SEMANTIC PARSER ENHANCEMENT FOR DIALOGUE DOMAIN EXTENSION WITH LITTLE DATA

Su Zhu *Lu Chen* *Kai Sun* *Da Zheng* *Kai Yu*

Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{paul2204, chenlusz, accreator, yums, kai.yu}@sjtu.edu.cn

ABSTRACT

Statistical semantic parser trained on sufficient in-domain data has shown robustness to speech recognition errors in end-to-end spoken dialogue systems. However, when the dialogue domain is extended, due to the introduction of new semantic slots, values and unknown speech pattern, the parsing performance may significantly degrade. Effective re-training of statistical semantic parser is therefore important. This paper describes a novel semantic parser enhancement approach for domain extension with very little new data. It employs automatic pseudo-data generation for parser re-training and domain independent rescoring to further improve parsing performance. The approach was evaluated on the DSTC3 (the third Dialog State Tracking Challenge) data corpus. Experiments showed that the proposed approach can yield consistent and significant improvements across all metrics of semantic parsing and dialog state tracking.

Index Terms— Spoken language understanding, Dialog state tracking, Domain adaptation, Semantic parser enhancement

1. INTRODUCTION

Statistical approaches of spoken language understanding (SLU) trained on sufficient in-domain data have shown to be robust to errors of automatic speech recognition (ASR) [1, 2, 3], especially in end-to-end spoken dialogue systems. Dialog state tracking is a process of estimating a distribution over all possible dialogue states in statistical spoken dialogue system [4]. Recently, in dialogue state tracking, statistical semantic parser has also helped improvements of state tracking, given sufficient labelled data [5, 6].

But when the dialogue domain is extended, the performance of a semantic parser might reduce as a result of the introduction of new semantic slots, values and unknown speech pattern (or ASR hypothesis pattern). The ability for SLU to cope well with the expanded domains and limited training data is quite attractive to the deployment of commercial dialogue systems. Unsupervised training has been shown to be helpful in expanding domains of SLU system [7]. But in this paper, semantic parser enhancement methods are adopted for dialogue domain extended with little labelled example data.

The Dialog State Tracking Challenge (DSTC) provides a first common testbed in a standard format, along with a suite of evaluation metrics for dialogue state tracking. The third Dialog State Tracking Challenge (DSTC3) [8] extended dialogue domain in contrast to the Second Dialog State Tracking Challenge (DSTC2) corpus [9]. There is only a little labelled data (*seed data*) for the extended dialogue domain. So, it is necessary to perform semantic parser enhancement using the *seed data*, i.e. example dialogues of the *extended domain*, and sufficient dialogue data of the *original domain*.

This paper proposes two approaches for SLU enhancement in dialog state tracking. First, ASR hypotheses simulation is used for re-training semantic parsers for the extended dialogue domain. ASR hypotheses simulation is implemented on word level to generate additional training data adapted from the original to the extended domain. These new sentences are generated for the new slots and values, which contains new sentence patterns and new text contexts of the values. Unaligned data in SLU is useful since ASR errors would often make troubles in word-semantic alignment. In this paper, the method of ASR hypotheses simulation is specific to the SLU system with unaligned training data [3] which requires full text sentences for training.

In addition to parser re-training, this paper adopts a novel method to make use of system act for dialogue domain extension. System act is the semantics that the machine feeds back to the user and has good relationship with what the user would say next. Henderson et al. trained semantic tuple classifiers by concatenating the text feature and the last system act feature [2]. But in the extended dialogue domain, it is more convenient to use the system act features independent of the dialogue domain. In this paper, the last system act features independent of dialogue domain are exploited to train a SLU rescoring system in the original domain and applied to the

This work was supported by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning and the China NSFC project No. 61222208.

semantic parser enhancement in the extended domain.

This paper is organized as follows. In section 2, as two methods of semantic parser enhancement, ASR hypotheses simulation and SLU rescoring are described. Section 3 briefly describes the corpus of the DSTC3 domain and the evaluation metrics, and presents the results on semantic parsing and dialogue state tracking. Finally, section 4 concludes the paper.

