Concept Transfer Learning for Adaptive Language Understanding

Su Zhu and **Kai Yu**
Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{paul2204,kai.yu}@sjtu.edu.cn

Abstract

Semantic transfer is an important problem of the language understanding (LU), which is about how the recognition pattern of a semantic concept benefits other associated concepts. In this paper, we propose a new semantic representation based on combinatory concepts. Semantic slot is represented as a composition of different atomic concepts in different semantic dimensions. Specifically, we propose the concept transfer learning methods for extending combinatory concepts in LU. The concept transfer learning makes use of the common ground of combinatory concepts shown in the literal description. Our methods are applied to two adaptive LU problems: semantic slot refinement and domain adaptation, and respectively evaluated on two benchmark LU datasets: ATIS and DSTC 2&3. The experiment results show that the concept transfer learning is very efficient for semantic slot refinement and domain adaptation in the LU.

1 Introduction

The language understanding (LU) module is a key component of the dialogue system (DS), parsing the users' utterances into the corresponding semantic concepts. For example, the utterance *"Show me flights from Boston to New York"* can be parsed into *(fromloc.city_name=Boston, toloc.city_name=New York)* (Pieraccini et al., 1992). Typically, the LU problem is regarded as a slot filling task. With sufficient in-domain data and deep learning models (e.g. recurrent neural networks), the statistical methods have achieved high performance in the slot filling task recently (Kurata et al., 2016; Vu, 2016; Liu and Lane, 2016).

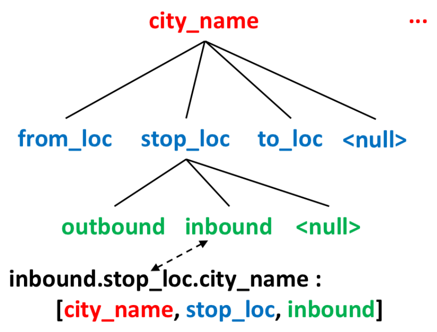


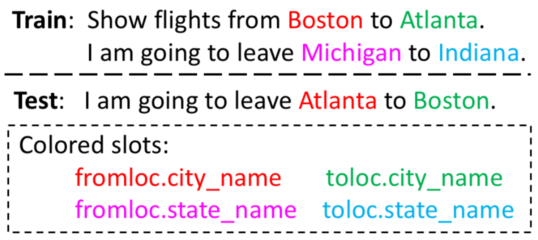
Figure 1: An example of atomic concept tree. The root of this tree is *city_name*. There may be other trees with roots, like *airport_name*, *date_time* etc. The example shows the semantic slot *inbound.stop_loc.city_name* can be represented as a compositional vector of atomic concepts.

However it is very hard to get sufficient in-domain data for training LU model (Tur et al., 2010), especially when the semantic slot extends or dialogue domain changes. In recent years, more and more applications of DS have been released along with the development of the mobile internet, e.g. Apple Siri, Amazon Alexa, Google Home, Microsoft Cortana etc. The ability for LU approaches to cope with changing domains and limited training data is of particular interest for the deployment of commercial dialogue system.

Ontology construction is important and sophisticated for LU, defining the concepts and their relations. In conventional LU, ontology is always defined as a set of semantic slots which are in a plain level without hierarchical structures. In this paper, we represent the semantic slots with a description structure based on combinatory concepts, as shown in Figure 1. Each semantic slot is composed of different atomic concepts in different semantic dimensions. The pattern learning in atomic concept level will helps share knowledge from the data.

In this paper, we try to establish a semantic representation standard that describes the semantic slots based on combinatory concepts. The semantic slots and their relations are represented by trees of atomic concepts (Figure 1; it can be expanded to a directed graph in future work). Each branch of the tree denotes a semantic slot, e.g. the slot of *"date_of_birth"* can be defined as a branch of atoms [*"date"*, *"birth"*].

The atomic concepts can be classified into two categories, one is value-aware (domain-independent) and the other is context-aware (domain-specific) (e.g. *"city_name"* and *"date"* are value-aware, *"stop_loc"*, *"inbound"* and *"birth"* are context-aware). Modelling on the atomic concepts but not the joint semantic slot helps find out the linguistic patterns of related slots by semantic sharing, and even decrease the required amount of data. For example, the training and test sets are unmatched in Figure 2, whereas the patterns of atomic concepts (e.g. *"fromloc"*) can be shared.



Train: Show flights from Boston to Atlanta.
I am going to leave Michigan to Indiana.

