A Discriminative Model Based Entity Dictionary Weighting Approach for Spoken Language Understanding

Xiaohu Liu, Ruhi Sarikaya

Microsoft Corporation, Redmond, USA
{derekliu,ruhi.sarikaya}@microsoft.com

Abstract

Spoken language understanding (SLU) systems use various features to detect the domain, intent and semantic slots of a query. In addition to n-grams, features generated from entity dictionaries are often used in model training. Clean or properly weighted dictionaries are critical to improve model's coverage and accuracy for unseen entities during test time. However, clean dictionaries are hard to obtain for some applications since they are automatically generated and can potentially contain millions of entries (e.g. movie names, person names) with significant noise in them. This paper proposes a discriminative model based approach to weight entities in noisy dictionaries using multiple knowledge resources. The model makes use of features extracted from query click logs, knowledge graph and live search results for accurate entity weighting. Experiments for both intent detection and slots tagging tasks in entertainment search covering five domains show significant gains over the baselines.

Index Terms: Spoken language understanding, named entity lists, query click logs, knowledge graphs.

1. Introduction

Spoken language understanding (SLU) systems aim to identify the user's intent from the natural language query and extract semantic slots to fulfill the user's request [1, 2, 4, 5]. Typically, SLU systems use classifiers such as Support Vector Machine (SVM) [6, 7] trained on supervised data for domain and intent detection, and taggers such as Conditional Random Field (CRF) models [8] for slot tagging. When a user issues a query, domain and intent classifiers determine the domain and the intent of the query and the slot tagger tags semantic slots associated with the intent. The tagged slots are used to fetch results from the knowledge backend.

In order to train the classifiers we collect a set of in-domain queries belonging to each domain that the application supports. These queries are then annotated manually using a pre-defined semantic schema. The semantic meaning of a query is represented with a tuple of domain, intent, and slot list in the semantic space.

⟨DOMAIN, INTENT, SLOT_LIST⟩

A slot list is a list of key-value pairs:

⟨SLOT_NAME, SLOT_VALUE⟩

A query belongs to one or more domains, such as MOVIES, APPS, PLACES, or GAMES. Within a domain, the query is further classified into one of domain-specific intents, such as FIND_MOVIES, PLAY_MOVIES, PURCHASE_SONGS, PLAY_SONGS to precisely determine the user's intent. Semantic slots associated with each intent specify detailed parameters for the intent. For example, the semantic representation of the query "find the most recent tom cruise movies" is as follows:

⟨MOVIES, FIND_MOVIES, [⟨ MOVIE_STAR, "tom cruise" ⟩, ⟨ RELEASE_DATE, "most recent"⟩] ⟩

Classifiers are trained with features from the query, the conversation context and background knowledge. It has been proven that high quality dictionaries can provide valuable features to make the SLU models more robust to unseen entities in the training data at runtime by providing broader entity coverage [9]. A dictionary is a list of entities that belong to the same semantic category. For example, a movie name dictionary contains a large number of movie names. Knowing whether a query contains an entity in a dictionary is a good signal (i.e. feature) to use in all SLU models. Large high quality dictionaries are however hard to obtain since they are automatically created through data mining without human validation. Some of these dictionaries can contain as many as millions of entities (e.g. song titles, movie names) making manual inspection and cleaning costly. Moreover, there is also on-going cost to keep the dictionaries up to date. For example, new movies and songs come up at a rate faster than can be manually processed. For these reasons, large dictionaries contain large amounts of noise them. The noise comes in two main forms: 1) illegitimate entities added to the dictionaries, 2) inconsistency in the entity tokenization and/or normalization. If the amount of noise in the dictionary is high, using dictionary features in SLU models does not provide any value and, in some cases, can even hurt the model performance. There is a need to automatically clean or more generally to weight dictionary entities. The weight assigned to an entity reflects the likelihood that it is a genuine entity. A low weight indicates that the entity is more likely to be noise.

Previously a dictionary weighting approach was proposed in [9], which automatically learns entity weights by mining user clicks from web search logs. We found that this weighting algorithm has limitations. For example, since it only uses search click logs [9], many legitimate entities are not weighted properly. If entities are not included in the logs, this weighting technique has no evidence to assign proper scores to those entities. As a result, those entities are treated as unknown and given a default low weighting score.