2. SEMANTIC PARSER ENHANCEMENT

This section explains how to enhance semantic parser for dialogue domain extension. In this task, there is a lot of labelled data in the original domain. The original domain is smaller but related to the extended domain. The extended semantic ontology and a small set of seed data are known in advance.

In this paper, the semantics of an utterance from user is represented in functor form called *dialogue act* [10] consisting of an *acttype* and a list of *slot-value* pairs, for example:

request(name,food=chinese)

where "request" is the dialogue *acttype*,"name" is a *slot* requested, "food=chinese" is a *slot-value* pair which provides some information to the system and "food" is informed. They are all called *semantic items*.

In semantic parser, slots could be divided into *enumerable* and *non-enumerable* slots [3]. For example, an enumerable slot "pricerange" in DSTC3 is only associated with "expensive", "free", "cheap" and "moderate", whereas non-enumerable slots can take any number of possible values theoretically (e.g., "foodtype").

2.1. ASR Hypotheses Simulation and Parser Re-Training

In general, the semantic data of different dialogue domains can be classified as:

- **Domain-independent**: Data samples independent of any specific domain, e.g. users say hello or thank you.
- **Domain transferable**: Data samples appearing in both the original and the extended domain.
- **Non-transferable & domain constrained**: Data samples specific for one domain, which couldn't be transferred to another domain.

ASR hypotheses simulation focuses on generation of the third part data samples. With these transferred and generated data, the semantic parser for the extended domain is easy to be built. The whole process is illustrated in Fig. 1.

2.1.1. Pattern generation

Typically, data pair for SLU parser is denoted as:

text : *I need moderately priced swedish food*

act : *inform(pricerange=moderate,food=swedish)*

where the text comes from the user input (transcription or ASR hypotheses) and act is the corresponding semantics. The

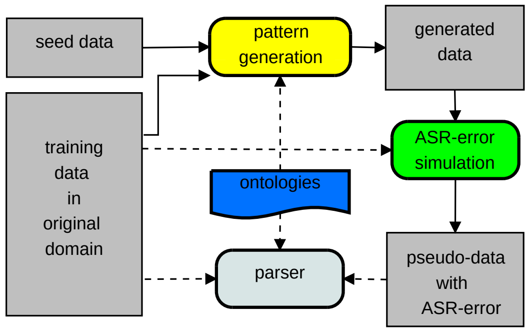


Fig. 1. *The architecture of ASR hypotheses simulation.*

value set for the non-enumerable slot is usually very large. Hence it is useful to use a specific class label to replace the value. After replacement, the training sample become a sample pattern which contains the text pattern and corresponding act pattern. Following the example above, the text pattern is "I need moderately priced [food] food", and the act pattern is "inform(pricerange=moderate,food=[food])" where "[food]" is the class label.

When the original domain is extended, new slots, new values for original slot and a few new text patterns would appear in the extended dialogue domain. So it is necessary to generate enough training sample patterns related to the new slot-value pairs and new text patterns to train the semantic parser in the extended domain. It is effective to do the pattern simulation separately for new slots, new value sets and new text patterns.

Before pattern generation, it is necessary to discuss about the word sequence alignment between the text pattern and the specific slot or value. In this paper, it is assumed that there is a word sequence corresponding to the slot or value in the text pattern, which is called spoken pattern. For example, "pricerange" and "price range" are the spoken patterns of slot *pricerange*, "in the south part of town" is one of the spoken patterns of the value "south" which belongs to the slot *area*. In this paper, the spoken patterns for each slot and slot-value pair are manually summarized as handcraft rules from the labelled data. In the furture work, it is possible to map the slot and value to a sequence of words in an absolutely automatic process, especially for a large dialogue domain.