Test: I am going to leave Atlanta to Boston.

Colored slots:
fromloc.city_name toloc.city_name
fromloc.state_name toloc.state_name

Figure 2: An example of unmatched LU datasets.

In this paper, we investigated the combinatory concepts preliminarily based on atomic concept trees for semantic representation. We proposed to learn slot filling model on the granularity of atomic concepts by considering these atoms independent and dependent respectively. Our methods are applied to the semantic slot refinement and domain adaptation problem on the datasets of ATIS (Hemphill et al., 1995) and DSTC 2&3 (Henderson et al., 2013) respectively, about the extending of combinatory concepts. Our main contributions can be summarized as:

- We propose the novel atomic concept trees for semantic representation in the LU, which refines the plain semantic slots to be hierarchical structures.
- The slot filling based on concept transfer learning is a very efficient way for solving the extending of combinatory concepts in the LU, as shown in the experimental results on the ATIS and DSTC 2&3.
- Our experimental results also show the concept transfer learning method achieves the state-of-the-art performance (F_1-score 96.08%) in the ATIS task, only using the lexicon features.

2 Related Work

Slot Filling in LU A grammar induction method was presented by Zettlemoyer and Collins (2007), learning a probabilistic combinatory categorial grammar (PCCG) from logical-form annotations. As a rule-based method, the PCCG's work (Zettlemoyer and Collins, 2007) is close to a hierarchical concepts structure in grammar generation and combination. But this rule-based method does not have high generalization capability for atomic concept sharing, and depends on a well-defined lexicon set initially.

Recent research about statistical slot filling in LU has been focused on RNN (recurrent neural network) and its extensions. At first, Yao et al. (2013) used RNN outperforming CRF (conditional random field) on the ATIS dataset. Mesnil et al. (2013) tried bidirectional and hybrid RNN to investigate using RNN for slot filling. Yao et al. (2014) introduced LSTM (long-short memory networks) and deep LSTM architecture for this task and obtained a marginal improvement over RNN. Vu et al. (2016) proposed to use the ranking loss to train a bidirectional RNN. Peng et al. (2015) proposed RNN-EM which used an external memory architecture to improve the memory capability of RNN. Inspired by the encoder-decoder nerual network (Bahdanau et al., 2014), Kurata et al. (2016) proposed an encoder-labeler model for slot filling, and Liu and Lane (2016); Zhu and Yu (2017) adapted the attention model to the slot filling task. However, these work only predicted a joint semantic slot (one-hot vector), not a structure of atomic concepts.

Domain Adaptation in LU For the domain adaptation in LU, Zhu et al. (2014) proposed to generate spoken language surface forms with patterns of the source domain and the ontology of the target domain, but not automatically. With regard to the unsupervised LU, Heck and Hakkani-Tur (2012) exploited the structure of semantic knowledge graphs from the web to create natural language surface forms of entity-relation-entity portions of knowledge graphs. About the open-domain LU, Chen and Rudnicky (2014) used structured knowledge resources (e.g. Freebase, Wikipedia, FrameNet) to induce types of slots for generating semantic seeds, and enriched the semantics of spoken queries with neural word embeddings for a new domain. However, the performance of the unsupervised LU is not satisfied. For zero-shot learning in LU, Ferreira et al. (2015); Yazdani and Henderson (2015) proposed a model to calculate the similarity between input sentence and any possible semantic items. In this paper, we focus on the extending of combinatory concepts, not only domain adaptation.

3 Atomic Concept Trees

Although the semantic representation is one of the most important problems of LU, there is no unified surface form for the domain ontology. Even for the same semantic slot, the names of this slot maybe different. For example, the city where the flight departs may be called as *"from_city"*, *"depart_city"* or *"fromloc.city_name"*. The ontology definitions from different groups maybe similar but not consistent. It is not optimal for data reuse. Meanwhile the semantic slots defined in traditional LU system are in a plain level, without a structure to indicate their relations.

To solve this problem, we propose to use atomic concepts to represent the semantic slots. The atomic concept is exploited to define the semantic unit and represent the semantic slots as the atomic concept trees (Figure 1 is an example). The semantic slot composed of these combinatory concepts can keep a unified semantic representation and flexibly extend semantic knowledge.