In this work, we propose a new discriminative model based approach that leverages multiple knowledge sources to improve dictionary weighting performance. The proposed technique uses a knowledge graph and live search results in addition to the query click logs. This model based approach has the advantage to incorporate multiple sources of signals during the weighting process and therefore learns the proper weights for each entity.

In the next section, we describe the algorithm work flow. In Section 3, we describe how a knowledge graph is used to extract useful entity type features. In Section 4, we explain how we leverage search click logs and enhance the technique by using a live search engine. Then in Section 6 we explain how to build SVM classifiers using all features and weight candidate dictionaries. We evaluate the weighting approach for assisting intent and slot models in five SLU domains: movies, music, applications, games, and places. The experimental results are presented in Section 7 followed by the conclusion in Section 8.

2. Proposed Dictionary Weighting Architecture

SLU systems are trained to generate the best semantic representation using models built from queries annotated with semantic labels. Entity dictionaries are also heavily used to build large web-scale SLU systems to improve model's coverage and thus accuracy for unseen entities in the training data during runtime. These dictionaries (i.e. "candidate dictionary" in Fig. 1) are automatically mined from the web, search logs and other resources and consequently contain significant amount of noise. A typical movie name dictionary may contain hundreds of thousands movie names, but it also contains significant amount of invalid movie names or entities are not actually movie names. For example, many alphabetic letters and digits appear in the collected movie name dictionary, some of which may be valid movies, but many of which are not.

The goal of this study is to assign a weight to each entity in the dictionary. The weight determines the likelihood that the entity is indeed a member of the target entity type (e.g. movie name). Using the assigned weights, a large dictionary is divided into multiple dictionaries resulting in dictionaries weighted from high to less likely entity lists; the least likely is expected to capture the invalid entities in the full dictionary.

In Fig. 1 we depict the proposed dictionary weighting architecture. We use three resources to weight dictionaries: 1) a seed entity list obtained from the supervised training data (e.g. all the entities tagged as "movie name" in the training data), 2) search click logs and live search results, 3) a knowledge graph. The seed list is high quality due to human supervision and serves as positive samples to train the discriminative weighting model. A random entity list is also extracted from Bing search logs and is used as the negative samples for training the model. The model learns to discriminate between a genuine entity vs. a random entity, which could any entity in the web.

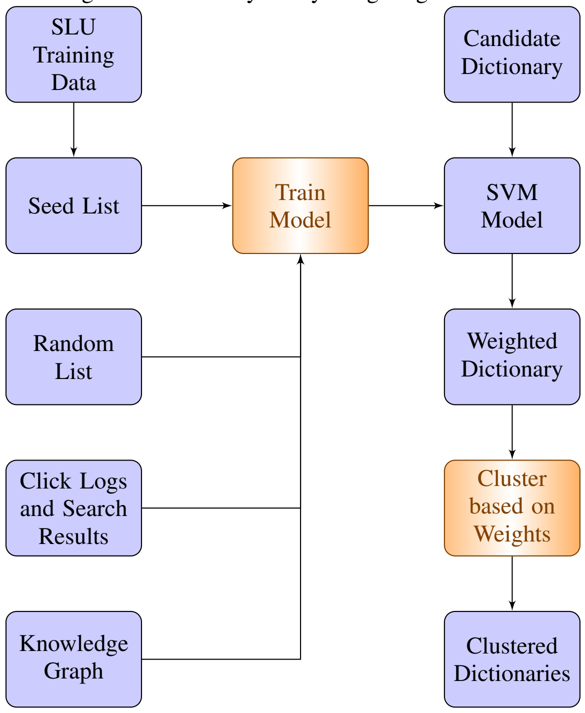
Search engines return multiple search results represented by URLs. The URLs clicked by users are considered more relevant to the query. A click log for a search query contains the query itself and URLs clicked by users in the format of:

$$\langle \text{QUERY}, [\langle \text{CLICKED\_URL}, \text{CLICK\_COUNT} \rangle] \rangle$$

Whether or not an entity belongs in an entity dictionary can be evaluated by considering the domain of the websites users click on when searching for that entity. An entity can be weighted by comparing the click distribution of the entity and aggregated click distribution of the seed list.