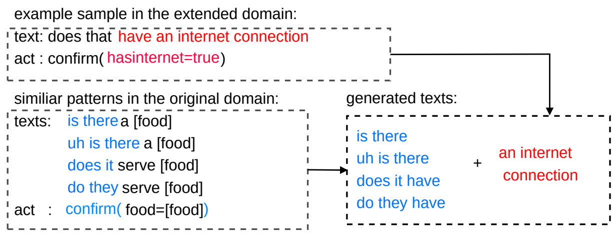
For a new slot in the extended domain, it is necessary to generate new training sample patterns and insert possible values with the old training sample patterns in the original domain. For all training sample patterns (seed text patterns and seed act patterns) in the seed data of the extended domain consisting of the new slot $\hat{s}$:

- **New requested slot**: If $\hat{s}$ is a requested slot in the seed act patterns, we choose a requestable slot s randomly in the original domain to get each training sample pattern with text pattern t and act pattern a consisting of the requested s. Then, we replace the spoken pattern of s

with that of $\hat{s}$ in t to get a new text pattern $\hat{t}$, as well as replacing s with $\hat{s}$ in a to get a new act pattern $\hat{a}$. Now, a new training sample pattern $\hat{t}$ with $\hat{a}$ is generated for requesting $\hat{s}$.

- **New informed slot**: If $\hat{s}$ is an informed slot in the seed act pattern a, we cut the slot-value pair of $\hat{s}$ from a to get the rest act pattern a_{rest}. Then it is needed to find out each training sample pattern (text pattern t and act pattern a') in the original domain where the a' is the same with a_{rest} or a' is the combination of a_{rest} and a slot-value pair s-v. Then, for each possible value $\hat{v}$ of $\hat{s}$, we combine t with the spoken pattern of $\hat{v}$ or replace the the spoken pattern of v with $\hat{v}$ in t to get $\hat{t}$, as well as combine a_{rest} with $\hat{s} = \hat{v}$ to get the corresponding act pattern $\hat{a}$. If $\hat{s}$ is a non-enumerable slot, it needs to do class replacing first and the possible value of $\hat{s}$ is only the class label.

As illustrated in Fig.2, the new training sample pattern is generated for a new slot in the extended domain.



example sample in the extended domain:
text: does that have an internet connection
act : confirm(hasinternet=true)
similiar patterns in the original domain:
texts: is there a [food]
uh is there a [food]
does it serve [food]
do they serve [food]
act : confirm(food=[food])
generated texts:
is there
uh is there
does it have
do they have
+
an internet connection

Fig. 2. *An example for training sample pattern generation, where 'have' is omitted when 'is' acts as a predicate in the sentence.*

When a new value $\hat{v}$ is added for the slot s while s exists in both the original domain and the extended domain, we do the generation similar as the above.

- **New value & transferable**: If s is a non-enumerable slot (e.g. "name","food"), there is no need to do generation because the values of s are replaced with the same label no matter in original domain or extended domain.
- **New value & extended domain specific**: If s is an enumerable slot (e.g. "pricerange","area"), we should find out each training sample pattern (text pattern t and act pattern a) in the original domain where a contains the slot-value pair s-v (v is different with $\hat{v}$). Then, we replace the spoken pattern of v with that of $\hat{v}$ in t to get new text pattern $\hat{t}$, as well as replacing v with $\hat{v}$ in a to get a new act pattern $\hat{a}$.

After pattern generation for new slots and new value sets from the training sample patterns P_o of the original domain, we get additional training sample patterns P_g for the extended domain. P_o is extracted from the training data D_o of the original domain.

Likewise as the example data, training sample patterns P_e from the extended domain seed data might contain the new patterns which couldn't be found in the original domain. So it is necessary to take the new patterns together for training the semantic parser. However P_e is a small set, it is necessary to expand it for all unseen possible values, and then generate new patterns from p_e ($p_e \in P_e$) consisting of an enumerable slot s with all the possible values of s. For example, the total number of all possible values of slot *area* is 15, but there are only 3 of them appearing in the seed data, whereas the other 12 values are used in expanding. After P_e is expanded, the set containing all generated training sample patterns for the extended domain are $P_g + P_e$. Finally, fill the class labels of $P_g + P_e$ with corresponding values of the extended domain randomly to get the complete data $D_g + D_e$. Thus D_o,D_g and D_e make up the training set of the semantic parser in the extended domain.

