Extracting Query Facets from Search Results

Weize Kong and James Allan
Center for Intelligent Information Retrieval
School of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{wkong, allan}@cs.umass.edu

ABSTRACT

Web search queries are often ambiguous or multi-faceted, which makes a simple ranked list of results inadequate. To assist information finding for such faceted queries, we explore a technique that explicitly represents interesting facets of a query using groups of semantically related terms extracted from search results. As an example, for the query "baggage allowance", these groups might be different airlines, different flight types (domestic, international), or different travel classes (first, business, economy). We name these groups query facets and the terms in these groups facet terms. We develop a supervised approach based on a graphical model to recognize query facets from the noisy candidates found. The graphical model learns how likely a candidate term is to be a facet term as well as how likely two terms are to be grouped together in a query facet, and captures the dependencies between the two factors. We propose two algorithms for approximate inference on the graphical model since exact inference is intractable. Our evaluation combines recall and precision of the facet terms with the grouping quality. Experimental results on a sample of web queries show that the supervised method significantly outperforms existing approaches, which are mostly unsupervised, suggesting that query facet extraction can be effectively learned.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Query formulation*

General Terms

Algorithms, Experimentation

Keywords

Query Facet, Semantic Class Extraction, Multi-faceted Query

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

1. INTRODUCTION

Web search queries are often ambiguous or multi-faceted [27]. Current popular approaches try to diversify the result list to account for different search intents or query subtopics [24]. A weakness of this approach is that the query subtopics are hidden from the user, leaving him or her to guess at how the results are organized.

In this work, we attempt to extract query facets from web search results to assist information finding for these queries. We define a query facet as a set of coordinate terms – i.e., terms that share a semantic relationship by being grouped under a more general hypernym ("is a" relationship). For example, for the query *mars landing*, three possible query facets are shown in Table 1.

Table 1: Example query facets

Query: mars landing
1. Curiosity, Opportunity, Spirit
2. USA, UK, Soviet Union
3. video, pictures, news
Query: baggage allowance
1. Delta, Jetblue, AA, Continental, ...
2. domestic, international
3. first class, business class, economy class
4. weight, size, quantity
Query: mr bean
1. comics, movies, tv, books
2. the curse of mr bean, mr bean goes to town, ...
3. rowan atkinson, richard wilson, jean rochefort, ...
4. mr bean, irma gobb, rupert, hubert, ...

The first query facet, {*Curiosity*, *Opportunity*, *Spirit*}, includes different Mars rovers. The second query facet, {*USA*, *UK*, *Soviet Union*}, includes countries relevant to Mars landings. These are both facets where the terms are instances of the same semantic class. Somewhat differently, the last facet, {*video*, *pictures*, *news*}, includes labels for different query subtopics. These labels can be viewed as instances of a special semantic class, the subtopics of the query *mars landing*.

Query facets can be used to help improve search experience in many ways. Like in faceted search [6], query facets can help users to navigate through different topics of the search results by applying multiple filters. Using the examples in Table 1, for query *baggage allowance*, a user can select *Delta*, *international*, *business class*, *quantity* from each of its facets, to find pages discussing the number of bags allowed on Delta's international business class flights. Query facets

can also be used as query suggestions or clarification questions to help users specify search intent. For example, for the query *mars landing*, a system might suggest the query facet {*video*, *pictures*, *news*} or generate clarification questions like "*Which Mars rover are you looking for? a) Curiosity, b) Opportunity, c) Spirit*". Query facets are also useful for exploratory search since they succinctly summarize interesting facts for the issued query. For example, facets for the query *mr bean* list episodes titles, characters and casts for the Mr. Bean television series.

In this paper we develop a supervised method based on a graphical model for query facet extraction. The graphical model learns how likely it is that a term should be selected and how likely it is that two terms should be grouped together in a query facet. Further, the model captures the dependencies between the two factors. We propose two algorithms for approximate inference on the graphical model since exact inference is intractable. Also, we design an evaluation metric for query facet extraction, which combines recall and precision of the facet term, with the grouping quality.

The rest of this paper is organized as follows. We discuss related work in Section 2, and then present the problem formulation in Section 3. Section 4 describes the general framework we use for query facet extraction. Section 5 describes our graphical model based approach in detail. Section 6 briefly describes two alternate approaches that we use as baselines. We describe the dataset as well as the metrics we used for evaluation in Section 7, and report experimental results in Section 8. Finally, we conclude the work in Section 9.