The knowledge graph contains rich information about entities, including entity types. Entity type is used as an important resource for additional features in the proposed weighting approach to train a dictionary weighting model. For each entity, features extracted from the search click logs and knowledge graph are used to augment lexical n-gram features derived from the entities in both the seed list (positive class) and random list (negative class). A binary SVM model is trained using the feature representations to weight a candidate dictionary (shown in Fig. 1) which contains noisy entities. Each candidate entity is given a weighting score. Using K-means clustering approach, the candidate dictionary is then split into multiple clusters. Each cluster forms a smaller dictionary and is used to generate additional features for training downstream SLU models. SLU models should give different feature weights to the new dictionaries. The noisy dictionaries should be weighted lower in the SLU models. As a result, the impact of the noise in candidate dictionaries on SLU models is reduced.

Figure 1: Dictionary Entity Weighting Architecture



3. Knowledge Graph Features

A knowledge graph (e.g. freebase.com) represents a generic semantic space using entities (e.g. movie names, persons, places, organizations, etc.) and relations between entities. The graph contains a large set of tuples in a resource description framework (RDF) defined by W3C. A tuple typically consists of two entities: a subject and an object, and linked by some relation. Knowledge graphs have powered major commercial search engines as the graphs contain huge number of entities, attributes of the entities, and relation with other entities. Moreover, a knowledge graph is not static, it grows over time and is updated periodically. Knowledge graphs are a valuable resource for SLU systems as well [11, 12]. Traversing the social graph has been proposed [13] for Facebook. Linking textual documents, such as Wikipedia, to knowledge graphs is studied in this framework [14]. Using the semantic space defined in the knowledge graph for SLU tasks relies heavily on the research of semantic web [15, 16] and semantic search [17].

One of the key contributions of this work is to use the knowledge graph to weight dictionary entities. The most interesting information is the entity type defined in the graph for every entity. In the knowledge graph, the relation "type" represents the entity type. Table 1 shows some examples of entities and their relations in the knowledge graph. From the graph, we learn that "Romeo & Juliet" could be a film name, or music album as it has two types: "film.film" and "music.album". The

Table 1: Entities and relation examples in the knowledge graph

Subject	Relation	Object
Ridley Scott	type	film.producer
Ridley Scott	type	tv.producer
Romeo & Juliet	type	film.film
Romeo & Juliet	type	music.album
Romeo & Juliet	producer	Baz Luhrmann
Romeo & Juliet	producer	Ridley Scott

Table 2: Top clicked URLs of two movies

A lot like love	Romeo & Juliet
imdb.com	imdb.com
en.wikipedia.org	en.wikipedia.org
rottentomatoes.com	rottentomatoes.com
movies.msn.com	shakespeare.mit.edu
movies.yahoo.com	sparknotes.com

type information from the knowledge graph can be helpful to weight an entity list of “movie name”, “music track”, or “movie producer”. Another example of a type relation in the table is “producer”: “Baz Luhrman” and “Ridley Scott” are two of the producers of the film “Romeo & Juliet”.

We currently only use the type information from the graph as features to build discriminative classifiers. A knowledge graph search tool is used to search entities in the graph. The tool returns multiple matched entities and each match is associated with a score to represent the relevance of the matched entity with the input entity [18]. A series of Deep Structured Semantic Models (DSSM) is proposed for entity ranking. A deep neural network is used to rank a set of entities for a given input. First, a non-linear projection is performed to map the input and the entities to a common low-dimensional semantic space. Then, the relevance of each entity given the query is calculated as the cosine similarity between their vectors in that semantic space.

Each matched entity may have multiple types defined in the graph. We compute the score of a type t for a given entity e as follows:

$$K_e(t) = \sum_{e_i \in kgsr(e), t \in types(e_i)} \frac{sim(e, e_i)}{|types(e_i)|} \quad (1)$$

where $kgsr(e)$ is the knowledge graph search results for e, and $types(e)$ denotes all possible entity types of e. The score $sim(e, e_i)$ assigned by knowledge graph search tool measures the similarity between a matched entity e_i and the input entity e using algorithms proposed in [18].