2.1.2. *ASR error simulation*

The semantic training data with ASR errors is known to provide robustness to the speech recognition errors so that it is necessary to involve ASR errors in parser re-training. As described in the previous subsection, the text pattern can be extracted from ASR hypotheses or transcription. There are two approaches to simulate ASR errors for the generated data samples for these two cases respectively.

Firstly, if the text patterns for generation are extracted from the ASR hypotheses in the original domain, the ASR errors are already involved in the generated data $D_g + D_e$. This simple approach is called *ASR error tied*.

Secondly, if the text patterns for generation are transcriptions, a more complex ASR error simulation method is applied to simulate ASR errors for $D_g + D_e$. This approach is called *word-level confusion*. In this method, slots and sentence patterns are separately dealt with. Sentence patterns are extracted from ASR results from the original domain. Patterns maintain the whole sentence except slot values are changed, as seen in the example below:

transcription	:	moderately priced swedish food
transcription pattern	:	[*pricerange*] priced [*food*] food
ASR output pattern	:	uh [*pricerange*] priced [*food*] code

In which [*pricerange*] and [*food*] are slot labels. Transcriptions with same pattern will be combined and their ASR results are stored as possible simulation results.

All possible slot values are preprocessed to generate possible error outcomes[11]. Those values are transformed on phoneme level and then put into a Finite State Transducer network to generate possible errors. All target words are from seed transcriptions. Example:

slot value	:	*thai*
confusion	:	*uhh thai, i, thai a, uh tie, uh thai, tie, thai*

When a new transcription is given, the method will generate simulated ASR results with seen ASR results of its transcription pattern and possible confusion of its slot values.

2.1.3. Training

Following the *semantic tuple classifiers* approach [2, 3], a binary classifier is trained for each *slot-value pair*, and predicts the presence of that slot-value pair in the utterance. Specially, when slots are requested, a binary classifier is also trained for each *request-slot* pair. Similarly, a multi-class classifier is estimated for all the dialogue act types. For classifications, we use Support Vector Machines (SVMs) by the LibSVM package [12] which can output probabilities. Finally, a decoding process of these semantic tuples is taken to produce dialogue acts which might be multiple. This decoding must keep to a simple set of rules, e.g. dialogue act type *request* must appear together with at least one unbounded slot.

The semantic parser is built by exploiting n-gram feature of user utterances. n is allowed to range from 1 to 3, i.e. uni-grams, bi-grams and tri-grams are counted to be features. Also, the n-gram features from the top hypothesis and the N-best list of ASR hypotheses can be used in the training of semantic parser respectively.

2.2. Domain-independent SLU Rescoring

As described in the introduction, system dialogue act is helpful in semantic parsing and has good relationship with what the user would say possibly next. However, system dialogue act may differ distinctly between two different dialogue domains. So it is not possible to train semantic tuple classifiers by concatenating user utterance feature and system dialogue act feature [2] in semantic parser enhancement.

In order to solve this problem, an approach using the system act feature independent of slot-value is adopted to rescore the probability of each independently semantic tuple classifier output. Three rescoring models for the semantic tuples are built: act type rescoring model (a multi-class classifier) for the *act type* semantic tuple, request-slot rescoring model (a binary classifier) for all *slot-value* semantic tuples and slot-value pair rescoring model (a binary classifier) for all *request-slot* semantic tuples.

The features used in rescoring are listed below:

1. **STC score**: The output probability of the original semantic tuple classifier (STC).
2. **System act type**: A feature for each possible system dialogue act type, giving the indication of whether each act type exists in the last system act.
3. **Acttype-slot**: A feature giving the indication of whether each (acttype, slot) pair exists in the last system act.
4. **Slot-value**: A feature giving the indication of whether each (slot, value) pair exists in the last system act.

