog systems literature. UI's, on the other hand, are proposed to address this problem. The work by Hakkani-Tür et al. (Hakkani-Tür et al., 2012) on identifying *action updates* in a multi-domain dialog system is closely related to the current work. Some of their action updates are similar to UI's. However, unlike the current work, they did not deal with numeric updates and did not distinguish between types of multi-value slots (explained in Section 3.1.1).

3 Approach

In task-based dialogs, users complete a task by giving sequences of utterances in which they specify slot-values with corresponding intents. Dialog systems extract this information using dialog act detection and slot-filling as part of SLU. The most common and helpful intents for completing a task are setting a value for a slot and denying a particular value for a slot. Traditionally, these two intents are determined by the *inform* and *deny* dialog acts. However, as explained in Section 1, a user may not always set and deny a value explicitly. While denials can be implicit, relative preferences can also be provided for slot-value(s). In case of numeric slots, user can set a value by incrementing or decrementing the previous values of slots. All these common scenarios are not handled by the *inform* and *deny* dialog acts.

UI Type	Definition
Append	Append a specified value to the slot.
Remove	Remove a specified value from the slot.
IncreaseBy	Increase a value of a slot by a specified amount.
DecreaseBy	Decrease a value of a slot by a specified amount.
Replace	Replace the value of a slot by a specified value.

Table 2: Types of update intents and their definitions.

In this work, we propose a new set of slot-specific intents which are directly related to the type of update expressed towards the slot. We call these intents ***update intents*** or UI's. The UI's express five common types of updates:

1. **Append**: A user specifies a value or multiple values for a multi-value slot. This is equivalent to "appending" the specified value(s) to a multi-value slot. (Refer to Section 3.1.1 for the definition of multi-value and numeric slots).

2. **Remove**: A user denies a value or multiple values for a multi-value slot implicitly or explicitly. This is equivalent to "removing" the specified value(s) from a multi-value slot.

3. **Replace**: A user specifies a preference for a slot value in case of multi-value slots. In case of numeric (single value) slots, this intent expresses setting and re-setting of a slot value (Utterances 4 and 7 in Table 1). This UI is defined with respect to the slot-value for which the preference is expressed. For example, in the utterance "I would prefer San Jose over Gilroy" the UI for San Jose is *replace*, whereas for Gilroy it is *remove*. Note that in case of multi-value slots, *replace* cannot be decomposed into a combination of an "append" and a "remove" update when the "remove" intent is

not specified. For example, in “I would prefer San Jose” there is no “remove” intent and, hence, simply using the “append” intent for San Jose would not extract the preference for San Jose.

4. **IncreaseBy**: A user specifies a value by which a particular numeric slot’s value is to be increased.

5. **DecreaseBy**: A user specifies a value by which a particular numeric slot’s value is to be decreased.

Table 1 shows examples of the above five UI’s. The third column shows the expected SLU output with UI’s.

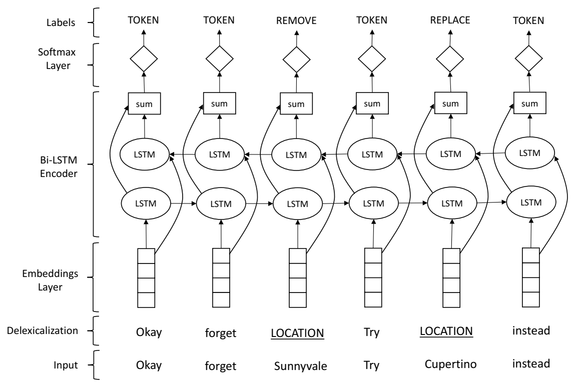

Figure 1: Model architecture

3.1 Modeling

Given a user utterance, the goal is to determine UI’s for all the slot-values present in it. We formulate this task as a classification problem. Given a user utterance and the mentioned slot-values, classify the update intents for all the slot-values in one of the five classes: Append, Remove, Replace, IncreaseBy and DecreaseBy.

We model the above problem as a sequence labeling task. We treat a user utterance as a sequence of words and slot-values. The labels for slot-values are the corresponding UI’s whereas for words (which are not slot-values), we define a generic label “TOKEN”. For model optimization and error computation, we do not consider the “TOKEN” labels. Figure 1 describes our model architecture. We used Bidirectional LSTMs (Graves and Schmidhuber, 2005) for sequence labeling. For input representation, we used GloVe word embeddings (Pennington et al., 2014). For regularization, we used dropout and early stopping. We give more details about model parameters in Section 5.