4. Search Click Log Features

Large-scale search engines such as bing.com serve millions of queries per day. Together with the search queries, user clicked URLs are also logged anonymously. Query click logs implicitly encode information that associates a query to relevant web sites. This information is used for many natural language processing (NLP) tasks. In [10] a large query log of more than twenty million queries is used to extract the semantic relations that are implicitly captured in the actions of users submitting queries and clicking answers. In [19], n-gram features are extracted from query-url pairs to enhance the web search ranking task. Click logs can also be used to refine entity lists extracted and processed from noisy sources [9]. For example, the top 5 clicked URLs (shortened by domain names only) for movies “A lot like love” and “Romeo & Juliet” are given in Table2. We see that 3 URLs are common out of top 5 clicked URLs. These URLs are mostly movie domain specific. We can use the clicked URLs as features to determine the likelihood of an entity being a member of a movie dictionary. The intuitive assumption behind this is that, for entities belonging to the same dictionary: (1) URLs returned from search engine converge to a small set; (2) the URLs clicked the most are more relevant to the query.

One issue with using only click logs is that some entities may not be covered in the query logs since logs are extracted for a limited time frame (e.g. six months) for computational reasons. It is not computationally possible to process all the search queries that ever hit a search engine. Even the search engines employ a moving time window for processing and storage of the search logs. In these cases there may be no evidence we can use to weight these entities. For example, “apollo thirteen” is a movie name that appeared in the movie training data, but it does not appear in search logs. In addition to the limit on date range for log extraction, the most common reasons for missing logs are: (1) some entities are very old, and they are not searched recently, e.g. “apollo thirteen” was released 18 years ago; (2) some entities are quite new and they have not shown up in the logs. One way to solve the issue of missing logs for entities is to search bing.com at real time, so that we can have extra URLs to use in addition to URLs in search logs. Given that the search engine is updated on a daily basis, real-time search can make sure we capture the newest entities.

We run live search through the search engine for all entities no matter if they appear in click logs or not. Each URL returned from a live search is considered to have one click. For simplicity, we refer all URLs, both from click logs and live search results, as “clicked URLs” hereafter since they are utilized similarly in entity weighting process.

5. Cross-Entropy Based Entity Weighting Approach

Here, we implemented the baseline cross-entropy based dictionary weighting technique [9]. In [9] a score function that reflects the likelihood that an entity is a true member of a dictionary (e.g. movie name entity list) using clicked URLs is defined. The resulting score is expected to be higher for entities that have a clear membership (e.g. ”sleepless in seattle”) in the dictionary, and lower for entities that are ambiguous (e.g. ”start up”) or do not belong to the dictionary at all.

We measure how likely a URL is clicked for entities in a given entity list as follows:

$$P(url|entityList) = \frac{Count(url|entityList)}{\sum_i Count(url_i|entityList)} \quad (2)$$

The count function is defined as total clicks from all entities in the list:

$$Count(url|entityList) = \sum_{e \in entityList} Count(url|e) \quad (3)$$

where $Count(url|e)$ is the number of times the url is clicked when the search query is e.

To weight an entity in a dictionary, we compare the clicked URL distribution in two entity lists. The first one is a seed list representing the target dictionary and it consists of all entities annotated with the same entity tag in the SLU training data, such as, "movie name", "song name", "place name", etc. The second list is a collection of entities randomly extracted from the search logs representing the background entities. For each entity, a weighting score is computed based on clicked URL distribution over two entity lists:

$$\begin{aligned}L(e) &= \sum_i (P(url_i|e) * log(P(url_i|seedList)) - \\ &\quad \sum_i (P(url_i|e) * log(P(url_i|randomList)) \\ &= \sum_i P(url_i|e) * log(\frac{P(url_i|seedList)}{P(url_i|randomList)})\end{aligned} \tag{4}$$

Essentially $L(e)$ measures the difference of two cross entropies: the first cross entropy is for the clicked URL distribution in the entire seed list, and the second cross entropy is for the clicked URL distribution in the random entity list.