We make a discipline for atomic concept construction manually. For a given vocabulary C of the atomic concepts, a semantic slot s can be represented by a branch $[c_1, c_2, ..., c_k]$ of the tree, where $c_i \in C$ is in the i-th semantic dimension and k is the tree-depth. In particular, a *"null"* atomic concept is introduced for each dimension. As illustrated in Table 1, it is an example of slot representation in the ATIS task. To avoid a scratch concept branch, we make a constraint:

$$C_i \cap C_j = \{null\}, 1 \leq i \neq j \leq k$$

where C_i $(1 \leq i \leq k)$ denotes all possible atomic concepts which could exist in dimension i. For example, if we define two slots [*"city_name", "fromloc"*] and [*"state_name", "toloc"*], then the city where the flight arrives should be defined as [*"city_name", "toloc"*] but not [*"toloc", "city_name"*]. The concept branch is ordered.

The principle for defining slot as a concept branch is : lower dimension less context-aware. For example, *"city_name"* and *"time"* depend on rare context (value-aware). They should be in the first dimension. *"fromloc"* depends on the context like a pattern of *"a flight leaves [city_name]"*, which should be in the second dimension.

slot	**concept branch**
city_name	[city_name, null]
fromloc.city_name	[city_name, fromloc]
arrive_time.time	[time, arrive_time]
airline_name	[airline_name, null]

Table 1: An example of slot representation by atomic concepts in the ATIS task.

The combinatory concepts dependent on atomic coupling can share the same units with others. It is flexible for transfer learning based on semantic units, and even finding a new semantic slot.

4 Concept Transfer Learning

4.1 Atomic-Concepts Based Slot Filling

The slot filling is typically considered as a sequence labeling problem. In this paper, we only consider the sequence-labeling based slot filling task. The input (word) sequence is denoted by $\mathbf{w} = (w_1, w_2, ..., w_n)$, and the output (slot tag) sequence is denoted by $\mathbf{s} = (s_1, s_2, ..., s_n)$. Since a slot may be mapped to several continuous words, we follow the popular in/out/begin (IOB) representation (e.g. an example in Figure 3).

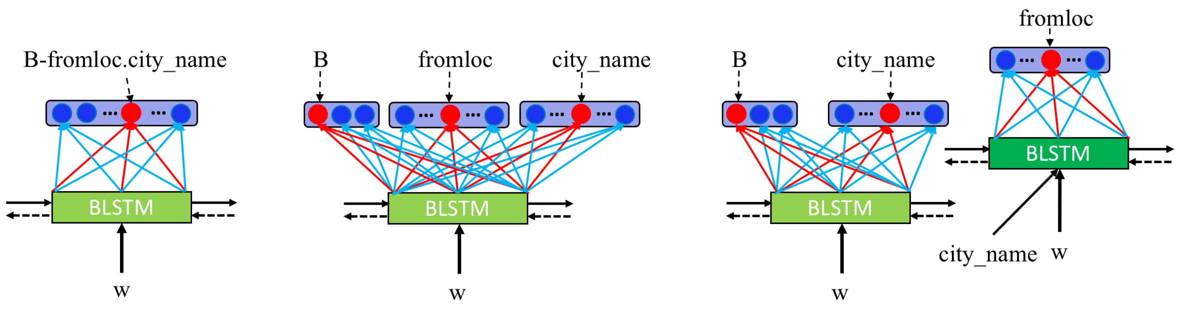
Words	show	flights	from	Boston	to	New	York	today
Slots	O	O	O	B-FromCity	O	B-ToCity	I-ToCity	B-Date

Figure 3: An example of annotation for slot filling.

The typical slot filling is to predict the slot-tag sequence given a word sequence. It is named as **joint slot filling** (**JS**), since the slot is regarded as a single and united symbol, like *"B-fromloc.city_name"* in Figure 4(a).

4.1.1 Bidirectional LSTM RNN

In this paper, the popular RNN is used to model the sequence labeling problem. Traditional RNN



(a) Joint slot filling (JS) (b) Atomic concept independent (AC) (c) Atomic concept dependent (ACD)

Figure 4: Examples for atomic-concepts based slot filling in the ATIS task.

has the problem of vanishing or exploding gradients, which means the long-term dependencies can hardly be estimated. LSTM is designed to alleviate this problem as proposed in (Graves, 2012). In sequence labeling task, we have access to both the past and future features for a given time. A bidirectional LSTM RNN (BLSTM) can be exploited to capture the past features (via forward states) and future features (via backward states) for a specific time frame, as shown in Figure 5.

