Understanding the Semantic Structure of Noun Phrase Queries

Xiao Li
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
xiaol@microsoft.com

Abstract

Determining the semantic intent of web queries not only involves identifying their semantic class, which is a primary focus of previous works, but also understanding their semantic structure. In this work, we formally define the semantic structure of noun phrase queries as comprised of *intent heads* and *intent modifiers*. We present methods that automatically identify these constituents as well as their semantic roles based on Markov and semi-Markov conditional random fields. We show that the use of semantic features and syntactic features significantly contribute to improving the understanding performance.

1 Introduction

Web queries can be considered as implicit questions or commands, in that they are performed either to find information on the web or to initiate interaction with web services. Web users, however, rarely express their intent in full language. For example, to find out "what are the movies of 2010 in which johnny depp stars", a user may simply query "johnny depp movies 2010". Today's search engines, generally speaking, are based on matching such keywords against web documents and ranking relevant results using sophisticated features and algorithms.

As search engine technologies evolve, it is increasingly believed that search will be shifting away from "ten blue links" toward understanding intent and serving objects. This trend has been largely driven by an increasing amount of structured and semi-structured data made available to search engines, such as relational databases and semantically annotated web documents. Searching over such data sources, in many cases, can offer more relevant and essential results compared with merely returning web pages that contain query keywords. Table 1 shows a simplified view of a structured data source, where each row represents a movie object. Consider the query "johnny depp movies 2010". It is possible to retrieve a set of movie objects from Table 1 that satisfy the constraints *Year = 2010* and *Cast* $\ni$ *Johnny Depp*. This would deliver direct answers to the query rather than having the user sort through list of keyword results.

In no small part, the success of such an approach relies on robust understanding of query intent. Most previous works in this area focus on query intent classification (Shen et al., 2006; Li et al., 2008b; Arguello et al., 2009). Indeed, the intent class information is crucial in determining if a query can be answered by any structured data sources and, if so, by which one. In this work, we go one step further and study the semantic structure of a query, *i.e.*, individual constituents of a query and their semantic roles. In particular, we focus on *noun phrase* queries. A key contribution of this work is that we formally define query semantic structure as comprised of *intent heads* (IH) and *intent modifiers* (IM), *e.g.*,

$$[_{\text{IM:}\textit{Title}}\ \text{alice in wonderland}]\ [_{\text{IM:}\textit{Year}}\ 2010]\ [_{\text{IH}}\ \text{cast}]$$

It is determined that "cast" is an IH of the above query, representing the essential information the user intends to obtain. Furthermore, there are two IMs, "alice in wonderland" and "2010", serving as filters of the information the user receives.

Identifying the semantic structure of queries can be beneficial to information retrieval. Knowing the semantic role of each query constituent, we

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1337–1345,
Uppsala, Sweden, 11-16 July 2010.

Title	Year	Genre	Director	Cast	Review
Precious	2009	Drama	Lee Daniels	Gabby Sidibe, Mo'Nique,...	
2012	2009	Action, Sci Fi	Roland Emmerich	John Cusack, Chiwetel Ejiofor,...	
Avatar	2009	Action, Sci Fi	James Cameron	Sam Worthington, Zoe Saldana,...	
The Rum Diary	2010	Adventure, Drama	Bruce Robinson	Johnny Depp,Giovanni Ribisi,...	
Alice in Wonderland	2010	Adventure, Family	Tim Burton	Mia Wasikowska, Johnny Depp,...	

Table 1: A simplified view of a structured data source for the *Movie* domain.

can reformulate the query into a structured form or reweight different query constituents for structured data retrieval (Robertson et al., 2004; Kim et al., 2009; Paparizos et al., 2009). Alternatively, the knowledge of IHs, IMs and semantic labels of IMs may be used as additional evidence in a learning to rank framework (Burges et al., 2005).

A second contribution of this work is to present methods that automatically extract the semantic structure of noun phrase queries, *i.e.*, IHs, IMs and the semantic labels of IMs. In particular, we investigate the use of transition, lexical, semantic and syntactic features. The semantic features can be constructed from structured data sources or by mining query logs, while the syntactic features can be obtained by readily-available syntactic analysis tools. We compare the roles of these features in two discriminative models, Markov and semi-Markov conditional random fields. The second model is especially interesting to us since in our task it is beneficial to use features that measure segment-level characteristics. Finally, we evaluate our proposed models and features on manually-annotated query sets from three domains, while our techniques are general enough to be applied to many other domains.

