LEXICON MODELING FOR QUERY UNDERSTANDING

Jingjing Liu[1], Xiao Li[2], Alex Acero[2] and Ye-Yi Wang[2]

[1]MIT Computer Science & Artificial Intelligence Laboratory, Cambridge, MA 02139
[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052

ABSTRACT

Lexicons are important resources for semantic tagging. However, commonly used lexicons collected from entity databases suffer from multiple problems, such as ambiguity, limited coverage and lack of relative importance. In this work we present a lexicon modeling technique that automatically expands the lexicon and assigns weights to its elements. For lexicon expansion, we use a generative model to extract patterns from query logs using known lexicon seeds, and discover new lexicon elements using the learned patterns. For lexicon weighting, we propose two approaches based on generative and discriminative models to learn the relative importance of lexicon elements from user click statistics. Experiments on text queries in multiple domains show that our lexicon modeling technique can significantly improve semantic tagging performance.

Index Terms— lexicon modeling, semantic tagging, query understanding

1. INTRODUCTION

Recently, there has been an emergence of personal assistant systems, which can process both spoken and textual queries and help users with customized tasks such as hotel booking or restaurant reservations [1]. In such a system, natural language understanding is unified with keyword query understanding for information retrieval and task completion applications. For example, given a spoken query "show me avatar reviews" or simply a keyword query "avatar review", the system is able to understand that the user's intent is to find reviews of the movie titled "avatar", and to take relevant actions accordingly.

There are three key components required by such a query understanding engine: (1) domain classification; (2) domain-dependent intent detection; and (3) semantic tagging (or slot filling). For example, the query "book me a double room for 2 at Marriott Seattle on Friday" should be classified into the *Hotel* domain with the intent *Book_Hotel*. Furthermore, semantic slots should be extracted as follows: book me a <*RoomType*>double</*RoomType*> room for <*Guest Number*>2</*GuestNumber*> at <*HotelName*>Marriott</*Hotel Name*> <*Location*>Seattle</*Location*> on <*Reservation Date*>Friday</*ReservationDate*>. This paper is concerned with the last component, i.e., segmenting a query into slots and classifying the slots into semantic roles, for both natural language and keyword queries.

There has been a large body of work on semantic tagging in the area of spoken language understanding [2, 3, 4, 5], but relatively few studies concerning keyword query understanding. Only recently, Li et al. [6, 7] investigated both supervised and semi-supervised learning approaches to web query tagging using Markov and semi-Markov conditional random fields (CRFs). In particular, [7] proposed the use of lexical, semantic and syntactic features in semi-Markov CRF based models where lexicon-based semantic features prove to be crucial in enhancing tagging performance.

A lexicon is a dictionary of entities of a certain class, e.g., a list of movie titles. Each element in a lexicon is a surface form of an entity. Lexicons are normally collected from a structured entity database. However, such lexicons have limited coverage when used for query tagging, as the surface forms of entities in user queries are often different from their formal forms in a structured database. For example, the movie entitled "the devil wears prada" is often referred as "devil wearing prada" or "the devil in prada" in user queries. Also, all the elements in a lexicon are treated equally, despite the fact that some surface forms are more popular or ambiguous than others. For example, "Neighbors" is less likely to be used as a restaurant name than "McDonald's" even if they both appear as restaurant names in a database.

There have been some studies on automatic lexicon learning [8, 9, 10]. Pasca and Durme [8, 9] proposed approaches to extracting attribute names (e.g., cost and side effect for the concept Drug) from documents and query logs in a weakly-supervised learning framework. Wang et al. [10] investigated semi-supervised learning algorithms that leverage structured data (HTML lists) from the Web to automatically generate semantic-class lexicon.

In this work, we focus on lexicon modeling for a semi-Markov CRF based semantic tagging system [6]. We propose a generative model that models the probability of a surface form given an entity class. The estimation of such a model, including the acquisition of the surface forms as well as their weights, is performed automatically, leveraging external resources such as query logs. In this way, the modeling of the lexicons is decoupled from the training of semi-Markov CRFs, bringing great flexibility in system update. Furthermore, we investigate a discriminative model where the posteriors of an entity class given surface forms are used as weights. We evaluate the proposed approach on annotated text data, including both natural language and keyword queries, in different domains. Experiments show that the proposed lexicon modeling approach can improve semantic tagging significantly.