3.1.1 Learning Across Domains

In many cases, it is not possible to list all the values of a slot in the ontology. Even if the values are listed, it may not be practical to generate a training data containing all the values, if there are too many values for the slot. In such cases, it is beneficial to unlink the learning from particular slot-values and link it, instead, to the slot itself. This is because the word patterns used to refer to values of the same slot are similar and hence can be shared across the values. For example, a user would use similar word patterns to refer to values of slot “LOCATION”. One way to do this is by replacing slot-values in utterances with the name of the slot. This is also called *delexicalization* and has been used successfully in many previous works (Henderson et al., 2014c; Mrkšić et al., 2015). In our model, we also delexicalize slot-values with the name of the slot as shown in Figure 1.

Delexicalization with slot names is helpful in generalizing to slot-values not seen in the training data in one domain. However, it cannot be used for cross-domain generalization as different domains may not share the same slots. To address this problem, we define three generic slot types depending upon the values (numeric/non-numeric) a slot can take and whether a slot can take multiple values simultaneously (list-based) or not:

1. **Numeric slot**: Slots whose values can be increased and decreased. NGUEST in Table 1 is a numeric slot. A numeric slot is a single value slot, i.e., “appending” and “removing” (multiple) values are not allowed for numeric slots. This slot supports IncreaseBy, DecreaseBy and Replace UI’s.

2. **Disjunctive multi-value (DMV) slot**: Slots which can take multiple values only in disjunction, i.e., when user specifies those values as options. In a restaurant search domain, examples of DMV slots are location and cuisine. LOC slot in Table 1 is a DMV slot.

3. **Conjunctive multi-value (CMV) slots**: These are list type slots which can take multiple values in conjunction. Examples are slots containing names of people going to an event, items in a shopping list, etc. Slot PNAMES in Table 1 is a CMV slot. Both CMV and DMV slots support Append, Remove and Replace UI’s.

Different domains may not share same slots but they often share slots with same types. For example, list of groceries in the shopping domain and list of people in a dinner invite in the restaurant domain are of type CMV. Similarly, the number of guests in a dinner reservation and the number of items of a particular grocery are of type numeric. If we delexicalize slot-values with slot types, we can transfer learning for a slot type in one domain to the same slot type in another domain.

There can be cases where there are different ways (word patterns) to specify updates to two slots even if they are of the same type, because of differences in the corresponding domains or some other reason. For example, lets say slots S_1 and S_2 are in different domains but share the same slot-type S and we have training data for slot S_1. S_1 and S_2 have similarities owing to their common slot-type but have certain differences in the ways users can express update intents for them. In such a case, to generate training data for S_2, we would need data capturing the differences between the two slots because the examples with common features are already contained in S_1's training data. Generating this additional data is easier than generating the full data for S_2. The amount of additional data required will depend upon the degree by which the slots (S_1 and S_2) differ. When applied to a large number of slots and domains, this strategy would significantly reduce the time and effort that goes into data generation. To demonstrate this, we conduct training and testing experiments on two domains, restaurants and online shopping, and report results in Section 5.2.

4 Data Preparation

To evaluate our approach, we needed dialogs containing numeric, CMV, and DMV slots in the domain ontology along with the proposed update intents expressed in user utterances. Existing datasets with annotated dialog acts such as WOZ 2.0 (Wen et al., 2017), ATIS (Dahl et al., 1994), Switchboard DA corpus [1], Dialog State Tracking Challenge (DSTC) datasets (Henderson et al., 2014a; Williams et al., 2013; Henderson et al., 2014b) and ICSI meeting recorder DA corpus (Shriberg et al., 2004) did not satisfy these requirements. DSTC 2 and DSTC 3 datasets contained DMV slots but not the CMV (list-based slots) and numeric slots [2]. DSTC 4 (Kim et al., 2015), DSTC 5 (Kim et al., 2016) and DSTC 6 (Boureau et al., 2017) introduced a new set of speech acts which contains "HOW_MUCH" act for the numeric price range and time slots. However, the act only supports the *Replace* UI and not the *IncreaseBy* and *DecreaseBy* UI's. Moreover, the three datasets are not public. Therefore, we generated our own datasets.