2 Related Works

2.1 Query intent understanding

As mentioned in the introduction, previous works on query intent understanding have largely focused on classification, *i.e.*, automatically mapping queries into semantic classes (Shen et al., 2006; Li et al., 2008b; Arguello et al., 2009). There are relatively few published works on understanding the semantic structure of web queries. The most relevant ones are on the problem of query tagging, *i.e.*, assigning semantic labels to query terms (Li et al., 2009; Manshadi and Li, 2009). For example, in "canon powershot sd850 camera silver", the word "canon" should be tagged as *Brand*. In particular, Li et al. leveraged click-through data and a database to automatically derive training data for learning a CRF-based tagger. Manshadi and Li developed a hybrid, generative grammar model for a similar task. Both works are closely related to one aspect of our work, which is to assign semantic labels to IMs. A key difference is that they do not conceptually distinguish between IHs and IMs.

On the other hand, there have been a series of research studies related to IH identification (Pasca and Durme, 2007; Pasca and Durme, 2008). Their methods aim at extracting attribute names, such as *cost* and *side effect* for the concept *Drug*, from documents and query logs in a weakly-supervised learning framework. When used in the context of web queries, attribute names usually serve as IHs. In fact, one immediate application of their research is to understand web queries that request factual information of some concepts, *e.g.* "asiprin cost" and "aspirin side effect". Their framework, however, does not consider the identification and categorization of IMs (attribute values).

2.2 Question answering

Query intent understanding is analogous to question understanding for *question answering* (QA) systems. Many web queries can be viewed as the keyword-based counterparts of natural language questions. For example, the query "california national" and "national parks califorina" both imply the question "What are the national parks in California?". In particular, a number of works investigated the importance of *head noun* extraction in understanding *what-type* questions (Metzler and Croft, 2005; Li et al., 2008a). To extract head nouns, they applied syntax-based rules using the information obtained from part-of-speech (POS) tagging and deep parsing. As questions posed in natural language tend to have strong syntactic structures, such an approach was demonstrated to be accurate in identifying head nouns.

In identifying IHs in noun phrase queries, however, direct syntactic analysis is unlikely to be as effective. This is because syntactic structures are in general less pronounced in web queries. In this

work, we propose to use POS tagging and parsing outputs as features, in addition to other features, in extracting the semantic structure of web queries.

2.3 Information extraction

Finally, there exist large bodies of work on information extraction using models based on Markov and semi-Markov CRFs (Lafferty et al., 2001; Sarawagi and Cohen, 2004), and in particular for the task of named entity recognition (McCallum and Li, 2003).

The problem studied in this work is concerned with identifying more generic "semantic roles" of the constituents in noun phrase queries. While some IM categories belong to named entities such as IM:*Director* for the intent class *Movie*, there can be semantic labels that are not named entities such as IH and IM:*Genre* (again for *Movie*).

3 Query Semantic Structure

Unlike database query languages such as SQL, web queries are usually formulated as sequences of words without explicit structures. This makes web queries difficult to interpret by computers. For example, should the query "aspirin side effect" be interpreted as "the side effect of aspirin" or "the aspirin of side effect"? Before trying to build models that can automatically makes such decisions, we first need to understand what constitute the semantic structure of a noun phrase query.

3.1 Definition

We let $\mathcal{C}$ denote a set of query intent classes that represent semantic concepts such as *Movie*, *Product* and *Drug*. The query constituents introduced below are all defined w.r.t. the intent class of a query, $c \in \mathcal{C}$, which is assumed to be known.

Intent head

An *intent head* (IH) is a query segment that corresponds to an **attribute name** of an intent class. For example, the IH of the query "alice in wonderland 2010 cast" is "cast", which is an attribute name of *Movie*. By issuing the query, the user intends to find out the values of the IH (*i.e.*, cast). A query can have multiple IHs, *e.g.*, "movie avatar director and cast". More importantly, there can be queries without an explicit IH. For example, "movie avatar" does not contain any segment that corresponds to an attribute name of *Movie*. Such a query, however, does have an implicit intent which is to obtain general information about the movie.

