EXTENDING DOMAIN COVERAGE OF LANGUAGE UNDERSTANDING SYSTEMS VIA INTENT TRANSFER BETWEEN DOMAINS USING KNOWLEDGE GRAPHS AND SEARCH QUERY CLICK LOGS

Ali El-Kahky[1]*,Xiaohu Liu*[2]*, Ruhi Sarikaya*[2]*, Gokhan Tur*[2]*, Dilek Hakkani-Tur*[2]*, Larry Heck*[2]

[1]Columbia University, USA
[2]Microsoft Research

[1] ame2154@columbia.edu

[2]{derekliu,ruhi.sarikaya,gokhan.tur,dilek.hakkani-tur,larry.heck}@microsoft.com

ABSTRACT

This paper proposes a new technique to enable Natural Language Understanding (NLU) systems to handle user queries beyond their original semantic schemas defined by intents and slots. Knowledge graph and search query logs are used to extend NLU system's coverage by transferring intents from other domains to a given domain. The transferred intents as well as existing intents are then applied to a set of new slots that they are not trained with. The knowledge graph and search click logs are used to determine whether the new slots (i.e. entities) or their attributes in the graph can be used together with transfered intents without re-training the underlying NLU models with the expanded (i.e. with new intents and slots) schema. Experimental results show that the proposed technique can in fact be used in extending NLU system's domain coverage in fulfilling the user's request.

Index Terms— Natural language understanding, knowledge graphs, intent trasnfer, semantic schemas, domain expansion.

1. INTRODUCTION

NLU systems have been built for specific tasks covering one or more domains. DARPA Air Travel Information System (ATIS) was the first large scale effort to build NLU systems covering air travel and hotel reservation [21] domains. Later such systems have been built by various groups to cover specific tasks [8]. All of these systems are implemented independently (without a global publicly available ontology/schema) with specific tasks in mind. Most of these systems are built in a fully supervised fashion requiring hand–labeled data for building domain, intent and slot models. NLU systems are typically composed of domain detectors, intent detectors and slot taggers. Domain detection is run to decide whether a query is in-domain (in the case of single domain NLU systems) or one of the multiple covered domains (in the case multi-domain NLU systems). Once the domain is identified, the respective intent detector and slot tagger are run to identify user's intent and tag entities in the query.

Recent advances with virtual personal assistant systems (e.g. Siri, Google Now, Dragon Assistant) covering a wide range of NLU domains boundaries between covered and uncovered NLU domains are becoming fuzzy. For example, an NLU system can be designed for handling only the air travel reservation task but the users may expect the system to handle new intents (i.e. actions) such as checking the flight status or making hotel and/or car reservations. These are all natural extensions of the original air travel reservation task. The standard approach to solve this problem is to re-design the semantic schema for the air travel reservation domain and add new intents and slots to cover related yet new domains. This also requires collecting and annotating additional data and re-training the NLU models.

In this paper, we propose a new technique which takes a step to transfer intents [1] from one domain to another and apply them to the previously unknown slots by leveraging knowledge graph and search query click logs. This method enables sharing intents and slots across independently constructed domains. For example, assume that we built two NLU systems: 1) "events" domain where the users can "search" and "buy" tickets to concerts, games and other activities (e.g."buy a ticket to Justin Timberlake concert in Seattle"), 2) "places" domain where the users can search for businesses and places and asks for phone numbers, driving directions to them. In the first system "get_driving_directions" or "call_business" intents are not covered but they are covered in the second system. If the user says "get me driving directions to the concert" in the second turn of a conversation session, which is a natural follow up query, the method presented in this paper can be used to automatically cover the new intent "get_driving_directions". The method achieves this by using knowledge graph [3, 16, 17] and search engine click logs.

This paper is organized as follows: in Section 2, we explain how to use the knowledge graph for intent transfer. In Section 3, we describe how to use the search query click logs to ensure sensible intent transfers between domains. We summarize the related work in Section 4 and present experimental results in Section 5. Conclusions and possible future directions are presented in the final section.