We generated user utterances in two domains: restaurants and online shopping. In each domain, eight human editors generated user utterances independent of each other. The sets of editors were different across the two domains. Table 3 explains the slots used in the two domains. For each domain, we defined certain tasks which are listed in Table 4. Editors wrote conversations to complete those tasks. Since this was not a real dialog system, editors were asked to assume appropriate bot responses based on their requests such as "Okay", "Added", "Removed", "Done" during the conversation. Also, since the focus was on identifying update intents and not on the overall SLU, (dialog act detection, slot-filling, etc.), we did not build our own custom slot-tagger and, instead, asked the editors to annotate the slot-values with the corresponding slot name in addition to the update intents. Here is a sample annotation for the task "restaurant reservation".

$$Drop \overbrace{\underbrace{one}_{\text{DecreaseBy}}}^{\text{NGUEST}} person, \overbrace{\underbrace{Joe}_{\text{Remove}}}^{\text{PNAMES}} can't\ make\ it.$$

For the shopping domain, 305 conversations with 1308 user utterances were generated. For the restaurant domain, 280 conversations with 1323 user utterances were generated. The distribution of utterances among editors is 96, 110, 212, 79, 176, 258, 211 and 166 for the shopping domain. For the restaurant domain, the editorial distribution is 322, 181, 116, 106, 143, 107, 109 and 239. The distribution of Append, Remove, Replace, IncreaseBy and DecreaseBy UI's for restaurant domain is 1022, 301, 601, 92, 112 respectively. The corresponding distribution for the shopping domain is 1249, 241, 521, 297, 90. Note that, an utterance may have more than one UI.

5 Experiments and Results

5.1 Experimental Setting

We implement the proposed architecture in Section 3 using Keras (Chollet et al., 2015), a high-level neural networks API, with the Tensorflow (Abadi et al., 2015) backend. Training is

[1]http://compprag.christopherpotts.net/swda.html

[2]The *pricerange* slot in DSTC2 and 3 is a categorical (and not a numeric) slot with a fixed set of values

Slot	Type	Definition
		Restaurants
PNAMES	CMV	List of names of people in a reservation.
NGUEST	Numeric	Number of people in a reservation.
MENUITEMS	CMV	List of menu items to be ordered.
CUISINE	DMV	Type of cuisine.
LOCATION	DMV	Location (city) of restaurant.
		Shopping
GITEMS	CMV	List of grocery items.
QTY	Numeric	Quantity of a particular (grocery or apparel) item.
ASTORE	DMV	Apparel shopping store.
AITEMS	CMV	List of apparels.
COLOR	DMV	Color of apparel.
SIZE	DMV	Size of apparel.

Table 3: List of slots, their type and definitions in the restaurant and shopping domains.

done by mini-batch RMSProp (Hinton et al., 2012) with a fixed learning rate. In all our experiments, mini-batch size is fixed to 64. Training and inference are done on a per-utterance level. The embedding layer in the model is initialized with 300-dimensional Glove word vectors obtained from common crawl (Pennington et al., 2014). Embeddings for missing words are initialized randomly with values between -0.5 and 0.5.

Evaluation: Using a random split of train and test sets would have examples from the same editor in both train and test sets which would bias the estimation. Therefore, we split our data into eight folds corresponding to the eight editors, i.e., each fold contains examples from only one of the editors. To evaluate our models, we train and validate on the data from seven folds and test the performance on the held-out (eighth) fold. We run this experiment for each editor, i.e., eight times, and average results across the eight folds. For validation, we use 15% of the training data. We use precision, recall and F-1 score to report the performance of our classifiers. Overall classification performance metrics are computed by taking the weighted average of the metrics for individual classes. A class's weight is the ratio of the number of instances in it to the total number of instances.

Parameter tuning: In each experiment, 15% of the current training set is utilized as a development set for hyper-parameter tuning and the model with best setting is applied to the test set to report the results. We tune learning rate, dropout via grid search on the development set. In addition, we utilize early stopping to avoid over-fitting. The optimal hyper-parameter settings for our classification experiments (reported in Table 5) is $dropout = 0.3$, $learningrate = 0.001$ for the restaurants domain and $dropout = 0.25$, $learningrate = 0.001$ for the shopping domain.