Hence, there are only Fea.$\{1,2\}$ available for act type rescoring, Fea.$\{1,3\}$ for request-slot rescoring and Fea.$\{1,3,4\}$ for slot-value pair rescoring.

The rescoring models are trained on the original domain and transferred to be used in the extended domain. These classifiers are also trained by using SVMs and their output probabilities are used as new scores. In this paper, act types are not rescored for the reason that there are too many act types to be well estimated.

3. EXPERIMENTS

3.1. The Third Dialog State Tracking Challenge (DSTC3)

DSTC3 is a task of developing robust and accurate state trackers to work in an extended dialogue domain. As the original dialogue domain, the DSTC2 dataset is in the domain of finding a restaurant in Cambridge. The corpus consists of 3235 dialogues with 25501 user utterances totally.

The DSTC3 corpus consists of dialogues in a tourist information system which could recommend pubs, coffee shops and restaurants in Cambridge. The dialogue domain of DSTC3 is more complex than DSTC2. The entire datasets from DSTC2 are available in DSTC3, as well as a small set of labelled data in DSTC3. Full details of the corpus are given in the challenge handbook [9]. But in this paper, the DSTC3 corpus was split into a train and test set again for development, and their labels are released after the DSTC3 challenging. These datasets are:

- **Seed**: 11 labelled example dialogues, the *seed data*.
- **Real-train**: the seed data and other 1144 labelled dialogues randomly chosen from the DSTC3 corpus.
- **Real-test**: 1120 dialogues used for evaluation which are the rest of DSTC3 corpus besides Real-train.

3.2. Semantic Parsing Experiments and Results

As an important metric to estimate the quality of output of a semantic parser, **F-score** is the harmonic mean of the **precision** and **recall** of semantic items in the top semantic hypotheses. For statistical dialogue systems, the Item Cross Entropy (**ICE**) between the N-best semantic hypothses and the semantic label assesses the overall quality of the semantic items distribution, and is shown to give a consistent performance ranking for both the confidence scores and the overall correctness of the semantic parser [13]. The lower ICE indicates the better performance.

In DSTC3, the SLU outputs are available and given by an in-domain semantic parser from the organizers. The SLU outputs provided by DSTC3 is called **Org-train**. For guaranteeing the comparability, another in-domain semantic parser is implemented with the semantic tuple classifiers approach trained on **Real-train** set by exploiting n-grams feature. As a baseline system, **Seed-train** is a more simple approach than ASR hypotheses simulation in SLU parser domain extension, with only all DSTC2 data and the DSTC3 seed data after expanding in section 2.1. The proposed approaches in this paper are referred to as **Simul-train**, where the ASR error tied method is called **Err-tied** and the word-level confusion method is called **Err-gen**.

Train data	Error simulation	F-score	ICE
Org-train		0.820	1.741
Real-train	—	0.863	1.060
Seed-train	—	0.808	1.734
Simul-train	Err-tied	0.831	1.494
	Err-gen	**0.833**	**1.483**

Table 1. *Performances of different semantic parsers evaluated on the Real-test set, as well as trained on the n-grams feature of ASR 1-best hypothesis. The Simul-train data consists of all DSTC2 data and the generated data in ASR hypotheses simulation.*

In Table 1, Real-train shows the ideal parser performance with the best metirc values that the parser is trained with sufficient labelled data of the extended domain. While the bold indicates the best score of each metric among different semantic parsers, except for Real-train system. Err-tied shows significant improvement in contrast to Org-train and Seed-train, as the p-values of significance testing in each metric are all nearly 0.01 (smaller than 0.05). Err-gen improves all metrics slightly to be better than Err-tied result. Although Err-gen has no significant difference with Err-tied, it is still used as a better system as it shows constantly better in each metric.