2. SEMANTIC TAGGING

This section gives a background of the semantic tagging model we use, which is more fully described in [7]. Assume that the domain of the query is known, and that we have a set of class labels defined for domain-dependent slots. Semantic tagging, then, can be formulated as a joint segmentation and classification problem, *i.e.*,

$$s^* = argmax_s \, p(s|x) \tag{1}$$

Given an input word sequence $x = (x_1, x_2, \ldots, x_M)$, the goal is to find $s = (s_1, s_2, \ldots, s_N)$, which denotes a segmentation of the input as well as a classification of all segments. Each segment is represented by a tuple $s_j = (u_j, v_j, y_j)$. Here u_j and v_j are the start and end indices of the segment, and y_j is a class label. We augment the segment sequence with two special tokens, Start and End, represented by s_0 and s_{N+1} respectively.

As in [7], we use semi-Markov CRF [11] as the segmentation/classification model:

$$p(s|x) = \frac{1}{Z_\lambda(x)} \exp\left\{\textstyle\sum_{j=1}^{N+1} \lambda \cdot f(s_{j-1}, s_j, x)\right\} \tag{2}$$

where the partition function $Z_\lambda(x)$ is a normalization factor; λ is a weight vector; and $f(s_{j-1}, s_j, x)$ is a vector of feature functions defined on segments. More precisely, f is of the function form $f(y_{j-1}, y_j, x, u_j, v_j)$. Given manually-labeled queries, we estimate λ that maximizes the conditional likelihood of training data while regularizing model parameters. The learned model is then used to predict the label sequence s for a future input sequence x.

[7] investigated the use of transition features, lexical features, semantic features and syntactic features in semi-Markov CRFs. An effective way of constructing semantic features is to inspect whether a hypothesized query segment matches any element in a given lexicon. In other words, the feature value is given by:

$$f(s_{j-1}, s_j, x) = \delta(s_j \epsilon L)\delta(y_j = b) \tag{3}$$

where L denotes a lexicon, b denotes a class, and $\delta(s_j \epsilon L)$ denotes that the current segment matches an element in lexicon L, which we refer to as "exact match".

3. LEXICON MODELING

Exact match has limitations, as the surface form of entities in users' queries may not be the same as in the lexicon. For example, a lexicon pre-collected from a restaurant database contains an element "Bamboo Garden Chinese restaurant", while in users' queries it is often referred as "Bamboo Garden" or "Bamboo Garden restaurant", which cannot be discovered using exact match. Also, there might be many restaurant names that users inquire about which are not in the database. In this section, we will explain how to better utilize pre-collected lexicons using fuzzy match, and how to expand lexicons automatically with external resources as well as introducing weights on lexicon elements into model training.

3.1. Fuzzy Match

A natural way of increasing the lexicon coverage is to use fuzzy match features instead of exact match features in Equation (3). In this work, we take as the feature value the maximum "similarity" between a query segment and all lexicon elements. Specifically, we treat each lexicon element as a "document" and compute the *idf* score of each word type accordingly. Let v_{s_j} and v_l denote the *tf-idf* vector of a query segment and that of a lexicon element respectively. The fuzzy match feature value is computed as:

$$f(s_{j-1}, s_j, x) = \max_{l \in L} \frac{v_{s_j} \cdot v_l}{|v_{s_j}||v_l|} \cdot \delta(y_j = b) \tag{4}$$

There is no change required in the original lexicons. But a disadvantage is in computation cost. Exact match is just a one-step table lookup. For fuzzy match, however, a segment has to be compared with each element in the lexicon that has any word overlapping with the segment. Given a segment containing a word that is very common in a lexicon, the computation can be expensive as an online operation.

3.2. Generative Model

A more efficient way of employing semantic features is to obtain an expanded lexicon that has a higher coverage. To automatically learn new lexicons, we leverage external web resources such as query logs (see Table 1 for an example snapshot of a query log). Here, we let y denote the entity class such as *HotelName* and *MovieTitle*, w denote the surface form of an entity, and d_y denote a web document that corresponds to an entity in the entity class y.

In a generative model, we use the normalized log probability of a surface form given a class $p(w|y)$ as the feature value for semi-CRF model training, where

$$p(w|y) = \textstyle\sum_{d_y} p(w|d_y) \cdot p(d_y|y) \tag{5}$$

Here $p(d_y|y)$ represents the popularity of the entity document d_y with respect to the entity class, and $p(w|d_y)$ represents the probability of a surface form given an entity document.