Intent modifier

In contrast, an *intent modifier* (IM) is a query segment that corresponds to an **attribute value** (of some attribute name). The role of IMs is to imposing constraints on the attributes of an intent class. For example, there are two constraints implied in the query "alice in wonderland 2010 cast": (1) the *Title* of the movie is "alice in wonderland"; and (2) the *Year* of the movie is "2010". Interestingly, the user does not explicitly specify the attribute names, *i.e.*, *Title* and *Year*, in this query. Such information, however, can be inferred given domain knowledge. In fact, one important goal of this work is to identify the semantic labels of IMs, *i.e.*, the attribute names they implicitly refer to. We use $\mathcal{A}_c$ to denote the set of IM semantic labels for the intent class c.

Other

Additionally, there can be query segments that do not play any semantic roles, which we refer to as Other.

3.2 Syntactic analysis

The notion of IHs and IMs in this work is closely related to that of linguistic *head nouns* and *modifiers* for noun phrases. In many cases, the IHs of noun phrase queries are exactly the head nouns in the linguistic sense. Exceptions mostly occur in queries without explicit IHs, *e.g.*, "movie avatar" in which the head noun "avatar" serves as an IM instead. Due to the strong resemblance, it is interesting to see if IHs can be identified by extracting linguistic head nouns from queries based on syntactic analysis. To this end, we apply the following heuristics for head noun extraction. We first run a POS-tagger and a chunker jointly on each query, where the POS-tagger/chunker is based on an HMM system trained on English Penn Treebank (Gao et al., 2001). We then mark the right most NP chunk before any prepositional phrase or adjective clause, and apply the NP head rules (Collins, 1999) to the marked NP chunk.

The main problem with this approach, however, is that a readily-available POS tagger or chunker is usually trained on natural language sentences and thus is unlikely to produce accurate results on web queries. As shown in (Barr et al., 2008), the lexical category distribution of web queries is dramatically different from that of natural languages. For example, prepositions and subordinating conjunctions, which are strong indicators of the syntactic

structure in natural languages, are often missing in web queries. Moreover, unlike most natural languages that follow the linear-order principle, web queries can have relatively free word orders (although some orders may occur more often than others statistically). These factors make it difficult to produce reliable syntactic analysis outputs. Consequently, the head nouns and hence the IHs extracted therefrom are likely to be error-prone, as will be shown by our experiments in Section 6.3.

Although a POS tagger and a chunker may not work well on queries, their output can be used as features for learning statistical models for semantic structure extraction, which we introduce next.

4 Models

This section presents two statistical models for semantic understanding of noun phrase queries. Assuming that the intent class $c \in \mathcal{C}$ of a query is known, we cast the problem of extracting the semantic structure of the query into a joint segmentation/classification problem. At a high level, we would like to identify query segments that correspond to IHs, IMs and Others. Furthermore, for each IM segment, we would like to assign a semantic label, denoted by IM:a, $a \in \mathcal{A}_c$, indicating which attribute name it refers to. In other words, our label set consists of $\mathcal{Y} = \{\text{IH}, \{\text{IM:}a\}_{a \in \mathcal{A}_c}, \text{Other}\}$.

Formally, we let $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ denote an input query of length M. To avoid confusion, we use i to represent the index of a word token and j to represent the index of a segment in the following text. Our goal is to obtain

$$\mathbf{s}^* = \operatorname*{argmax}_{\mathbf{s}} p(\mathbf{s}|c, \mathbf{x}) \tag{1}$$

where $\mathbf{s} = (s_1, s_2, \ldots, s_N)$ denotes a query segmentation as well as a classification of all segments. Each segment s_j is represented by a tuple (u_j, v_j, y_j). Here u_j and v_j are the indices of the starting and ending word tokens respectively; $y_j \in \mathcal{Y}$ is a label indicating the semantic role of s. We further augment the segment sequence with two special segments: *Start* and *End*, represented by s_0 and s_{N+1} respectively. For notional simplicity, we assume that the intent class is given and use $p(\mathbf{s}|\mathbf{x})$ as a shorthand for $p(\mathbf{s}|c, \mathbf{x})$, but keep in mind that the label space and hence the parameter space is class-dependent. Now we introduce two methods of modeling $p(\mathbf{s}|\mathbf{x})$.