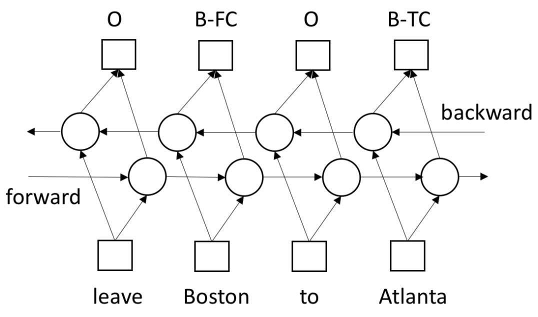


Figure 5: An example of BLSTM.

Therefore, the slot filling can be formulated as

$$p(\mathbf{s}|\mathbf{w}) = \prod_{i=1}^{n} p(s_i|\mathbf{w})$$

At the output layer, the *softmax* function is used to estimate the probability distribution over all the possible semantic slots in a specific time frame.

4.1.2 Atomic concept independent

The slot is indicated as an atomic concept branch based on the semantic representation of atomic concept trees. The tree structure is simplified to multi-level structure in modelling for preliminary investigation.

The slot filling can be transferred to a multi-task sequence labeling problem, regarding these atomic concepts **independently** (**AC**). Each task is to predict one of the atomic concepts. Thus, the slot filling problem can be formulated as

$$p(\mathbf{s}|\mathbf{w}) = \prod_{i=1}^{n} \prod_{j=1}^{k} p(c_{ij}|\mathbf{w})$$

where the semantic slot s_i is represented by an atomic concept branch $[c_{i1}, c_{i2}, ..., c_{ik}]$. As illustrated in the Figure 4(b), the semantic slot *fromloc.city_name* can be represented by $[city\_name, fromloc]$. The prediction of IOB is especially regarded as another task. All tasks share the parameters except for the output layers.

4.1.3 Atomic concept dependent

The atomic concepts can also be regarded **dependently** (**ACD**) that the atomic concept prediction depends on the former predicted results. The slot filling problem can be formulated as

$$p(\mathbf{s}|\mathbf{w}) = \prod_{i=1}^{n} [p(c_{i1}|\mathbf{w}) \prod_{j=2}^{k} p(c_{ij}|\mathbf{w}, c_{i,1:j-1})]$$

where $c_{i,1:j-1} = (c_{i,1}, ..., c_{i,j-1})$ is the predicted result of the former atomic concepts of slot tag s_i.

It is a structured multi-task learning. In this paper, we make some simplifications on the concept dependence. We predict the atomic concept only depending on the last atomic concept, as shown in the Figure 4(c).

In decoding stage, we combine the predicted atomic concepts with probability multiplication. The evaluation is performed on the top-best hypothesis. Although the atomic-concepts based slot filling may predict a slot unseen in the atomic concept trees. We didn't perform any post-processing but considered the unseen slot as a wrong prediction.

4.2 Combinatory Concepts Extending

Combinatory concepts expands by involving several new semantic slots into the source ontology to obtain the target ontology. For the combinatory concepts extending in the LU, we follow the typical adaptation of neural network (NN) based model, which trains a NN model with data from the source data S and fine-tunes the model with data from the target T. The training procedure is:

1. Initializing the NN based model m randomly.
2. Updating m with the data from S.
3. Adjusting the architecture of m: $adjust\_NN\_arch(m, S, T)$.
4. Updating m with the data from T.

For the joint slot filling (e.g. Figure 4(a)), the procedure of $adjust\_NN\_arch(m, S, T)$ is extending the output size and initializing the extended weights of the output layer randomly. For the atomic-concepts based slot filling (e.g. Figure 4(b,c)), the procedure of $adjust\_NN\_arch(m, S, T)$ is extending the output layers according to the new involved slots and initializing the extended weights of the output layers randomly. A new BLSTM for concepts in the next level is especially built and initialized in Figure 4(c).

In this paper, we mainly focus on the semantic slot differentiation. For example, slot *"time"* can be differentiated down into *"time"*, *"arrive.time"*, *"depart.time"* and *"return.time"*, and *"return.time"* can be differentiated up into *"return.day"* and *"return.month"* in the flight information domain. In other words, we mainly focus on the new involved slots that part of the atomic concept branch has existed in S.

5 Experiments

We evaluated our atomic-concepts based methods on two tasks of LU: (1) semantic slot refinement, and (2) domain adaptation.