Train Data	n-best	Prec.	Recall	F-score	ICE
Real-train	1	0.914	0.818	0.863	1.060
	10	0.911	0.829	0.868	1.090
Err-gen	1	0.870	0.799	0.833	1.483
	10	0.855	0.798	0.826	1.759

Table 2. *Performances of different enhanced parsers based on ASR n-best list on the Real-test set. Err-gen means the data from ASR hypotheses simulation with the ASR error simulation method of word-level confusion.*

The ASR N-best list is known to provide stronger features than the ASR 1-best hypothesis for semantic parser [2]. As Table 2 shows, N-best list features increase the F-score but increase the ICE slightly in the parsers (rows 1 and 2) trained on Real-train. But the semantic parsers trained on Err-gen (rows 3 and 4) don't improve the F-score or ICE either by using N-best list. The ASR error simulation method adopted here didn't generate errors well on N-best list.

Table 3 shows that the method of SLU rescoring with the last system act feature improves each SLU metric slightly. Although the SLU rescoring doesn't get significant improvement, it is used in our best system.

The previous experiments only concern a single parser. Since **Org-train** is provided by the DSTC3 organizer, it is interesting to investigate whether parser combination can help or not. Table 4 shows Err-gen with rescoring gets significant improvement in the ICE , after being combined with Org-train

Parser	Prec.	Recall	F-score	ICE
Real-train	0.914	0.818	0.863	1.060
+ system act	0.926	0.822	0.871	0.970
Err-gen	0.870	0.799	0.833	1.483
+ rescore	0.871	0.801	0.834	1.483

Table 3. *Performances of different parsers trained with ASR 1-best output, on the Real-test set. The parser trained on Real-train uses the last system act by concatenating the system act feature. The parser trained on Err-gen data uses the last system act by rescoring.*

by score averaging. Because of that the Org-train helps semantic parsing in ASR error patterns which didn't exist in ASR hypotheses simulation. The combined parser of Err-gen with rescoring and Org-train is named as the **refined** parser which will be used in the next subsection.

Parser	Prec.	Recall	F-score	ICE
Org-train	0.850	0.792	0.820	1.741
Err-gen + rescore	**0.871**	**0.801**	**0.834**	1.483
+ Org-train (refined)	0.870	0.796	0.831	**1.265**

Table 4. *Performances of semantic parser combination on the Real-test set.* **refined** *is used to refer to the combined parser.*

3.3. Dialog State Tracking Challenge Results

This subsection shows the performance of semantic parser enhancement in the DSTC3 challenge, assuming that the DSTC3 corpus is not labelled besides the seed data. The evaluation dataset contains Real-train (excepts for the seed data) and Real-test set in section 3.1.

In dialog state tracking, the hidden dialogue states consist of three components: user's **joint goals**, search **method**, **request slots**. The user's joint goals specify the user's search constraints. In this paper, the data and evaluation metrics we used come from DSTC2&3 [9]. Acc denotes the accuracy of the top-scored hypothesis at each turn, and L2 denotes the squared $l2$ norm between the estimated score distribution and correct distribution. The dialog state tracker used in this paper is a novel rule-based model which is depicted as a special kind of polynomial functions satisfying some linear constraints, and outperformed all four baselines in DSTC3 [14].

Table 5 shows the SLU results on the DSTC3 challenge evaluation. The *submitted* [1] semantic parser is the one which we used in the DSTC3 challenge. The *refined* is the best

[1]The SLU output comes from the combination of three parsers: the Err-tied method on the train set of DSTC2, Err-tied method on all datasets of DSTC2 and the Org-train. This combination overfits the DSTC2 domain to get higher F-score as the DSTC3 specific slots appear not often. But these specific slots will stay in the latent dialog states to impact the tracker.

semantic parser which we found after the challenge and described in the previous subsection.