4.1 CRFs

One natural approach to extracting the semantic structure of queries is to use linear-chain CRFs (Lafferty et al., 2001). They model the conditional probability of a label sequence given the input, where the labels, denoted as $\mathbf{y} = (y_1, y_2, \ldots, y_M)$, $y_i \in \mathcal{Y}$, have a one-to-one correspondence with the word tokens in the input.

Using linear-chain CRFs, we aim to find the label sequence that maximizes

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp\left\{\sum_{i=1}^{M+1} \lambda \cdot f(y_{i-1}, y_i, \mathbf{x}, i)\right\}. \tag{2}$$

The partition function $Z_\lambda(\mathbf{x})$ is a normalization factor. λ is a weight vector and $f(y_{i-1}, y_i, \mathbf{x})$ is a vector of feature functions referred to as a feature vector. The features used in CRFs will be described in Section 5.

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 $\mathbf{y}$ for future input sequences $\mathbf{x}$. To obtain $\mathbf{s}$ in Equation (1), we simply concatenate the maximum number of consecutive word tokens that have the same label and treat the resulting sequence as a segment. By doing this, we implicitly assume that there are no two adjacent segments with the same label in the true segment sequence. Although this assumption is not always correct in practice, we consider it a reasonable approximation given what we empirically observed in our training data.

4.2 Semi-Markov CRFs

In contrast to standard CRFs, semi-Markov CRFs directly model the segmentation of an input sequence as well as a classification of the segments (Sarawagi and Cohen, 2004), *i.e.*,

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp \sum_{j=1}^{N+1} \lambda \cdot f(s_{j-1}, s_j, \mathbf{x}) \tag{3}$$

In this case, the features $f(s_{j-1}, s_j, \mathbf{x})$ are defined on segments instead of on word tokens. More precisely, they are of the function form $f(y_{j-1}, y_j, \mathbf{x}, u_j, v_j)$. It is easy to see that by imposing a constraint $u_i = v_i$, the model is reduced to standard linear-chain CRFs. Semi-Markov CRFs make Markov assumptions at the segment level, thereby naturally offering means to

CRF features		
A1: Transition	$\delta(y_{i-1} = a)\delta(y_i = b)$	transiting from state a to b
A2: Lexical	$\delta(x_i = w)\delta(y_i = b)$	current word is w
A3: Semantic	$\delta(x_i \in \mathcal{W}_\mathcal{L})\delta(y_i = b)$	current word occurs in lexicon $\mathcal{L}$
A4: Semantic	$\delta(\mathbf{x}_{i-1:i} \in \mathcal{W}_\mathcal{L})\delta(y_i = b)$	current bigram occurs in lexicon $\mathcal{L}$
A5: Syntactic	$\delta(\text{POS}(\mathbf{x}_i) = z)\delta(y_i = b)$	POS tag of the current word is z
Semi-Markov CRF features		
B1: Transition	$\delta(y_{j-1} = a)\delta(y_j = b)$	Transiting from state a to b
B2: Lexical	$\delta(\mathbf{x}_{u_j:v_j} = \mathbf{w})\delta(y_j = b)$	Current segment is $\mathbf{w}$
B3: Lexical	$\delta(\mathbf{x}_{u_j:v_j} \ni w)\delta(y_j = b)$	Current segment contains word w
B4: Semantic	$\delta(\mathbf{x}_{u_j:v_j} \in \mathcal{L})\delta(y_j = b)$	Current segment is an element in lexicon $\mathcal{L}$
B5: Semantic	$\max_{l \in \mathcal{L}} s(\mathbf{x}_{u_j:v_j}, l)\delta(y_j = b)$	The max similarity between the segment and elements in $\mathcal{L}$
B6: Syntactic	$\delta(\text{POS}(\mathbf{x}_{u_j:v_j}) = \mathbf{z})\delta(y_j = b)$	Current segment's POS sequence is $\mathbf{z}$
B7: Syntactic	$\delta(\text{Chunk}(\mathbf{x}_{u_j:v_j}) = c)\delta(y_j = b)$	Current segment is a chunk with phrase type c

Table 2: A summary of feature types in CRFs and segmental CRFs for query understanding. We assume that the state label is b in all features and omit this in the feature descriptions.

incorporate segment-level features, as will be presented in Section 5.