2. INTENT TRANSFER USING KNOWLEDGE GRAPH

Each intent in an NLU system corresponds to an action the system executes and each action requires a set of inputs in the form of slots (extracted from the user query) either to fetch information from the back-end or show results to the user. When an intent is transferred to a new domain, it is likely that it will be applied to the slots that it was not trained with in the domain it is coming from. So the question is whether we can find legitimate slots in the new domain to execute the correct system action for the new intent. In the example of "show me driving directions to <concert_event> Justin Timberlake concert <concert_event>", "get_driving_directions" is a new intent transferred to the "events" domain from the "places" domain. This intent takes "address", "location", "business" as the possible

[1]"Intent transfer" means two things: 1) "transferring" an intent to new slot(s), 2) transferring an intent from one domain to another.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

slot types but not the "concert_event" slot type in the "places" domain.

We use the knowledge graph to lookup entities from the query and decide if the given value (e.g. "Justin Timberlake concert") for a slot (e.g. "concert_event") has a type in the knowledge graph that is compatible with the supported slot type (e.g. "address", "location", "business") of a given intent (e.g. get_driving_directions). If the entity is not compatible with types observed in the training data, we use the knowledge graph to find a property (i.e. attribute) of that entity that has a compatible type. By compatible, we mean that two types are the same or one of them is sub-type of the other in the knowledge graph. In our example, the intent "get_driving_directions" can only support type "location.location", therefore "address" and "location" are compatible slot types, but "concert" and "business" are not. We describe how to find the compatible slot types and sub-types for a given intent using the knowledge graph in the next subsection.

2.1. Weighting Slot Types Using Knowledge Graph

In order to find accepted slot types for an intent in NLU domains, we first study intents in all NLU domains for which we have labeled data and semantic schema. For all intetnts, we want to find out the relation between slots and types in the knowledge graph. Figure 1 shows an example list of slots in the NLU domains connected to entities that appear in those slots in NLU training data with a weight equals to the number of times each entity appears with each slot. For example, "Boston" is used as "Travel Departure City" four times in the training data. These entities are connected to the different graph types shown on the right side in the figure. Note that an entity can have a large number of types in the knowledge graph. For example "Chicago" is an entity and it has 29 types such as "location", "city", "team location", "travel.destination","music.record label"," book.subject", "olympics.bidding city".

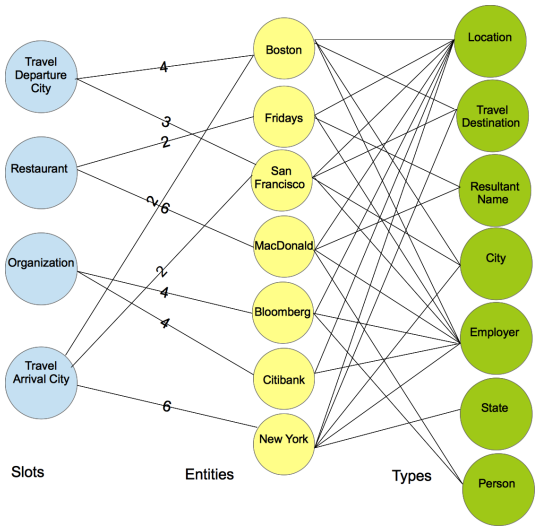

Fig. 1. Slots, their entities and knowledge graph types

We consider two approaches to rank the graph types for each slot. First, we weight a type t in slot d in a naive way using the following expression:

$$s_{t,d} = \sum_{e \in entities(d): \exists edge(e,t)} weight_{d,e} \quad (1)$$

where $entities(d)$ is the entity set for slot d, and $weight_{d,e}$ denotes the count of entity e appears in slot d. For example the score of type "City" in slot "Travel Departure City" is $3 + 4 = 7$, since "Boston" and "San Francisco"both are of type "City" and they appear in slot "Tavel Departure City" 4 and 3 times respectively.