5.1 Semantic Slot Refinement

In this task, we perform an adaptation from a set of semantic slots (source set) to a set of more complex slots (target set). It is a common problem in the development of commercial dialogue system. We simulate this problem in the ATIS dataset (Hemphill et al., 1995). Examples for the source and target data are shown in Table 2.

5.1.1 Datasets

ATIS: We use the standard ATIS corpus as the **target** set, which has been widely used as a benchmark by the LU community. The training data consists of 4978 sentences and the test data consists of 893 sentences. We randomly selected 80% of the training data for model training (3983 sentences) and the remaining 20% for validation.

ATIS_sd: For simulating the slot refinement, a **source** set is defined for the ATIS manually. All the differentiated semantic slots are gathered to their ancestral slots, as shown in Table 2. The maximum depth of the atomic concept trees is two in the target set, and only the bottom-level nodes are kept in the source set. The number of slots in source set is 45, and the number of slots in target set is 90. There are 45 atomic concepts in the bottom-level, and 10 in the top-level.

ATIS_X_test: To simulate the unmatched training and test sets, the standard ATIS test set is adjusted to ATIS_X_test. The value of each slot in a test sentence is randomly replaced with an unseen one. The unseen value sets are collected from the training set according to the bottom-level concepts (e.g. *"city_name"*, *"airport_name"*). For example, if the value set of *"fromloc.city_name"* is {*"New York"*, *"Boston"*} and the value set of *"toloc.city_name"* is {*"Boston"*}, then the unseen value for *"toloc.city_name"* is *"New York"*. The test sentence *"Flights to [xx:toloc.city_name]"* can be replaced to *"Flights to [New York:toloc.city_name]"*. Finally, the ATIS_X_test gets the same sentence number to the standard ATIS test set.

5.1.2 Experimental Settings

To estimate the efficiency of slot refinement, we use all the training data of ATIS_sd (source set) and part of the ATIS (target set) training data (like 50, 100, 500, 1000 and all sentences) to train an adapted slot filling model. Our models are evaluated on the test set of ATIS, and also the ATIS_X_test to test the generalization capability by simulating the unmatched training and test sets.

We deal with unseen words in the test set by marking any words with only one single occurrence in the training set as $\langle unk \rangle$. We also converted sequences of numbers to the string DIGIT, e.g. 1990 is converted to DIGIT*4 (Zhang and Wang, 2016). For the architecture of BLSTM, we set the dimension of word embeddings to 100 and the number of hidden units to 100. Only the

ATIS_sd	I want to go from [Boston:city_name] to [Atlanta:city_name] on [monday:day_name].
ATIS	I want to go from [Boston:fromloc.city_name] to [Atlanta:toloc.city_name] on [monday:depart_date.day_name].

Table 2: Examples of simulated slot refinement in the ATIS task. We use *[value:slot]* for annotation.

current word is used as input without any context words. For training, the network parameters are randomly initialized in accordance with the uniform distribution (-0.2, 0.2). The stochastic gradient descent (SGD) is used for updating parameters. The *dropout* with a probability of 0.5 is applied to the non-recurrent connections during the training stage for regularization.

We try different learning rates by grid-search in range of $[0.008, 0.04]$. We keep the learning rate for 100 epochs and save the parameters that give the best performance on the validation set. Finally, we report the F_1-score of the semantic slots on the test set with parameters that have achieved the best F_1-score on the validation set. The F_1-score is calculated using CoNLL evaluation script[1].

5.1.3 Our systems

System	Output label	Training set from
JS_T	Joint Slot	Target set
AC_T	Atomic Concept indep.	Target set
JS_TS	Joint Slot	+ Source set
AC_TS	Atomic Concept indep.	+ Source set
ACD_TS_1	Atomic Concept Dep.	+ Source set
ACD_TS_2	Atomic Concept Dep.	+ Source set

Table 3: Our systems for the slot refinement simulated in the standard ATIS task.

Different systems are built for comparison, as illustrated in the Table 3. **JS_T** uses only the training data of the target set (ATIS), and predicts the joint semantic slots (e.g. Figure 4(a)). **AC_T** also uses only the training data of the ATIS, and predicts the atomic concepts independently (e.g. Figure 4(b)). **JS_TS** and **AC_TS** use the training data of the source set (ATIS_sd) to pretrain the model (as referenced in Section 4.2), compared with **JS_T** and **AC_T** respectively.