Parser	Prec.	Recall	F-score	ICE
Org-train	0.852	0.797	0.824	1.719
Submitted	**0.881**	**0.811**	**0.845**	1.235
Refined	0.870	0.801	0.834	**1.229**

Table 5. *SLU results of DSTC3 challenge evaluation.*

Table 6 shows the tracker results of DSTC3 evaluation on different semantic outputs. The *submitted* is reported as 'team5/entry0' in the DSTC3 challenge, which outperforms all four baselines. The tracker results on the *refined* SLU output which we provided after the challenge show significant improvement in joint goals. But it decreases the request-slots accuracy in contrast to the *submitted*, as the combination doesn't alway provide gains on each metric.

Parser	Joint goal		Method		Request	
	Acc.	L2	Acc.	L2	Acc.	L2
Org-train	0.582	0.690	0.967	**0.062**	0.909	0.161
Submitted	0.610	0.556	**0.968**	0.091	**0.945**	**0.090**
Refined	**0.626**	**0.551**	**0.968**	0.080	0.924	0.098

Table 6. *Tracker results of DSTC3 challenge evaluation.*

4. CONCLUSIONS

This paper describes two novel approaches of semantic parser enhancement for dialogue domain extension with little labelled data of the extended domain. The ASR hypotheses simulation provided very good performance in terms of both the F-score and ICE metric. The domain-independent SLU rescoring provided an improvement again, eventhough only slightly. Moreover, for DSTC3 challenge, the SLU output combination improved the quality of SLU probability distribution. The SLU enhancement also helps in dialog state tracking.

Future work should include the automatic extraction of spoken patterns for the corresponding slot and value. In future, we also plan to investigate how to simulate ASR errors on word level, especially for unseen words in the previous data. In general, the N-best list feature should provide better performance on SLU which wasn't achieved in this paper. Furthermore the scoring for the pseudo N-best ASR hypotheses will be explored in future work.

5. REFERENCES

[1] Christian Raymond and Giuseppe Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *INTERSPEECH*, 2007, pp. 1605–1608.

[2] Matthew Henderson, Milica Gasic, Blaise Thomson, and et al, "Discriminative spoken language understanding using word confusion networks," in *SLT*, 2012, pp. 176–181.

[3] François Mairesse, Milica Gasic, Filip Jurcícek, and et al, "Spoken language understanding from unaligned data using discriminative classification models," in *Proceedings of ICASSP*, 2009.

[4] Matthew Henderson, Blaise Thomson, and Jason Williams, "The second dialog state tracking challenge," in *Proceedings of the SIGdial 2014 Conference, Baltimore, USA, June*, 2014.

[5] Kai Sun, Lu Chen, Su Zhu, and Kai Yu, "The SJTU system for dialog state tracking challenge 2," in *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2014, p. 318.

[6] Jason D Williams and Jason Williams, "Web-style ranking and SLU combination for dialog state tracking," in *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2014, p. 282.

[7] Larry Heck and Dilek Hakkani-Tur, "Exploiting the semantic web for unsupervised spoken language understanding," in *SLT*, 2012, pp. 228–233.

[8] Matthew Henderson, Blaise Thomson, and Jason Williams, "Announcing the Third Dialog State Tracking Challenge," 2014.

[9] Matthew Henderson, Blaise Thomson, and Jason Williams, "Dialog State Tracking Challenge 2 & 3," 2013.

[10] Steve Young, "CUED standard dialogue acts," *Report, Cambridge University Engineering Department, 14th October*, 2007.

[11] Jost Schatzmann, Blaise Thomson, and Steve Young, "Error simulation for training statistical dialogue systems," in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007, pp. 526–531.

[12] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2011.

[13] Blaise Thomson, Kai Yu, Milica Gasic, and et al, "Evaluating semantic-level confidence scores with multiple hypotheses," in *INTERSPEECH*, 2008, pp. 1153–1156.

[14] Kai Sun, Lu Chen, Su Zhu, and Kai Yu, "A generalized rule based tracker for dialogue state tracking," in *submitted to IEEE SLT 2014*, 2014.