5 Features

In this work, we explore the use of transition, lexical, semantic and syntactic features in Markov and semi-Markov CRFs. The mathematical expression of these features are summarized in Table 2 with details described as follows.

5.1 Transition features

Transition features, *i.e.*, A1 and B1 in Table 2, capture state transition patterns between adjacent word tokens in CRFs, and between adjacent segments in semi-Markov CRFs. We only use first-order transition features in this work.

5.2 Lexical features

In CRFs, a lexical feature (A2) is implemented as a binary function that indicates whether a specific word co-occurs with a state label. The set of words to be considered in this work are those observed in the training data. We can also generalize this type of features from words to n-grams. In other words, instead of inspecting the word identity at the current position, we inspect the n-gram identity by applying a window of length n centered at the current position.

Since feature functions are defined on segments in semi-Markov CRFs, we create B2 that indicates whether the phrase in a hypothesized query segment co-occurs with a state label. Here the set of phrase identities are extracted from the query segments in the training data. Furthermore, we create another type of lexical feature, B3, which is activated when a specific word occurs in a hypothesized query segment. The use of B3 would favor unseen words being included in adjacent segments rather than to be isolated as separate segments.

5.3 Semantic features

Models relying on lexical features may require very large amounts of training data to produce accurate prediction performance, as the feature space is in general large and sparse. To make our model generalize better, we create semantic features based on what we call *lexicons*. A lexicon, denoted as $\mathcal{L}$, is a cluster of semantically-related words/phrases. For example, a cluster of movie titles or director names can be such a lexicon. Before describing how such lexicons are generated for our task, we first introduce the forms of the semantic features assuming the availability of the lexicons.

We let $\mathcal{L}$ denote a lexicon, and $\mathcal{W}_\mathcal{L}$ denote the set of n-grams extracted from $\mathcal{L}$. For CRFs, we create a binary function that indicates whether *any* n-gram in $\mathcal{W}_\mathcal{L}$ co-occurs with a state label, with $n = 1, 2$ for A3, A4 respectively. For both A3 and A4, the number of such semantic features is equal to the number of lexicons multiplied by the number of state labels.

The same source of semantic knowledge can be conveniently incorporated in semi-Markov CRFs. One set of semantic features (B4) inspect whether the phrase of a hypothesized query segment matches *any* element in a given lexicon. A second set of semantic features (B5) relax the exact match constraints made by B4, and take as the feature value the maximum "similarity" between the query segment and *all* lexicon elements. The fol-

lowing similarity function is used in this work ,

$$s(\mathbf{x}_{u_j:v_j}, l) = 1 - \text{Lev}(\mathbf{x}_{u_j:v_j}, l)/|l| \quad (4)$$

where Lev represents the Levenshtein distance. Notice that we normalize the Levenshtein distance by the length of the lexicon element, as we empirically found it performing better compared with normalizing by the length of the segment. In computing the maximum similarity, we first retrieve a set of lexicon elements with a positive *tf-idf* cosine distance with the segment; we then evaluate Equation (4) for each retrieved element and find the one with the maximum similarity score.

Lexicon generation

To create the semantic features described above, we generate two types of lexicons leveraging databases and query logs for each intent class.

The first type of lexicon is an IH lexicon comprised of a list of attribute names for the intent class, *e.g.*, "box office" and "review" for the intent class *Movie*. One easy way of composing such a list is by aggregating the column names in the corresponding database such as Table 1. However, this approach may result in low coverage on IHs for some domains. Moreover, many database column names, such as *Title*, are unlikely to appear as IHs in queries. Inspired by Pasca and Van Durme (2007), we apply a bootstrapping algorithm that automatically learns attribute names for an intent class from query logs. The key difference from their work is that we create templates that consist of semantic labels at the segment level from training data. For example, "alice in wonderland 2010 cast" is labeled as "IM:*Title* IM:*Year* IH", and thus "IM:*Title* + IM:*Year* + #" is used as a template. We select the most frequent templates (top 2 in this work) from training data and use them to discover new IH phrases from the query log.