2. RELATED WORK

Related work of query facet extraction can be divided into the following topics.

2.1 Search Results Diversification

Search result diversification has been studied as a method of tackling ambiguous or multi-faceted queries while a ranked list of documents remains the primary output feature of Web search engine today[24]. It tries to diversify the ranked list to account for different search intents or query subtopics. A weakness of search result diversification is that the query subtopics are hidden from the user, leaving him or her to guess at how the results are organized. Query facet extraction addresses this problem by explicitly presenting different facets of a queries using groups of coordinate terms.

2.2 Search Results Clustering/Organization

Search results clustering is a technique that tries to organize search results by grouping them into, usually labeled, clusters by query subtopics [4]. It offers a complementary view to the flat ranked list of search results. Most previous work exploited different textual features extracted from the input texts and applied different clustering algorithms with them. Instead of organizing search results in groups, there is also some work [14, 15, 16] that summarizes search results or a collection of documents in a topic hierarchy. For example, Lawrie et al. [14, 15] used a probabilistic model for creating topical hierarchies, in which a graph is constructed based on conditional probabilities of words, and the topic words are found by approximately maximizing the predictive power and coverage of the vocabulary. Our work is different from these work in that our target is to extract different facets of a query from search results, instead of organizing the search results.

2.3 Query Subtopic/Aspect Mining

To address multi-faceted queries, much previous work studied mining query subtopics (or aspects). A query subtopic is often defined as a distinct information need relevant to the original query. It can be represented as a set of terms that together describe the distinct information need [29, 31, 5] or as a single keyword that succinctly describes the topic [28]. Different resources have been used for mining query subtopics, including query logs [30, 11, 32, 29, 31, 33], document corpus [2] and anchor texts [5].

A query subtopic is different from a query facet in that the terms in a query subtopic are not restricted to be coordinate terms, or have peer relationships. Query facets, however, organize terms by grouping "sibling" terms together. For example, {*news*, *cnn*, *latest news*, *mars curiosity news*} is a valid query subtopic for the query *mars landing*, which describes the search intent of Mars landing news, but it is not a valid query facet, given our definition, since the terms in it are not coordinate terms. A valid query facet that describes Mars landing news could be {*cnn*, *abc*, *fox*}, which includes different news channels. In a recent work [7], Dou et al. developed a system to extract query facets from web search results and showed the potential of doing so. However, the unsupervised method they proposed is far from optimal, and it does not improve by having human labels available. Also, to the best of our knowledge, their evaluation can be problematic in some cases, which will be discussed in Section 7.2.3.

2.4 Semantic Class Mining

Semantic class mining can be used to help query facet extraction. **Class attribute extraction** [17, 18] aims to extract attributes for a target semantic class usually specified by as a set of representative instances. For example, given a semantic class *country*, together with some instances like *USA*, *UK*, *China*, some class attributes can be *capital city*, *president*, *population*. Those extracted class attributes can be used as query subtopics. However, class attribute extraction targets semantic classes, not general search queries.

Semantic class extraction aims to automatically mine semantic classes represented as their class instances from certain data corpus. Existing approaches can be roughly divided into two categories: distributional similarity and pattern-based [25]. The distributional similarity approach is based on the distributional hypothesis [8], that terms occurring in analogous contexts tend to be similar. Different types contexts has been studied for this problem, including syntactic context [20] and lexical context [21, 1, 19]. The pattern-based approach applied textual patterns [9, 22], HTML patterns [26] or both [34, 25] to extract instances of a semantic class from some corpus. The raw semantic class extracted can be noisy. To address this problem, Zhang et al. [34] used topic modeling to refine the extracted semantic classes. Their assumption is that, like documents in the conventional setting, raw semantic classes are generated by a mixture of hidden semantic class. In this paper, we apply pattern-based semantic class extraction on the top-ranked Web documents to extract candidates for finding query facets.

2.5 Faceted Search

Faceted search is a technique for accessing information organized according to a faceted classification system, allowing users to digest, analyze and navigate through multidimensional data. It is widely used in e-commerce and digital libraries [6]. Faceted search is similar to query facet extraction in that both of them use sets of coordinate terms to represent different facets of a query. However, most existing works for faceted search are build on as specific domain or predefined categories [7], while query facet extraction does not restrict queries in a specific domain, like products, people, etc.