ACD_TS_1 and **ACD_TS_2** predict the atomic concepts dependently. **ACD_TS_2** uses the pre-predicted concepts as additional features (e.g. Figure 4(c)). It resembles a two-steps model. We fix the parameters of the first model learnt from the ATIS_sd, and update the new model (the word embeddings are fixed and shared with the first model) learnt from the ATIS. As a simplification, **ACD_TS_1** gathers the pre-predicted concepts which are not *null* as the input for predicting concepts in the next level. For example, the prediction result of *"... from [New York:city_name] to ..."* is gathered to be *"... from [city_name] to ..."*.

[1]http://www.cnts.ua.ac.be/conll2000/chunking/output.html

5.1.4 Experimental Results and Analysis

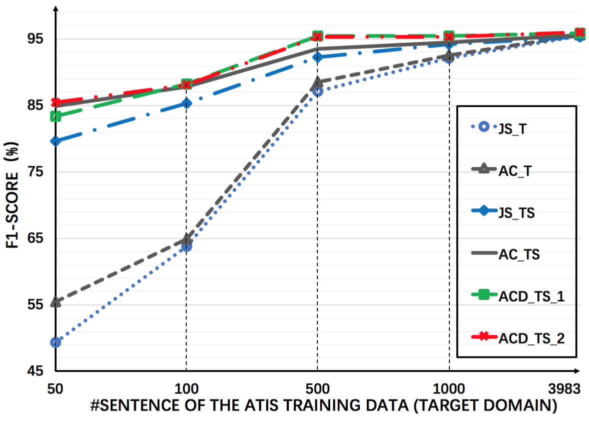


Figure 6: The results of different methods on the ATIS test set with different training sizes (50/100/500/1000/3983) of the target set (ATIS).

Semantic slot refinement (simulated)

Figure 6 shows the comparison of our systems on the ATIS test set with different training sizes of the target set (ATIS). We can see that:

1) The performance of each system is going to be better with more data of the target set. With sufficient data, all systems finally achieve similar results.

2) By using the data of the source set (ATIS_sd) for pre-training, the performance is improved with a big margin when the data of target set is limited. For example, **JS_TS** gains about 21% over **JS_T**, and **AC_TS** gains about 23% over **AC_T**, if only 100 sentences of the target set are available.

3) **ACD_TS_1** and **ACD_TS_2** get convergence with much less data (only 500 sentences) of the target set, by predicting the atomic concepts dependently. These two methods are very efficient for the slot refinement of LU.

Comparison with the existing systems

Since the target set of the simulated slot refinement is the standard ATIS, we can compare our systems with the published results on the standard ATIS test set. All the training data (3983 sentences) of the target set is exploited.

The results are shown in Table 4. We can see that RNN outperforms CRF because of the ability of capturing long-term dependencies. LSTM beats RNN by solving the problem of vanishing or exploding gradients. BLSTM further improves the result by considering the past and future features both. Encoder-decoder achieves the state-of-the-art performance by modelling the label dependencies. Encoder-labeler is a similar method to the Encoder-decoder. These systems are designed to predict the joint semantic slots traditionally.

AC_T gets a slight improvement (+0.15%) over **JS_T** by predicting the atomic concepts independently instead of the joint slots. Moreover, **ACD_TS_1** and **ACD_TS_2** predict the atomic concepts dependently. These two systems gain 0.31% (not significant) and 0.50% (significant level 95%) over the **AC_T** respectively. **ACD_TS_2** especially achieves the new state-of-the-art performance on the standard ATIS task.

Model	F_1-score	
	ATIS	ATIS_X_test
CRF (Mesnil et al., 2013)	92.94	–
RNN (Mesnil et al., 2013)	94.11	–
LSTM (Yao et al., 2014)	94.85	–
BLSTM (Zhang and Wang, 2016)	95.14	–
Encoder-decoder (Liu and Lane, 2016)	95.72	–
Encoder-labeler (Kurata et al., 2016)	95.66	–
Our BLSTM (JS_T)	95.43	79.59
Our Encoder-decoder	95.79	82.88
AC_T	95.58	80.90
ACD_TS_1	95.89	**86.28**
ACD_TS_2	**96.08**	86.16

Table 4: Comparison with the published results on the standard ATIS task. The performance of our systems on the unmatched test set ATIS_X_test.