The second approach is to weight graph types for all slots jointly by TF-IDF. Term Frequency Inverse Document Frequency (TF-IDF) is a widely used metric in numerous natural language processing applications. TF-IDF consists of computing Term Frequency (TF) and Inverse Document Frequency (IDF) and TF-IDF score is the product of the two. We first create a vector of entities for each slot j with length equal to total number of entities and assign the ith component of this vector to be $s_{i,j}$ which is computed in Eqn. (1). Then we apply an algorithm similar to the TF-IDF re-weighting but with two modifications.

In this work, we consider using the graph types as the terms and slots as the documents. Equations for TF for type t from the knowledge graph and IDF for type t in slot d are as follows:

$$TF(t,d) = 0.5 + \frac{0.5 f(t,d)}{max_{t' \in d} f(t',d)} \quad (2)$$

$$IDF(t) = log \frac{|D|}{\sum_{d \in D} \mathbb{1}\{t \in d\}} \quad (3)$$

where $|D|$ is the number of slots we have, $\mathbb{1}\{t \in d\}$ is one if type t appears in slot d and $f(t,d)$ is then number of time type t appears in slot d.

The basic idea is to rank frequent types higher if they are specific types to a certain slot. In our example, the first approach in Eqn. (1) gives graph type "employer" a score 7 for slot "Travel Departure City" which is the same as types like "City" and "Travel destination". While in the TF-IDF approach, "Employer" will rank lower for "Travel Departure City" since it is associated with many other slots.

One issue in our example is that graph type "State" in slot "Travel Arrival City" will get a higher score than graph type "Travel Destination" because "Travel Destination" appears with two slots while "State" appears with only one slot. To adress this issue, we modify IDF so that it does not down-weight a graph type even if appears in many different slots as long as those slots are similar. We measure similarity between slot d' and d as the percentage of entities appear with both slots denoted by $share(d',d)$. The modified IDF equation becomes:

$$IDF(t,d) = log \frac{|D|}{\sum_{d' \in D} e^{-\alpha shared(d',d)}} \quad (4)$$

where α is a tuning parameter that controls how fast the decay happens. We tune α by optimizing the graph type rank performance on a small development set of 5 slots. With this modification, IDFs are slot specific.

The second issue in Eqn. (1) is that it does not assign weights to the edges between entities and different graph types. A simple fix for this is to give a weight that is inversely proportional to the number of graph types appearing with each entity. This has the effect of weighting ambiguous entities (entities with many different graph types) lower than unambiguous ones. In our example, edges between "Boston" and its types will get a weight of $\frac{1}{4}$ while edges between "New York" and its types take $\frac{1}{5}$. This will change $f(t,d)$ used in calculating TF to be :

$$f(t,d) = \sum_{e \in entities(d): t \in types(t)} \frac{1}{|N_e|} \quad (5)$$

Table 1. Top graph types for slots using two different weighting approaches

Slots	Top types learned using TF-IDF	Top types learned using modified TF-IDF
Restaurant: name	business.employer business.operation organization.organization	local.restaurant business.operation organization.organization
Travel: absolute location	location.location statistics.economic group statistics.population group	travel.destination periodicals.newspaper circulation area location.dated location
Restaurant: cuisine	book.subject location.location food.cuisine	food.cuisine food.food food.dish
Games: character	fictional universe character film character game character	fictional universe character game character film character

where N_e is the total number of graph types entity e has. The function $entities(d)$ returns all entities in document d (i.e. slot d).

Table 1 shows some example results for top graph types retrieved with the two approaches. With the enhanced TF-IDF, type "local.restaurant" is correctly ranked as the top type for slot "Restaurant:name".