Secondly, we have a set IM lexicons, each comprised of a list of attribute values of an attribute name in $\mathcal{A}_c$. We exploit internal resources to generate such lexicons. For example, the lexicon for IM:*Title* (in *Movie*) is a list of movie titles generated by aggregating the values in the *Title* column of a movie database. Similarly, the lexicon for IM:*Employee* (in *Job*) is a list of employee names extracted from a job listing database. Note that a substantial amount of research effort has been dedicated to automatic lexicon acquisition from the Web (Pantel and Pennacchiotti, 2006; Pennacchiotti and Pantel, 2009). These techniques can be used in expanding the semantic lexicons for IMs when database resources are not available. But we do not use such techniques in our work since the lexicons extracted from databases in general have good precision and coverage.

5.4 Syntactic features

As mentioned in Section 3.2, web queries often lack syntactic cues and do not necessarily follow the linear order principle. Consequently, applying syntactic analysis such as POS tagging or chunking using models trained on natural language corpora is unlikely to give accurate results on web queries, as supported by our experimental evidence in Section 6.3. It may be beneficial, however, to use syntactic analysis results as additional evidence in learning.

To this end, we generate a sequence of POS tags for a given query, and use the co-occurrence of POS tag identities and state labels as syntactic features (A5) for CRFs.

For semi-Markov CRFs, we instead examine the POS tag sequence of the corresponding phrase in a query segment. Again their identities are combined with state labels to create syntactic features B6. Furthermore, since it is natural to incorporate segment-level features in semi-Markov CRFs, we can directly use the output of a syntactic chunker. To be precise, if a query segment is determined by the chunker to be a chunk, we use the indicator of the phrase type of the chunk (*e.g.*, NP, PP) combined with a state label as the feature, denoted by B7 in the Table. Such features are not activated if a query segment is determined not to be a chunk.

6 Evaluation

6.1 Data

To evaluate our proposed models and features, we collected queries from three domains, *Movie*, *Job* and *National Park*, and had them manually annotated. The annotation was given on both segmentation of the queries and classification of the segments according to the label sets defined in Table 3. There are 1000/496 samples in the training/test set for the *Movie* domain, 600/366 for the *Job* domain and 491/185 for the *National Park* domain. In evaluation, we report the test-set performance in each domain as well as the average performance (weighted by their respectively test-set size) over all domains.

Movie		Job		National Park	
IH	trailer, box office	IH	listing, salary	IH	lodging, calendar
IM:*Award*	oscar best picture	IM:*Category*	engineering	IM:*Category*	national forest
IM:*Cast*	johnny depp	IM:*City*	las vegas	IM:*City*	page
IM:*Character*	michael corleone	IM:*County*	orange	IM:*Country*	us
IM:*Category*	tv series	IM:*Employer*	walmart	IM:*Name*	yosemite
IM:*Country*	american	IM:*Level*	entry level	IM:*POI*	volcano
IM:*Director*	steven spielberg	IM:*Salary*	high-paying	IM:*Rating*	best
IM:*Genre*	action	IM:*State*	florida	IM:*State*	flordia
IM:*Rating*	best	IM:*Type*	full time		
IM:*Title*	the godfather				
Other	the, in, that	Other	the, in, that	Other	the, in, that

Table 3: Label sets and their respective query segment examples for the intent class *Movie*, *Job* and *National Park*.

6.2 Metrics

There are two evaluation metrics used in our work: segment F1 and sentence accuracy (Acc). The first metric is computed based on precision and recall at the segment level. Specifically, let us assume that the true segment sequence of a query is $\mathbf{s} = (s_1, s_2, \ldots, s_N)$, and the decoded segment sequence is $\mathbf{s}' = (s'_1, s'_2, \ldots, s'_K)$. We say that s'_k is a true positive if $s'_k \in \mathbf{s}$. The precision and recall, then, are measured as the total number of true positives divided by the total number of decoded and true segments respectively. We report the F1-measure which is computed as $2 \cdot \text{prec} \cdot \text{recall}/(\text{prec} + \text{recall})$.

Secondly, a sentence is correct if all decoded segments are true positives. Sentence accuracy is measured by the total number of correct sentences divided by the total number of sentences.