6. Entity Weighting Classifier

Previously we have seen the benefit of using weighted dictionaries with click logs [9]. This work further improves dictionary quality using a discriminative model based approach to incorporate features described in Section 3 and 4. For each entity list, an SVM model is built to combine features from the knowledge graph and click logs. Each knowledge graph type is used as an individual feature and the feature value is calculated as in Equation 1. A separate feature is created for click logs, and the feature value is the score computed in Equation 4.

The model training set consists of two parts: the positive samples created from the seed list, and negative samples created from the random list which is extracted from large search query logs. The positive samples are manually labeled; therefore we can assume those samples are of high quality and do indeed belong to the target entity type. However negative samples are automatically extracted from search logs and some of them might belong to the target entity type. To mitigate the noise, we make sure the negative sample list will not contain any entities from seed list or candidate list (i.e. the dictionary to be weighted).

Each sample is converted into a feature vector to train an SVM classifier. Each entity in a candidate list is then weighted using the resulting classifier. For example, an entity classifier for "movie name" is built from a training set of 66K samples (6K positive and 60K negative), and each sample has 1141 features. There are 1140 features for knowledge graph type scores and one feature for click log score. A movie name sample is then converted into a vector as follows:

$$\langle K_e(t_1), K_e(t_2), K_e(t_{1140}), L(e)\rangle$$

These features for the seed and random lists are used to train the binary SVM classifiers. Each entity in a candidate dictionary is weighted and given a score by a classifier created in Section 6. Since dictionaries are used as binary features in our SLU system, weighted dictionaries cannot be used directly. We simply bucket the weighted dictionaries using K-means clustering algorithm so that a weighted dictionary is split into n clusters. Each cluster is used as a smaller dictionary. As a result, the original single dictionary feature is expanded into n features for the intent and slot models. We tested the number of clusters n from 1 to 10, and the best performance results is achieved when n is set to 5. As n increases, some dictionaries are not weighted properly due to data sparseness: the training data does not have enough coverage of the entities in small clusters.

Table 3: Intent detection experiments (error rates %)

Experiments	movie	app	place	game	music
baseline: without dictionaries	6.2	5.0	8.1	9.2	5.8
unweighted dictionaries	5.5	4.6	7.5	7.7	5.5
dics weighted with click logs	5.5	4.5	7.3	7.7	5.4
dics weighted with click logs and knowledge graph	5.1	4.3	7.0	7.5	5.0

7. Experimental Results and Discussions

We built an SLU system for search that covers five domains: movies, apps, places, games and music. For each domain, an intent and a slot model are trained using the training data. Lexical n-gram features and dictionary features are used for model training. The entity weighting methods are evaluated in each domain and are compared for the performance of intent and slot models.

The baseline intent and slot models do not use any dictionaries. Intent and slot models built with the proposed dictionary weighting approach are compared to models with unweighted dictionaries, and models with dictionaries weighted with only click logs which was proposed in [9]. Table 3 shows the experimental results for intent detection models. The intent model performance is measured using intent classification error rates. We can see that the baseline model without using dictionaries is the worst performer in every domain. With unweighted dictionaries the error rates drop by 0.3 to 1.5 points. Models with dictionaries weighted with click logs performs better than unweighted dictionaries. The best performance comes from using proposed weighting approach shown in the last row.

All experiments used the same data sets and N-gram features, the only difference is whether entity dictionaries are utilized and how they are utilized. When a weighted dictionary is used, it is split into 5 smaller dictionaries based on entity weights. So the feature sets used in the last two tests are slightly larger. For example, 20 dictionary features are converted into 100 dictionary features in the movie domain.

Similar experiments are performed on slot tagging tasks. The performance comparison is presented in Table 4. We use F1 scores to measure the accuracy of CRF models. The proposed method in the last row consistently outperforms other approaches. While the improvement for the places domain is insignificant, the improvement in movie domain over the previously proposed dictionary weighting approach [9] is 3%. Also we notice that the models with dictionaries weighted only with click logs perform slightly worse than unweighted dictionaries in movie domain. This is due to the fact there are some legitimate entities missing in search logs and they are clustered into the bucket with the lowest weight.