2.2. Finding Entities Compatible with the Transferred Intent

After getting a weighted list of graph types for each slot, we can use a simple search algorithm to find the best entity that may be accepted by an intent. The pseudo-code for finding the best compatible entity is shown in Algorithm 1. The knowledge graph is used to look up the entity type, parent types and all of its properties, and return the entity itself or one of its properties (i.e. attributes) that is most compatible with the ranked slot types for the intent. The $*$ in Algorithm 1 denotes the dot product operator which simply measures the match between an entity types and the excepted types for the slot.

With this algorithm, we can enable existing intents (not transferred) within a domain to accept new slots, which are not covered in the existing semantic schema. For example, in the TV domain semantic schema "change_channel" intent never appears with the "tv_show" entity. However, the user can say "change channel to late night show" and expect the system to work. We can now transfer "change_channel" intent to take entity "tv_show" since we learned that "tv_channel" is an attribute (i.e. property) of "tv_show" in the graph.

3. INTENT TRANSFER BETWEEN DOMAINS USING SEARCH ENGINE LOGS

In addition to intent transfers to support more slots within the same domain, we also find out if certain intents can be used within other domains applied to similar or different slot types. We train a classifier to automatically validate a potential intent transfer between domains.

First, we identify the lexical and syntactic patterns that are associated with an intent using the following simple heuristics. For each intent we extract the top N frequent patterns (i.e. intent_patterns) used by this intent. This is achieved by extracting the most frequent word tri-grams that do not contain any slot values and contain no more than one stop word. Examples of top intent patterns discovered by this technique for some intents are shown in Table 2.

Procedure 1 Finding the most comptatible entity or entity attribute for a given slot

INPUT: : SlotTypes: list of top N types for the slot
E: entity to fill the slot
OUTPUT: : (EntityName $\rightarrow$ P.name, PropertyValue $\rightarrow$ P.value): the best entity for the given slot
BestMatch=null, BestScore=-1
for all type T in SlotTypes **do**
  if E has T as type **then**
    $Return(Root, E)$
  end if
end for
for all type T in SlotTypes **do**
  for all property P in E.properties **do**
    if T $\in$P.types and P.types * SlotTypes $>$ BestScore **then**
      $BestMatch = (P.name, P.value)$
      $BestScore = P.types * SlotTypes$
    end if
  end for
  $ReturnBestMatch$
end for

After top patterns are extracted, we collect a list of entities that appear both in the training data and target domain which can potentially co-occur with the new intent. Artificial queries are generated using pairs of ⟨intent_pattern, entity⟩. We then mine the search logs to look for evidence that the intent patterns, which are essentially word ngrams and domain (represented by the entity) are good combinations. For example, a pair of ⟨Places:get_phone_number, Flights:flight_airline⟩ is used to test if "get_phone_number" can be transfered to "Flights" domain, and ⟨Finance:find_stockinfo, Games:game_company⟩ is used to check if "find_stockinfo" can be transfered to "Games" domain.

Note that not all candidate pairs are valid transfers. Some invalid transfer candidates still get nonzero search results because of ambiguous entities. For example the pair ⟨Movies:find_cast, Games:game_character⟩ is invalid, but because some movie names are also game characters (e.g. Superman), we can find many matched queries in the search logs.

In order to assess whether an intent transfer is valid we train a logistic regression classifier. The training data set for the classifier consists of 621 (intent_pattern, entity) pairs manually labeled as valid or invalid. For feature set, we use features from the search results (e.g. number of search results supporting this intent, domain pair, number of unique entities appeared in the mined queries).

This approach assumes that entities used for searching the logs belong to only the target domain. In order to handle the issue of ambiguous entities, we use a list of weighted entities for each domain, where each entity has a score denotes how likely it is from that domain. Such weighting list can be obtained from work like in [6], or scores from knowledge graph or the proposed TF/IDF in Section 2.1. We compare three weighting approaches and present the results in Section 5.