3. PROBLEM FORMULATION

Query facet extraction is the problem of finding query facets for a given query q from available resources, such as web search results. A **query facet** $F = \{t\}$ is a set of coordinate terms, terms that are part of a semantic set, which we call **facet terms**. These facet terms can be instances of a semantic class, for example *Curiosity*, *Opportunity*, *Spirit* are all Mars rovers. They can be labels for query subtopics, such as *video*, *pictures*, *news* for the query *mars landing*. We use $\mathcal{F} = \{F\}$ to denote the set of query facets. $T_{\mathcal{F}} = \{t|t \in F, F \in \mathcal{F}\}$ is the set of all the facets terms that appear in $\mathcal{F}$.

Query facets can be extracted from a variety of different resources, such as a query log, anchor text, taxonomy and social folksonomy. In this work, we only focus on extracting query facets from the top k web search results $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$. We intend to explore the use of other information sources for this problem in future work.

4. GENERAL FRAMEWORK

In this section, we describe the general framework we use for extracting query facet from web search results. Given a query q, we retrieve the top k search results, $\mathcal{D}$, as input to our system. Then query facets $\mathcal{F}$ are extracted by, first, extracting candidates from search results $\mathcal{D}$ and then finding query facets from the candidates.

4.1 Extracting candidate lists

Similar to Dou et al. [7], we use pattern-based semantic class extraction approach [25] to extract lists of coordinate terms from search results as candidates for query facets. In pattern-based semantic class extraction, patterns are applied on the corpus to discover specific relationships between terms. For example, the pattern "*NP* such as *NP*, *NP*, ..., and *NP*" can be used to extract coordinate terms and their hypernyms from text. Besides lexical patterns, HTML patterns are often used on HTML documents to extract coordinate terms from some HTML structures, like <UL>, <SELECT> and <TABLE>.

Table 2: Semantic class extraction patterns

Type	Pattern
Lexical	*item*, {,*item*}*, (and|or) {other} *item*
HTML	<select><option>*item*</option>...</select>
	item ...
	item ...
	<table><tr><td>*item*<td>...</table>

We use both of the two types of patterns, summarized in Table 2. In the table, all *items* in each pattern are extracted as a candidate list. For example, from the text sentence "*... Mars rovers such as Curiosity, Opportunity and Spirit*", according to the lexical pattern, we will extract a candidate list {*Curiosity*, *Opportunity*, *Spirit*}. For the lexical pattern, we also restrict those *items* to be siblings in the parse tree of that sentence. We use the PCFG parser [12] implemented in Stanford CoreNLP[1] for parsing documents. For HTML tables, following Dou et al. [7], lists from each column and each row are extracted.

After extracting the lists from the top ranked results $\mathcal{D}$, we further process them as follows. First, all the list items are normalized by converting text to lowercase and removing non-alphanumeric characters. Then, we remove stopwords and duplicate items in each lists. Finally, we discard all lists that contain fewer than two item or more than 200 items. After this process, we have a set of candidate lists $\mathcal{L} = \{L\}$, where each list $L = \{t\}$ is a set of list items.

4.2 Finding query facets from candidate lists

The candidate lists extracted are usually noisy [34], and could be non-relevant to the issued query, therefore they cannot be used directly as query facets. Table 3 shows four candidate lists extracted for the query *mars landing*. L_1 contains list items that are relevant to *mars landing*, but they are not coordinate terms. L_2 is a valid query facet, but it is incomplete – another Mars rover *Spirit* appears in L_3. L_3 is extracted from the sentence, "*It is bigger than the 400-pound Mars Exploration rovers, Spirit and Opportunity, which landed in 2004*". The list item "*the 400 pound mars exploration rovers*" is an extraction error.

Table 3: Four candidate lists for query *mars landing*

L_1:	curiosity rover, mars, nasa, space
L_2:	curiosity, opportunity
L_3:	the 400 pound mars exploration rovers, spirit, opportunity
L_4:	politics, religion, science technology, sports, ...