Case study: As illustrated in Table 5, the joint slot filling (JS_T) predicts the label of *"late"* wrongly, whereas the atomic-concepts based slot fillings (i.e. AC_T and ACD_TS_1/2) get the accurate annotation. The word of "late" is never covered by the slot *"period_of_day"* in the training set. It is hard for the joint slot filling (JS_T) to predict an unseen mapping correctly. Luckily, the "late" is covered by the family of the slot *"period_of_day"* in the training set, e.g. *"arrive_time.period_of_day"*. Therefore, AC_T and ACD_TS_1/2 can learn this by modelling the atomic concepts separately.

Unmatched training and test sets

We also want to evaluate the system's generalization for the unseen values. Our methods are tested on the ATIS_X_test simultaneously. From Table 4, we can see that: 1) The typical slot filling model (**JS_T**) is not on par with other models. In contrary, the **Encoder-decoder** model outperforms the **JS_T** due to its label dependency modelling capability. 2) The atomic-concepts based slot filling gets a slight improvement over the **JS_T**, considering the concepts independently (**AC_T**). 3) The atomic-concepts based slot fillings (**ACD_TS_1** and **ACD_TS_2**) gain a large margin over **AC_T**, considering the concepts dependently. Because they provide more specific features for predicting the later atomic concept.

5.2 Domain Adaptation

5.2.1 Datasets

Our methods are also evaluated on the DSTC 2&3 task (Henderson et al., 2013) which involves a realistic domain adaptation problem.

DSTC 2 (source domain): The DSTC 2 dataset comprises of dialogues from the restaurant information domain in Cambridge. We use the **dstc2_train** set (1612 dialogues) for training and the **dstc2_dev** (506 dialogues) for validation.

DSTC 3 (target domain): The DSTC 3 introduces the tourist information domain about restaurant, pubs and coffee shops in Cambridge, which is an expanded domain of the DSTC 2. We use the seed data of **dstc3_seed** (only 11 dialogues) as the training set of the target domain.

DSTC3_S_test: In this paper, we focus on the three new semantic slots mainly: *"has_tv, has_internet, children_allowed"* [2]. They only exist in the DSTC 3 and have few appearances in the seed data. A test set is chosen for specific evaluation on these new semantic slots, by gathering all the sentences (688 sentences) whose annotation contains these three slots and selecting 1000

[2]For each slot of *"has_tv, has_internet, children_allowed"*, the semantic annotation *"request(slot)"* is replaced with *"confirm(slot=True)"*.

Reference	... could get in [boston:city_name] [late:**period_of_day**] [night:period_of_day]
JS_T	... could get in [boston:city_name] [late:**airport_name**] [night:period_of_day]
AC_T	... could get in [boston:city_name] [late:**period_of_day**] [night:period_of_day]
ACD_TS_1/2	... could get in [boston:city_name] [late:**period_of_day**] [night:period_of_day]

Table 5: Examples show how concept transfer learning benefits. We use *[value:slot]* for annotation.

sentences irrelevant to these three slots randomly from the *dstc3_test* set. This test set is named as **DSTC3_S_test** (1688 sentences).

The union of a slot and its act is taken as a joint semantic slot (e.g. *"confirm.food=Chinese"*), since each slot is tied with an act (e.g. *"inform"*, *"deny"* and *"confirm"*) in DSTC 2&3 task. The slot and act are taken as the atomic concepts. For the slot filling task, only the semantic annotation with aligned information is kept, e.g. the semantic tuple *"request(phone)"* is ignored. We use the manual transcripts as the input, and make slot-value alignment by spoken value matching simply.

5.2.2 Experimental Results and Analysis

The experimental settings are similar to the above ATIS experiments, whereas there is no validation data in DSTC 3. Therefore, we report the best F_1-score (performance upper bound) of our methods.

Model name	Label	Training set type	F_1-score on DSTC3_S_test
JS_T	JS	dstc3_seed	82.86
AC_T	AC	dstc3_seed	83.76
JS_TS	JS	+ dstc2_train	83.94
AC_TS	AC	+ dstc2_train	87.75
ACD_TS_1*	ACD	+ dstc2_train	**90.53**

Table 6: The performance of our methods in the DSTC 2&3 task. JS denotes joint semantic slot, and AC(D) denotes atomic concept (dependent).