Table 2. Examples of top patterns used for intents

Intent	Top Patterns
Travel:checkflight	find flights to, search for flights, show flights to, find flights from, find flights for
LiveTV:change channel	change the channel, to watch channel, switch to channel, change to channel, turn to channel
Weather:check weather	weather forecast for, the weather forecast, show me weather, show weather for, the weather conditions

4. RELATED WORK

The task of intent determination is emerged partly from commercial call-routing applications for customer care centers, such as AT&T How May I Help You system [8]. While the routes are motivated by the business needs, for call classification, it makes more sense to consider semantically coherent utterances, as defined by intents, which can then be mapped to routes [10]. Later, unsupervised learning approaches have been proposed to cluster these intents automatically using incoming calls [2]. To the best of our knowledge, the task of transferring intents to expand the NLU domain coverage is new and no previous work have been done to this specific point. On the other hand, there have been some recent work on using query click logs and knowledge graphs in NLU systems.

Query logs have been shown to be useful in improving domain and intent classification [4, 13, 14, 20], and bootstrapping models for new domains [7]. On another study,we have augmented the training data for intent detection using similar queries via a Bayesian framework [1]. Availability of the knowledge graphs has resulted in new studies in the NLU field. For example, traversing the social graph have been proposed [17] for Facebook, linking textual documents such as Wikipedia to knowledge graphs is studied in this framework [3, 16].

In our previous work [12], we used the Wikipedia entries of the knowledge graph nodes to bootstrap slot filling models. Similarly, in the Semantic Web community, pattern matching based methods have been used [9, 15, 18]. We have focused on combining semantic knowledge graph with query click logs for various tasks. Most recently, we have used the query click logs and the existing knowledge graph to detect relations or intents missing the graph using distant learning [11]. For slot filling, [19] used seed structured in-domain sites in the nodes of the knowledge graph for a given domain to extract the frequent queries hitting those sites. Those queries are then used as bootstrap training data for slot filling. The node pairs in the knowledge have been used for web search to realize their natural language forms to bootstrap relation or intent detection models [5].

5. EXPERIMENTS AND RESULTS

We first created a set of test scenarios to evaluate intent transfer to new slots. Some example scenarios are given in Table 3. The template slots are slots desined for intents, and test slots are slots to which we want to transfer intents. For example, the intent "change_channel" is designed to take slot "channel_name", but we want to verify if the intent can be transfered to"tv_show". From all scenarios, 200 examples are created by representing intents with intent n-gram patterns and slots with slot values. We evaluate the accuracy of intent transfer by slot filling in five domains and the test results are presented in Table 4. The results are promising since those queries were not be accepted in the baseline system. Also the results are expected to improve further with the fast growth of the knowledge graph.

Table 3. Examples of slot filling scenarios

Domains	**Intents**	**Template Slots**	**Test Slots**
Movies	find movie	movie name	movie writers
LiveTV	change channel	livetv channel name	TV show
Weather	check weather	absolute location	university name

Table 4. Slots filling results

Domains	**Accuracy**
LiveTV	86.70%
Movies	93.90%
Flight	70.00%
Weather	76.70%
Finance	62.50%
Sports	81.80%

Table 5. Intent transfer to new domains

Domains	**Accuracy**
Restaurant	80.00%
Flight	63.30%
Local Business	75.60%
Travel	91.30%

In order to evaluate intent transfer across different domains, we used the mining technique described in Section 3 to extract candidate list of intents and domains from all 26 domains. Logistic regression models are trained with three different weighting methods: the basic entity weighting using equation (1), entity weighting using search engine logs in [6] and the modified TF-IDF weighting in Section 2.1. Accuracy is measured as the percentage of valid transfer predictions out of all mined transfers. The results are shown in Table 6 across all domains. The proposed weighting approach is much better than basline and other weighting methods.

We aslo evaluate transferring intents to a new domain. A classifier built using training data from all domains other than the target domian. We take a set of entities in a target domain and pair those entities with intents from all existing domains to check if any intents can be transferred to the target domain using the classifier. The experimental results in four target domains are shown in Table 5. For example, in the target "Travel" domain, 91.3% of intents that the model predicted as transferable from other domain are valid.