6.3 Results

We start with models that incorporate first-order transition features which are standard for both Markov and semi-Markov CRFs. We then experiment with lexical features, semantic features and syntactic features for both models. Table 4 and Table 5 give a summarization of all experimental results.

Lexical features

The first experiment we did is to evaluate the performance of lexical features (combined with transition features). This involves the use of A2 in Table 2 for CRFs, and B2 and B3 for semi-Markov CRFs. Note that adding B3, *i.e.,* indicators of whether a query segment contains a word identity, gave an absolute 7.0%/3.2% gain in sentence accuracy and segment F1 on average, as shown in the row B1-B3 in Table 5. For both A2 and B3, we also tried extending the features based on word IDs to those based on n-gram IDs, where $n = 1, 2, 3$. This greatly increased the number of lexical features but did not improve learning performance, most likely due to the limited amounts of training data coupled with the sparsity of such features. In general, lexical features do not generalize well to the test data, which accounts for the relatively poor performance of both models.

Semantic features

We created IM lexicons from three in-house databases on *Movie*, *Job* and *National Parks*. Some lexicons, *e.g.*, IM:*State*, are shared across domains. Regarding IH lexicons, we applied the bootstrapping algorithm described in Section 5.3 to a 1-month query log of *Bing*. We selected the most frequent 57 and 131 phrases to form the IH lexicons for *Movie* and *National Park* respectively. We do not have an IH lexicon for *Job* as the attribute names in that domain are much fewer and are well covered by training set examples.

We implemented A3 and A4 for CRFs, which are based on the n-gram sets created from lexicons; and B4 and B5 for semi-Markov CRFs, which are based on exact and fuzzy match with lexicon items. As shown in Table 4 and 5, drastic increases in sentence accuracies and F1-measures were observed for both models.

Syntactic features

As shown in the row A1-A5 in Table 4, combined with all other features, the syntactic features (A5) built upon POS tags boosted the CRF model performance. Table 6 listed the most dominant positive and negative features based on POS tags for *Movie* (features for the other two domains are not reported due to space limit). We can see that many of these features make intuitive sense. For

	Movie		Job		National Park		Average	
Features	Acc	F1	Acc	F1	Acc	F1	Acc	F1
A1,A2: Tran + Lex	59.9	75.8	65.6	84.7	61.6	75.6	62.1	78.9
A1-A3: Tran + Lex + Sem	67.9	80.2	70.8	87.4	70.5	80.8	69.4	82.8
A1-A4: Tran + Lex + Sem	72.4	83.5	72.4	89.7	71.1	82.3	72.2	85.0
A1-A5: Tran + Lex + Sem + Syn	74.4	84.8	75.1	89.4	75.1	85.4	74.8	86.5
A2-A5: Lex + Sem + Syn	64.9	78.8	68.1	81.1	64.8	83.7	65.4	81.0

Table 4: Sentence accuracy (Acc) and segment F1 (F1) using CRFs with different features.

	Movie		Job		National Park		Average	
Features	Acc	F1	Acc	F1	Acc	F1	Acc	F1
B1,B2: Tran + Lex	53.4	71.6	59.6	83.8	60.0	77.3	56.7	76.9
B1-B3: Tran + Lex	61.3	77.7	65.9	85.9	66.0	80.7	63.7	80.1
B1-B4: Tran + Lex + Sem	73.8	83.6	76.0	89.7	74.6	85.3	74.7	86.1
B1-B5: Tran + Lex + Sem	75.0	84.3	76.5	89.7	76.8	86.8	75.8	86.6
B1-B6: Tran + Lex + Sem + Syn	75.8	84.3	76.2	89.7	76.8	87.2	**76.1**	**86.7**
B1-B5,B7: Tran + Lex + Sem + Syn	75.6	84.1	76.0	89.3	76.8	86.8	75.9	86.4
B2-B6:Lex + Sem + Syn	72.0	82.0	73.2	87.9	76.5	89.3	73.8	85.6

Table 5: Sentence accuracy (Acc) and segment F1 (F1) using semi-Markov CRFs with different features.

example, IN (preposition or subordinating conjunction) is a strong indicator of Other, while TO and IM:*Date* usually do not co-occur. Some features, however, may appear less "correct". This is largely due to the inaccurate output of the POS tagger. For example, a large number of actor names were mis-tagged as RB, resulting in a high positive weight of the feature (RB, IM:*Cast*).