The performance of our methods in the DSTC 2&3 task is illustrated in Table 6. We can see that: 1) **JS_TS** and **AC_TS** achieve improvements over **JS_T** and **AC_T** respectively, by using the data of source domain (dstc2_train) to pretrain the BLSTM arguments. 2) **AC_TS** gains more than **JS_TS**, as the joint semantic slot is represented by atomic concepts. The atomic concepts promote the associated slots to share input features for the same atoms. 3) The atomic-concepts based slot filling considering the concepts dependently (**ACD_TS_1*** [3]) gains 2.78% over **AC_TS** which considers the concepts independently. Because **ACD_TS_1*** makes more specific features for predicting the later concepts.

[3] Compared with ACD_TS_1, ACD_TS_1* gathers the pre-predicted concepts to be an unified symbol. e.g. $\langle CCC \rangle$.

Case study: Several cases are also chosen to explain why the atomic-concepts based slot filling outperforms the typical joint slot filling, as shown in Table 7. From the above part of Table 7, we can see **JS_TS** predicts a wrong slot. Because the grammar *"does it have [something]"* is only for the joint slot *"confirm.hastv"* in the seed data. From the below part of Table 7, we can see that only **ACD_TS_1*** which considers the concepts dependently predicts the right slot. Since *"confirm.childrenallowed"* never exists in the seed data, **JS_TS** can't learn patterns about it. Limited by the quantity of the seed data, **AC_TS** also doesn't extract the semantic correctly.

Reference	does it have [internet:confirm.hasinternet]
JS_TS	does it have [internet:confirm.hastv]
AC_TS	does it have [internet:confirm.hasinternet]
ACD_TS_1*	does it have [internet:confirm.hasinternet]
Reference	do they allow [children:confirm.CA]
JS_TS	do they allow [children:CA]
AC_TS	do they allow [children:CA]
ACD_TS_1*	do they allow [children:confirm.CA]

Table 7: Example show how concept transfer learning benefits. CA denotes *childrenallowed.*

6 Conclusion and Future Work

In this paper, we propose a novel semantic representation based on atomic concept trees, which are composed of combinatory concepts. We also present the concept transfer learning for adaptive LU on the atomic concepts level, to solve the problem of combinatory concepts extending in LU. The experiments on the ATIS and DSTC 2&3 datasets show that the concept transfer learning based slot filling obtains promising results, outperforming traditional slot filling, due to the knowledge sharing of atomic concepts.

In future work, we will investigate more advanced slot filling methods based on the atomic concepts. Furthermore, we want to explore more flexible semantic representation for the adaptive LU, and use it to find new semantic slot.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Yun Nung Chen and A. I Rudnicky. 2014. Dynamically supporting unexplored domains in conversational interactions by enriching semantics with neural word embeddings. In *IEEE Spoken Language Technology Workshop*. pages 590–595.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefvre. 2015. Zero-shot semantic parser for spoken language understanding. In *16th Annual Conference of the International Speech Communication Association (InterSpeech)*.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg.

L Heck and D Hakkani-Tur. 2012. Exploiting the semantic web for unsupervised spoken language understanding. In *Spoken Language Technology Workshop*. pages 228–233.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1995. The atis spoken language systems pilot corpus. In *Proceedings of the Darpa Speech and Natural Language Workshop*. pages 96–101.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2013. Dialog state tracking challenge 2 & 3. [online] Available: http://camdial.org/mh521/dstc/.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2077–2083.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *17th Annual Conference of the International Speech Communication Association (InterSpeech)*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(3):530–539.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*. pages 3771–3775.

Baolin Peng, Kaisheng Yao, Li Jing, and Kam-Fai Wong. 2015. Recurrent neural networks with external memory for spoken language understanding. In *Natural Language Processing and Chinese Computing*, Springer, pages 25–35.

Roberto Pieraccini, Evelyne Tzoukermann, Zakhar Gorelov, J-L Gauvain, Esther Levin, C-H Lee, and Jay G Wilpon. 1992. A speech understanding system based on statistical representation of semantics. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. IEEE, volume 1, pages 193–196.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, pages 19–24.

Ngoc Thang Vu. 2016. Sequential convolutional neural networks for slot filling in spoken language understanding. In *17th Annual Conference of the International Speech Communication Association (InterSpeech)*.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pages 189–194.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *INTERSPEECH*. pages 2524–2528.

Majid Yazdani and James Henderson. 2015. A model of zero-shot learning of spoken language understanding. In *Conference on Empirical Methods in Natural Language Processing*. pages 244–249.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*. pages 678–687.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*.

Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu. 2014. Semantic parser enhancement for dialogue domain extension with little data. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 336–341.

Su Zhu and Kai Yu. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*. pages 5675–5679.