Baseline: We used n-grams based multinomial logistic regression as a baseline. N-grams based models have been extensively used in text classification (Biyani et al., 2016, 2013, 2012). Such models have also been found to be effective as semantic tuple classifiers for dialog act detection and slot filling tasks (Chen et al., 2013; Henderson et al., 2012). Since there can be multiple slot-values and, hence, multiple UI's expressed in a user utterance, the entire utterance cannot be used to extract n-grams for all the expressed UI's. Therefore, we segment user utterances into relevant contexts for the slot-values and classify the contexts into one of the five UI classes. A context for a value is an ordered list of words which are indicative of the update to be performed for the value. We use two approaches for segmentation based on the k words window approach: a) hard segmentation, b) soft segmentation. In the first approach, we assign the words around the value to its context based on the following constraints:

1. If an utterance contains only one value, the entire utterance is taken as the context for the value.

2. If there are n words (s.t. $n < 2k$) between two slot values then the preference is given to the right value. That is, k words are assigned to the context of the right value and $n - k$ words are assigned to the context of the left value.

3. All the words to the left of the first value (in the utterance) are added to the value's context. Similarly, all the words to the right of the last value are added to its context.

In soft segmentation, we do not perform a hard assignment of the words, between the two values to the context of one of the values. Instead, we encode the words into one of these categories based on its position with respect to the value and if it is in between two values (and let the model learn weights for words in each category): 1) towards left of a value and between two values, 2) towards right of a value and between two values, 3) towards left of a value, 4) towards right of a value.

We extracted unigrams and bigrams from the context of slot-values. We experimented with different window sizes and k=2 gave the best results.

Task	**Informable slots**	**Supported update intents**	**Example user utterances**
		Restaurants	
Search	location, cuisine	Append, Remove, Replace.	Utterances 1 to 3 in Table 1
Reservation	pnames, nguest	All UI's	Utterance 4 to 7 in Table 1
Order food	menuitems	Append, Remove, Replace.	1. Order a *cheese burger* and a *coke can*. 2. Can you do a *diet coke*.
		Shopping	
Grocery	gitems, qty	All UI's	1. *One dozen white eggs* and *one* pound of *apples*. 2. Add *two* more pounds of *apples*.
Apparel	aitems, astore, color, size, qty	All UI's.	1. Show me *blue sweaters* at *Target*. 2. I think *black* will suit better.

Table 4: Tasks in the two domains with corresponding info slots, supported update intents and example utterances. Slot-values in the utterances are in italics.

Class	Prec.	Re.	F-1	#Instances
		Restaurants		
Append	90.64	92.86	91.74	1022
Remove	85.66	81.40	83.48	301
Replace	89.05	93.34	91.15	601
IncreaseBy	95.29	88.04	91.53	92
DecreaseBy	96.88	83.04	89.42	112
Overall	90.02	90.65	90.27	2128
		Shopping		
Append	92.22	95.26	93.72	1245
Remove	85.71	74.69	79.82	241
Replace	85.63	82.50	84.04	520
IncreaseBy	98.30	97.31	97.80	297
DecreaseBy	91.36	82.22	86.55	90
Overall	90.86	90.18	90.45	2393

Table 5: Classification results on the two domains.

Method	Prec.	Re.	F-1
		Restaurants	
1.Baseline(soft)	84.28	82.84	81.96
2.Baseline(hard)	85.74	84.96	84.32
3.Model-delex	$90.66^{1,2}$	$85.14^{1,2}$	$87.74^{1,2}$
4.Model	$90.02^{1,2}$	$90.65^{1,2,3}$	$90.27^{1,2,3}$
		Shopping	
1.Baseline(soft)	82.62	81.12	79.86
2.Baseline(hard)	82.81	81.55	80.32
3.Model-delex	$86.30^{1,2}$	82.10	$84.05^{1,2}$
4.Model	$90.86^{1,2,3}$	$90.18^{1,2,3}$	$90.45^{1,2,3}$

Table 6: Comparison of different classification models on the two domains. Superscripts' denote statistical significance over the corresponding model with a p-value of 0.05 or less. Model-delex is the proposed model without delexicalization.

5.2 Results

In this section, we present the results of our classification and domain-independence experiments.