The proposed method has a larger performance gain in movie and music domains than the other three. The major reason is that candidate dictionaries for these two domains are larger: 56K and 70K entries, respectively. They contain more

Table 4: Slot tagging experiments (F1 measure %)

Experiments	movie	app	place	game	music
baseline: without dictionaries	62.56	79.10	74.67	72.44	67.62
unweighted dictionaries	68.49	80.23	84.47	83.76	70.28
dics weighted with click logs	68.11	80.73	85.42	84.02	71.46
dics weighted with click logs and knowledge graph	71.08	81.41	85.56	84.52	72.51

noisy entities therefore a good weighting approach can boost slot tagging performance.

8. Conclusion and Future Work

We proposed a discriminative model based approach that uses multiple knowledge sources to weight dictionary entities and improve web-scale spoken language understanding systems. Entity weighting classifiers are built using features extracted from a knowledge graph and search click logs. Intent and slot models trained with the proposed weighting approach perform consistently better than the baseline and models built with previous weighting methods. The knowledge graph used in this study is constructed mainly based on public resources such as freebase and wikipedia. Therefore the proposed method can be adopted for similar SLU tasks. Future work in this area includes using additional information sources for entity weighting. For example, the snippet in every search result gives richer information than URL itself. We currently only use type information from knowledge graph, but entity description is also available and can be useful as well.

9. References

[1] G. Tür and R. De Mori, Eds., "Spoken Language Understanding: Systems for Extracting Semantic Information from Speech", New York, NY: John Wiley and Sons, 2011.

[2] N. Gupta, G. Tür, D. Hakkani-Tür, S. Bangalore, G. Riccardi and M. Gilbert, "The AT&T Spoken Language Understanding System", IEEE Trans. Audio, Speech and Language Process. v. 14, no: 1, January, 2006.

[3] A. Deoras, R. Sarikaya, G. Tür, and D. Hakkani-Tür, "Joint Decoding for Speech Recognition and Semantic Tagging", Proc. of InterSpeech, 2012.

[4] A. Deoras, R. Sarikaya, G. Tür, and D. Hakkani-Tür, "Joint Decoding for Speech Recognition and Semantic Tagging", Proc. of InterSpeech, 2012.

[5] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing", Proc. of the ICASSP, Prague, Czech Republic, 2011.

[6] V. Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag, 1995.

[7] R. Sarikaya, H K J Kuo, V. Goel, Y. Gao, "Exploiting unlabeled data using multiple classifiers for improved natural language call-routing", Proc. of InterSpeech, 2005.

[8] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", Proc. of ICML, 2001.

[9] D. Hillard, A. Celikyilmaz, G. Tur, and D. Hakkani-Tür "Learning Weighted Entity Lists from Web Click Logs for Spoken Language Understanding", Proc. of Interspeech 2011.

[10] R. Baeza-Yates and A. Tiberi, "Extracting semantic relations from query logs", Proc. ACM SIGKDD, 2007.

[11] Larry Heck, Dilek Hakkani-Tur, and Gokhan Tur, "Leveraging Knowledge Graphs for Web-Scale Unsupervised Semantic Parsing", Proc. of Interspeech, August 2013

[12] Gokhan Tur, Minwoo Jeong, Ye-Yi Wang, Dilek Hakkani-Tur, and Larry Heck, "Exploiting the Semantic Web for Unsupervised Natural Language Semantic Parsing", Proc. of Interspeech, September 2012

[13] L. Rasmussen, "The Natural Language Interface of Graph Search", Proc. of the ACL, 2013.

[14] M. Mintz and S. Bills and R. Snow and D. Jurafsky, "Distant supervision for relation extraction without labeled data", Proc. of the ACL, 2009.

[15] S. A. McIlraith, T. C. Sun, and H. Zeng, "Semantic web services", IEEE Intelligent Systems, 2001.

[16] N. Shadbolt, W. Hall, and T. Berners-Lee, "The semantic web revisited", IEEE Intelligent Systems, 2006.

[17] R. Guha, R. McCool, and E. Miller, "Semantic search", Proc. of the WWW, Budapest, Hungary, 2003.

[18] P. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning Deep Structured Semantic Models for Web Search using Clickthrough Data", Proc. of CIKM 2013.

[19] H. Tseng, L. Chen, F. Li, Z. Zhuang, L. Duan, and B. Tseng, "Mining Search Engine Clickthrough Log for Matching N-gram Features", Proc. of EMNLP, 2009.