First, to identify the set of documents d_y relevant to the entity class y, we use pre-collected lexicon elements as seed queries and find documents relevant to these queries. We then extract patterns from these documents, which can be regular expressions in URLs or keywords in titles or snippets. Next, we identify new documents d_y that match the learned patterns, and extract unseen surface forms w from queries relevant to these documents as new lexicon candidates.

Take the movie domain as an example. Our task is to expand the *MovieTitle* lexicon which ideally would contain all surface forms of movie titles. We use elements in a lexicon obtained from a movie database as query seeds, and collect documents that users have clicked after issuing the seed queries. As shown in Table 1, the column of "query" is the set of queries that match lexicon seeds. The columns of "title" and "URL" list the titles and URLs of the documents that users clicked on for each query. An example pattern is "www.imdb.com/title" in URLs. Thus, we extract all the URLs from the query log that match the learned pattern (e.g., URLs rooted at "www.imdb.com/title"), and take the queries relevant to these URLs as new lexicon candidates.

Query	Title of clicked doc.	URL of clicked doc.	#
the devil wears prada	The Devil Wears Prada (2006)-Full cast and crew	**http://www.imdb.com/title**/tt0458352/fullcredits	9
the cay	The Cay (1974) (TV) - Memorable quotes	**http://www.imdb.com/title**/tt0246477/quotes	3
the good shepherd	The Good Shepherd (2006)-Full cast and crew	**http://www.imdb.com/title**/tt0343737/fullcredits	11

Table 1. An example snapshot of a query log in the movie domain ("#" denotes the total count of clicks).

Patterns can also be learned from document titles. For example, we desire to expand a lexicon in the restaurant domain that contains restaurant names. Most clicked documents corresponding to such queries contain domain-relevant contextual keywords in their titles (e.g., "restaurant", "steakhouse" and "bar & grill" in Table 2). With known lexicon seeds, we learn the most frequent contextual keywords from the query log, and discover new lexicon candidates from queries that are relevant to the titles that contain these keywords.

Query	Title of clicked doc.	URL of clicked doc.	#
olive garden	Olive Garden Italian **Restaurant** - Zip Locator	http://www.olivegarden.com/locate.asp	8
hyde park	Hyde Park **Bar & Grill** - since 1982 - **Menu**	http://www.hydeparkbarandgrill.com/menu.html	3
silver fox	Silver Fox **Steakhouse** | Richardson **Seafood Restaurant**	http://www.silverfoxcafe.com/richardson_location.php	2

Table 2. Example of query log in the restaurant domain.

In users' queries, lexicons often co-occur with some context words (e.g., "quotes" and "cast" for movies; "coupons" and "menu" for restaurants). To get clean surface forms that do not contain such context words, we use lexicon seeds to learn the most frequent query patterns (e.g., <movie title> *quotes*; <movie title> *cast*), and remove these patterns from the learned lexicon candidates.

Given the cleaned surface forms, the final stage is lexicon weighting, i.e., how to estimate $p(w|y)$. Here we use user click information ("#" in Table 1 and 2) as an indication of relevance. The probability of a relevant document d_y (e.g., a URL or a title) given a class y is defined as the ratio of click count on d_y over the click count on all the documents relevant to the class:

$$p(d_y|y) = \frac{click(d_y)}{\sum_{d_y^*} click(d_y^*)} \qquad (6)$$

The probability of a surface form w given a relevant document d_y is defined as the ratio of click count on d_y triggered by query w over the total count of clicks on d_y triggered by all the relevant queries:

$$p(w|d_y) = \frac{click(w,d_y)}{click(d_y)} \qquad (7)$$

3.3. Discriminative Model

A limitation of generative models is that they fail to reflect ambiguity. A lexicon element with high likelihood score may be very confusable with other entity types. For example, "Paris" is a popular hotel in Las Vegas, but it's highly confusable with the lexicon of *CityName*.