5.2.1 Classification Results

Table 5 shows the classification results on the two domains. For both the domains, our model achieves more than 90% overall F-1 scores. Per-class results are also strong. The Append, Replace, and IncreaseBy classes achieve more than 91% F-1 scores for the restaurant domain. For the shopping domain, IncreaseBy is the best performing class (97% F-1) followed by Append and DecreaseBy. Despite having significantly fewer examples compared to the other classes, IncreaseBy and DecreaseBy perform very well. One of the reasons for this behavior could be that after delexicalization, for these two classes, there is only one slot (QTY in shopping and NGUEST in restaurants) for which the model learns the patterns. Other than these two classes, this slot is shared by the Replace class. Hence, given a delexicalized numeric slot-value, the model needs to discriminate between these three classes whose relative distribution is much smoother than the overall distribution of the five classes. For the other delexicalized slot-values, the model discriminates between Append, Remove and Replace, where the majority class has a much higher number of examples than the minority Remove class. Hence, we see that the Append class performs significantly

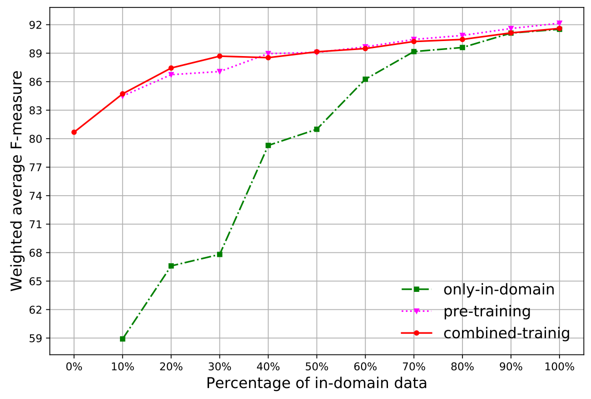

Figure 2: Restaurant as out-domain and shopping as in-domain.

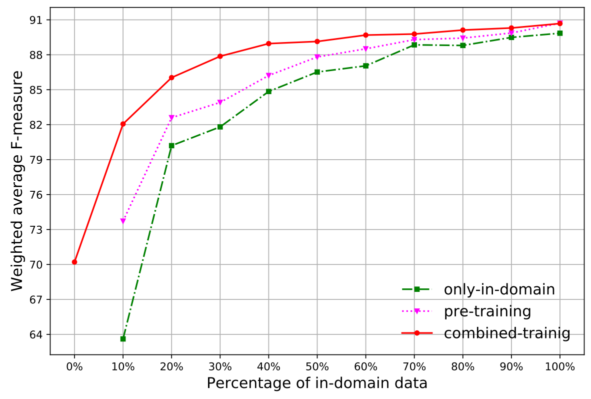

Figure 3: Shopping as out-domain and restaurant as in-domain

better than the Remove class.

We also compare our model with the two baselines explained in Section 5.1. Table 6 presents these results. We see that the proposed model significantly outperforms the two baselines. This shows that for UI classification, contextual information around a slot-value is captured much more effectively using sequence models than static classifiers. We also experimented with our model without delexicalization to verify the gains it brings. As can be seen, delexicalization does improve the performance in both domains.

5.2.2 Domain Independence Results

We conducted experiments to explore if learning of UI's in a domain can be used to predict UI's in a different domain. We use one of the domains as the "in-domain" (where learning is transferred to) and the other as the "out-domain" (where learning is transferred from). For this experiment, we set aside 20% of the in-domain data as the test set. At each step, we use 15% of the training data as the validation set. We explored two settings:

1. **Combined-training**: In this setting, we start by training our model on the entire out-domain data and then, incrementally, add a fraction (10%) of the in-domain data (left after taking out the test data) to the current training data, retrain the model (from scratch) on the combined data.

2. **Pre-training**: Here, we train a model on the out-domain data and fine-tune it with the in-domain data. At each step, we add a fraction (10%) of the in-domain data to the current training data and refit the pre-trained out-domain model on it by initializing the model weights to the weights of the model trained on the out-domain data.

Figures 2 and 3 report the results of these two settings. For Figure 2, the model trained only on the out-domain (restaurant) data achieves F-1 score of more than 80% on the in-domain test set. As we add more in-domain data, the F-1 score increases. With only 30% of the in-domain data, we get 89% F-1 score. Also, we see that pre-training and combined-training have similar performances.