Positive	Negative
(IN, Other),	(TO, IM:*Date*)
(VBD, Other)	(IN, IM:*Cast*)
(CD, IM:*Date*)	(CD, IH)
(RB, IM:*Cast*)	(IN, IM:*Character*)

Table 6: Syntactic features with the largest positive/negative weights in the CRF model for *Movie*

Similarly, we added segment-level POS tag features (B6) to semi-Markov CRFs, which lead to the best overall results as shown by the highlighted numbers in Table 5. Again many of the dominant features are consistent with our intuition. For example, the most positive feature for *Movie* is (CD JJS, IM:*Rating*) (*e.g.* 100 best). When syntactic features based on chunking results (B7) are used instead of B6, the performance is not as good.

Transition features

In addition, it is interesting to see the importance of transition features in both models. Since web queries do not generally follow the linear order principle, is it helpful to incorporate transition features in learning? To answer this question, we dropped the transition features from the best systems, corresponding to the last rows in Table 4 and 5. This resulted in substantial degradations in performance. One intuitive explanation is that although web queries are relatively "order-free", statistically speaking, some orders are much more likely to occur than others. This makes it beneficial to use transition features.

Comparison to syntactic analysis

Finally, we conduct a simple experiment by using the heuristics described in Section 3.2 in extracting IHs from queries. The precision and recall of IHs averaged over all 3 domains are 50.4% and 32.8% respectively. The precision and recall numbers from our best model-based system, *i.e.*, B1-B6 in Table 5, are 89.9% and 84.6% respectively, which are significantly better than those based on pure syntactic analysis.

7 Conclusions

In this work, we make the first attempt to define the semantic structure of noun phrase queries. We propose statistical methods to automatically extract IHs, IMs and the semantic labels of IMs using a variety of features. Experiments show the effectiveness of semantic features and syntactic features in both Markov and semi-Markov CRF models. In the future, it would be useful to explore other approaches to automatic lexicon discovery to improve the quality or to increase the coverage of both IH and IM lexicons, and to systematically evaluate their impact on query understanding performance.

The author would like to thank Hisami Suzuki and Jianfeng Gao for useful discussions.

References

Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. 2009. Sources of evidence for vertical selection. In *SIGIR'09: Proceedings of the 32st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of English web-search queries. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1021–1030.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *ICML'05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Jianfeng Gao, Jian-Yun Nie, Jian Zhang, Endong Xun, Ming Zhou, and Chang-Ning Huang. 2001. Improving query translation for CLIR using statistical models. In *SIGIR'01: Proceedings of the 24th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*.

Jinyoung Kim, Xiaobing Xue, and Bruce Croft. 2009. A probabilistic retrieval model for semistructured data. In *ECIR'09: Proceedings of the 31st European Conference on Information Retrieval*, pages 228–239.

John Lafferty, Andrew McCallum, and Ferdando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289.

Fangtao Li, Xian Zhang, Jinhui Yuan, and Xiaoyan Zhu. 2008a. Classifying what-type questions by head noun tagging. In *COLING'08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 481–488.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2008b. Learning query intent from regularized click graph. In *SIGIR'08: Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, July.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR'09: Proceedings of the 32st Annual International ACM SIGIR conference on Research and Development in Information Retrieva*.

Mehdi Manshadi and Xiao Li. 2009. Semantic tagging of web search queries. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191.

Donald Metzler and Bruce Croft. 2005. Analysis of statistical question classification for fact-based questions. *Jounral of Information Retrieval*, 8(3).

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically har-vesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguis-tics and the 44th annual meeting of the ACL*, pages 113–120.

Stelios Paparizos, Alexandros Ntoulas, John Shafer, and Rakesh Agrawal. 2009. Answering web queries using structured data sources. In *Proceedings of the 35th SIGMOD international conference on Management of data*.

Marius Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI'07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

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-08: HLT*.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *EMNLP'09: Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 238–247.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *CIKM'04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49.

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

Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. Building bridges for web query classification. In *SIGIR'06: Proceedings of the 29th Annual International ACM SIGIR conference on research and development in information retrieval*, pages 131–138.