Such a problem can be potentially tackled by a discriminative model. Here we intend to change the feature value while still using the lexicon expanded with the generative model approach. In the discriminative model, the normalized log posterior probability *p(y|w)* is used as the feature value:

$$p(y|w) = \sum_{d_w} p(y|d_w) \cdot p(d_w|w) \qquad (8)$$

where d_w is a web document relevant to the surface form w; $p(d_w|w)$ is a query-document relevance model; and $p(y|d_w)$ is a context document classification model, representing the probability that d_w belongs to class y.

Specifically, to estimate $p(d_w|w)$, we obtain a set of documents d_w relevant to w. We view each lexicon candidate w as a query and retrieve the snippets of the top-*n* ranked documents from a search engine as d_w. To estimate $p(y|d_w)$, we use a set of known lexicon elements as query seeds, use the top-*n* snippets of documents retrieved by each query as positive samples (see examples in Table 3), and use the top-*n* snippets retrieved by a set of domain-irrelevant lexicons (e.g., *CityName*) as negative examples. Then we train a MaxEnt entropy classifier with these examples, using *n*-grams in the document snippets as features. The learned classifier is then applied to the top-*n* documents retrieved by each new lexicon candidate, and the posterior classification score is used as $p(y|d_w)$.

Query	Snippet of clicked document
bourne identity	The Bourne Identity is an new script for the fourth installment in the Bourne series More August 07 2009 Doug Liman Is a Real Life Action Hero Bourne Identity director
dances with wolves	Dances with Wolves is a 1990 epic film based on the book of the same name which tells the story of a Civil War era United States Army lieutenant who travels to the American frontier

Table 3. Query-snippet examples in the movie domain.

4. EXPERIMENTS

We conducted experiments on textual queries formulated in both natural language and keywords in three domains: *Restaurant*, *Hotel* and *Movie* (as in Table 4). We asked human annotators to manually label the data. The annotators independently segmented each query into slots and assigned each slot a semantic class label selected from Table 5. Segments that do not belong to any of the semantic classes are assigned an "*Other*" label.

Domain	Training set		Test set	
	#Queries	#Slots	#Queries	#Slots
Restaurant	2340	5269	601	1331
Hotel	1572	3729	406	992
Movies	1768	2257	540	654

Table 4. Statistics on training/test sets in three domains.

Domain	Semantic classes
Restaurant	cuisine, restaurant type, amenities, menu item, restaurant name, described as, location, opening hour, star rating, price range, reservation date, reservation time, reservation party size, meal type
Hotel	hotel type, hotel name, location, room type, adult number, child number, reward program, smoking, checkin date, checkout date, nights, number of rooms, star rating, described as, price range, amenities
Movie	movie type, character, award, movie name, location, theater, date, release date, time, star rating, mpaa rating, genre, nationality, director, review site, year, language, star, number of tickets

Table 5. Semantic classes defined for each domain.

The evaluation metrics are precision, recall and F1 at the slot level, excluding "*Other*" slots. A slot is considered as identified correctly if and only if it is segmented correctly and tagged with the same label as annotated. We used a semi-Markov CRF model as our baseline. Transition features, lexical features and semantic features were used for model training [7]. For semantic features, we used exact match on baseline lexicons, which were obtained from databases for hotels, restaurants, and movies. To implement our lexicon modeling approach, we used query logs collected over a year from Bing. The lexicons we applied our approaches to are *HotelName* (90k entries), *RestaurantName* (500k entries) and *MovieTitle* (120k entries). For lexicon expansion, we extracted lexicon candidates using the pattern "www.imdb.com/title" from URLs in query log for the *MovieTitle* entity; we also extracted lexicon candidates from titles using 34 most frequent keywords (e.g., "bed and breakfast") in the hotel domain for *HotelName* and 45 keywords (e.g., "steakhouse") in the restaurant domain for *RestaurantName*. For discriminative models, we employed a maximum entropy classifier as the context classification model and used *n*-grams from the training data as features.

Experimental results on generative and discriminative models are given in Table 6 ("*P*" is precision and "*R*" is recall, both presented in percentage). "*BS*" denotes the baseline which uses exact match on the baseline lexicons. "*FM*" denotes fuzzy match on baseline lexicons, and "*LE*" denotes lexicon expansion, i.e., exact match on the expanded lexicon learned from the generative model. "*GLW*" denotes lexicon weighting by generative models and "*DLW*" denotes lexicon weighting by discriminative models, both using the same expanded lexicons. And "*FME*" denotes fuzzy match on expanded lexicons.