For Figure 3, the out-domain model achieves a much lower F-1 score on the in-domain data. This shows that the transfer is not symmetric. This could be due to the PNAME slot, which has no similar slots in the shopping domain. There is also a difference between the performance curve of pre-training and combined-training. This indicates that fine-tuning a pre-trained model is harder than combined training when patterns are not covered by the out-domain data.

6 Conclusions and Future Work

We proposed a new type of slot-specific user intents, ***update intents*** (UI's), that are directly related to the type of update a user intends for a slot. The UI's address user intents containing implicit denials, numeric updates and preferences for slot-values, which are not handled by the currently used dialog acts. We presented a sequence labeling model for classifying UI's. We also proposed a method to transfer learning of UI's across domains by delexicalizing slot-values with their slot types. For that, we defined three generic slot types. Experimental results showed strong performance for UI classification and promising results for the domain independence experiments. In the future, we plan to explore mechanisms to utilize the UI's in belief tracking.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. 2012. Classifying user messages for managing web forum data. In *Proceedings of the 15th International Workshop on the Web and Databases*. pages 13–18.

Prakhar Biyani, Cornelia Caragea, and Prasenjit Mitra. 2013. Predicting subjectivity orientation of online forum threads. In *Computational Linguistics and Intelligent Text Processing*, Springer, pages 109–120.

Prakhar Biyani, Cornelia Caragea, Amit Singh, and Prasenjit Mitra. 2012. I want what i need!: analyzing subjectivity of online forum threads. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pages 2495–2498.

Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. 2016. " 8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *AAAI*. pages 94–100.

Y-Lan Boureau, Antoine Bordes, and Julien Perez. 2017. Dialog state tracking challenge 6 end-to-end goal-oriented dialog track. Technical report, Tech. Rep.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2013. An empirical investigation of sparse log-linear models for improved dialogue act classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pages 8317–8321.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

William W Cohen, Vitor R Carvalho, and Tom M Mitchell. 2004. Learning to classify email into"speech acts". In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 43–48.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Dilek Hakkani-Tür, Gokhan Tur, Larry Heck, Ashley Fidler, and Asli Celikyilmaz. 2012. A discriminative classification-based approach to information state updates for a multi-domain dialog system. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative spoken language understanding using word confusion networks. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, pages 176–181.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 324–329.

Matthew Henderson, Blaise Thomson, and Steve J Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL Conference*. pages 292–299.

G Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e* .

Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R Hershey. 2015. Context sensitive spoken language understanding using role dependent lstm layers. In *Machine Learning for SLU Interaction NIPS 2015 Workshop*.

Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R Hershey. 2016. Context-sensitive and role-dependent spoken language understanding using bidirectional and attention lstms. In *INTERSPEECH*. pages 3236–3240.

Seokhwan Kim and Rafael E. Banchs. 2014. Sequential labeling for tracking dynamic dialog states. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*. pages 332–336.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. 2016. The fifth dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, pages 511–517.

Seokhwan Kim, Luis Fernando DHaro, Rafael E Banchs, Jason Williams, and Matthew Henderson. 2015. Dialog state tracking challenge 4.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010. Tagging and linking web forum posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 192–202.

Sungjin Lee and Amanda Stent. 2016. Task lineages: Dialog state tracking for flexible interaction. In *SIGDIAL Conference*. pages 11–21.

François Mairesse, Milica Gasic, Filip Jurcícek, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2009. Spoken language understanding from unaligned data using discriminative classification models. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, pages 4749–4752.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190* .

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*. pages 1777–1788.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Hang Ren, Weiqun Xu, Yan Zhang, and Yonghong Yan. 2013. Dialog state tracking using conditional random fields. In *Proceedings of the SIGDIAL 2013 Conference*. pages 457–461.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. Technical report, INTERNATIONAL COMPUTER SCIENCE INST BERKELEY CA.

Takashi Ushio, Hongjie Shi, Mitsuru Endo, Katsuyoshi Yamagami, and Noriaki Horii. 2016. Recurrent convolutional neural networks for structured speech act tagging. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, pages 518–524.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. volume 1, pages 438–449.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan W Black. 2013. The dialog state tracking challenge. In *SIGDIAL Conference*. pages 404–413.

Steve Young. 2007. Cued standard dialogue acts. *Report, Cambridge University Engineering Department, 14th October* .