Table 6. Intent transfer across existing domains

Models with different weighting approaches	**Accuracy**
Baseline: without weighting	54.3
Weighting using Equation 1	59.2
Weighting using search logs [6]	60.8
Weighting using modified TF-IDF	74.9

6. CONCLUSIONS

We propose two automatic approaches to expand the coverage of an NLU system using a set of existing NLU domains, search logs and the knowledge graph. First we expand domain coverage by identifying new slot types that a given intent can consume, and second we transfer intent from one domain to a new domain and pair up with new slots in the new domain. The experimental results show that the proposed techniques are promising and they can be potentially be used to handle out-of-domain and/or out-of-schema queries directed to an NLU system.

7. REFERENCES

[1] Gokhan Tur Asli Celikyilmaz, Dilek Hakkani-Tur. Leveraging web query logs to learn user intent via bayesian discrete latent variable model. In *ICML*, 2011.

[2] S. Bangalore, G. Di Fabbrizio, and A. Stent. Learning the structure of task-driven human-human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1249–1259, 2008.

[3] J. Christensen and M. Paşca. Instance-driven attachment of semantic annotations over conceptual hierarchies. In *Proceedings of the EACL*, 2012.

[4] D. Hakkani-Tür Dilek, Larry P. Heck, and G. Tur. Exploiting query click logs for utterance domain detection in spoken language understanding. In *ICASSAP*, 2011.

[5] Larry Heck Dilek Hakkani-Tur and Gokhan Tur. Using a knowledge graph and query click logs for unsupervised learning of relation detection. In *ICASSAP*, Lyon, France, 2013.

[6] Gokhan Tur Dustin Hillard, Asli Celikyilmaz and Dilek Hakkani Tur. Learning weighted entity lists from web click logs for spoken language understanding. In *In Prooceedings of the Interspeech*, 2011.

[7] Bootstrapping Domain Detection Using Query Click Logs for New Domains. Bootstrapping domain detection using query click logs for new domains. In *In Prooceedings of the Interspeech*, Lyon, France, 2011.

[8] A. L. Gorin, G. Riccardi, and J. H. Wright. How May I Help You? *Speech Communication*, 23:113–127, 1997.

[9] R. Guha, R. McCool, and E. Miller. Semantic search. In *Proceedings of the WWW*, Budapest, Hungary, 2003.

[10] N. Gupta, G. Tur, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Rahim. The AT&T spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):213–222, 2006.

[11] D. Hakkani-Tür, A. Celikyilmaz, L. Heck, and G. Tur. A weakly-supervised approach for discovering new user intents from search query logs. In *In Prooceedings of the Interspeech*, Lyon, France, August 2013.

[12] Larry P. Heck and Dilek Hakkani-Tür. Exploiting the semantic web for unsupervised spoken language understanding. In *SLT*, 2012.

[13] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *Proceedings of the ACM SIGIR*, Singapore, 2008.

[14] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the ACM SIGIR*, Boston, MA, 2009.

[15] S. A. McIlraith, T. C. Sun, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, pages 46–53, 2001.

[16] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the ACL*, Singapore, 2009.

[17] L. Rasmussen. The natural language interface of graph search. In *Proceedings of the ACL*, Sofia, Bulgaria, 2013.

[18] N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited. *IEEE Intelligent Systems*, pages 96–101, 2006.

[19] G. Tur, M. Jeong, Y.-Y. Wang, D. Hakkani-Tür, and L. Heck. Exploiting the semantic web for unsupervised natural language semantic parsing. In *In Prooceedings of the Interspeech*, Portland, OR, September 2012.

[20] Y.-Y. Wang, R. Hoffmann, X. Li, and J. Szymanski. Semi-supervised learning of semantic classes for query understanding: From the web and for the web. In *Proceedings of the CIKM*, Hong Kong, 2009.

[21] Wayne Ward et al. The cmu air travel information service: Understanding spontaneous speech. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 127–129, 1990.