Feature	**Hotel**			**Restaurant**			**Movie**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
BS	86.3	87.3	86.8	85.8	88.1	86.9	78.8	76.9	77.9
BS+FM	87.5	87.6	87.6	85.9	88.4	87.2	83.5	79.5	81.4
BS+LE	87.2	87.4	87.3	85.6	88.2	86.8	81.2	78.4	79.8
BS+LE+GLW	87.4	87.6	87.5	86.0	88.5	87.2	81.1	78.9	80.0
BS+LE+DLW	86.5	87.4	87.0	85.9	88.4	87.1	80.7	78.0	79.3
BS+FM+LE	88.7	87.9	88.3	**86.4**	**88.9**	**87.6**	84.1	80.7	82.4
BS+FM+LE+GLW	89.4	88.6	89.0	**86.7**	**89.1**	**87.9**	84.7	81.2	82.9
BS+FM+LE+DLW	88.4	88.2	88.3	86.3	88.6	87.4	83.8	80.7	82.2
BS+FM+LE+GLW+FME	**89.9**	**88.9**	**89.4**	86.0	88.4	87.2	**84.6**	**81.7**	**83.1**
BS+FM+LE+DLW+FME	**89.9**	**89.0**	**89.5**	86.3	88.6	87.4	**84.0**	**81.2**	**82.6**

Table 6. Semantic tagging performance using different feature sets.

Experiments show that, in the hotel and movie domains, each technique (*FM*, *LE*, and *LW*) helps to improve the baseline, and the best performance is obtained by combining all techniques in both generative and discriminative models. In the restaurant domain, the best performance on generative models is without FME and that on discriminative model is without FME and LW. The relative improvements in the movie and hotel domains are more than that in the restaurant domain. This is due to the coverage of pre-collected lexicons. In our data, the size of the pre-collected lexicon in the restaurant domain (500k) is much larger than that in the hotel and movie domains (90-120k). As it is a lexicon-expansion-based approach, the improvement over a smaller pre-collected lexicon is expected to be more significant than that over a larger pre-collected lexicon.

The performance obtained using lexicon weights learned from generative models is comparable with that from discriminative models. This shows that taking popularity and ambiguity into account in lexicon weighting both helped semantic tagging, although no one model seems to be significantly better than the other. In future work, we will explore how to combine these two types of features for better performance.

5. CONCLUSIONS

In this work, we proposed a lexicon modeling approach for semantic tagging. We utilized external resources such as web search query logs for automatic lexicon discovery and lexicon weighting with a generative model. We further investigated discriminative models for lexicon weighting that take entity ambiguity into account. Experiments on textual queries in multiple domains show that the proposed approach can improve the performance of semantic tagging significantly. Potentially the same approach can be applied to spoken queries given reliable speech recognition. For future work, we will apply the lexicon modeling approach to larger datasets. We will also explore the use of other external resources such as Wikipedia for automatic lexicon learning and weighting.

6. REFERENCES

[1] http://www.siri.com/. *S*iri - Your Virtual Personal Assistant**.**

[2] Dilek Hakkani-Tr, Gokhan Tur, Tutorial on Spoken Language Understanding. In *ICASSP'07*.

[3] Ye-Yi Wang, Li Deng, and Alex Acero. 2005. Spoken Language Understanding — An Introduction to the Statistical Framework, *in IEEE Signal Processing Magazine*.

[4] Stefan Hahn, Patrick Lehnen, Christian Raymond, and Hermann Ney. 2008. A Comparison of Various Methods for Concept Tagging for Spoken Language Understanding. In *Proceedings of LREC'08*.

[5] Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Re-Ranking Models For Spoken Language Understanding. In *Proceedings of EACL'09*.

[6] Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. *In Proceedings of SIGIR' 09*.

[7] Xiao Li. 2010. Understanding the Semantic Structure of Noun Phrase Queries. In *Proceedings of Association for Computational Linguistics, 2010.*

[8] Marius Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *Proceedings of IJCAI'07*.

[9] Marius Pasca and Benjamin Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-HLT*.

[10] Ye-Yi Wang, Raphael Hoffmann, Xiao Li and Jakub Szymanski. 2009. Semi-Supervised Learning of Semantic Classes for Query. Understanding – from the Web and for the Web. In *Proceedings of CIKM'09*.

[11] Sunita Sarawagi and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems (NIPS'04)*.