Since the candidate lists are frequently noisy, we need an effective way to find query facets from extracted candidate lists. More formally, given a set of candidate lists $\mathcal{L} = \{l\}$, the task is to find a set of query facets $\mathcal{F}$, where $T_{\mathcal{F}} \subseteq T_{\mathcal{L}}$. Similar to $T_{\mathcal{F}}$, $T_{\mathcal{L}} = \{t|t \in L, L \in \mathcal{L}\}$ is the set of all list items in $\mathcal{L}$. To address this problem, we develop a graphical model, which learns how likely a list item is a facet term, how likely two list items should be grouped in a query facet, and capture the dependencies between the two factors.

5. A GRAPHICAL MODEL FOR FINDING QUERY FACETS

In this section, we describe the directed graphical model we use to find query facts form noisy candidate lists. A directed graphical model (or Bayesian network) is a graphical model that compactly represents a probability distribution over a set of variables [23]. It consists of two parts: 1) a directed acyclic graph in which each vertex represents a variable, and 2) a set of conditional probability distributions that describe the conditional probabilities of each vertex given its parents in the graph.

We treat the task of finding query facets from candidate lists as a labeling problem, in which we are trying to predict

[1]http://nlp.stanford.edu/software/corenlp.shtml

1) whether a list item is a facet term, and 2) whether a pair of list items is in one query facet. Then, we used a directed graphical model to exploit the dependences that exist between those labels. Similar to conditional random fields [13], we directly model the conditional probability $P(y|x)$, where y is the label we are trying to predict and x is the observed data – list items and item pairs. Thus, it avoids modeling the dependencies among the input variables x, and can handle a rich set of features. For our graph model, exact maximum a posteriori inference is intractable; therefore, we approximate the results using two algorithms.

5.1 The Graphical Model

5.1.1 Graph

First we define all the variables in our graphical model. Let $Y = \{y_i\}$, where $y_i = 1\{t_i \in T_{\mathcal{F}}\}$ is a label indicating whether a list item t_i is a facet term. Here $1\{\cdot\}$ is an indicator function which takes on a value of 1 if its argument is true, and 0 otherwise. $p_{i,j}$ denotes the list items pair (t_i, t_j), and $P_{\mathcal{L}} = \{p_{i,j}|p_{i,j} = (t_i, t_j), t_i, t_j \in T_{\mathcal{L}}, t_i \neq t_j\}$ denotes all the items pairs in $T_{\mathcal{L}}$. Let $Z = \{z_{i,j}\}$, where $z_{i,j} = 1\{\exists F \in \mathcal{F}, t_i \in F \wedge t_j \in F\}$ is a label indicates whether the corresponding item pair $p_{i,j}$ should be grouped together in a query facet. The vertices in our graphical model are $V = T_{\mathcal{L}} \cup P_{\mathcal{L}} \cup Y \cup Z$. Note that the list items $T_{\mathcal{L}}$, and item pairs $P_{\mathcal{L}}$ are always observed.

As shown in Figure 1, there are three types of edges in the graph: 1) edges from each list item t_i to its corresponding labels y_i; 2) edges that point to each item pair label $z_{i,j}$ from the two corresponding list items y_i and y_j; 3) edges from each item pair $p_{i,j}$ to its corresponding label $z_{i,j}$.

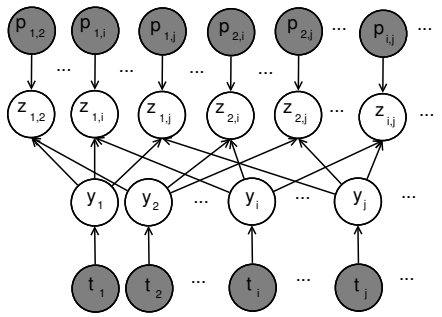

Figure 1: A graphical model for candidate list data

5.1.2 Conditional Probability Distribution

We use logistic-based conditional probability distributions (CPDs) for variable y_i and $z_{i,j}$, defined as in Equation 1 and Equation 2.

$$P(y_i = 1|t_i) = \frac{1}{1 + \exp\{-\sum_k \lambda_k f_k(t_i)\}} \quad (1)$$

$$P(z_{i,j} = 1|p_{i,j}, y_i, y_j) = \frac{y_i y_j}{1 + \exp\{-\sum_k \mu_k g_k(p_{i,j})\}} \quad (2)$$

f_k and g_k are features that characterize a list item and a item pair respectively. λ and μ are the weights associated with f_k and g_k respectively. Compared to a conventional logistic function, Equation 2 has an extra term, $y_i y_j$, in the numerator. When $y_i = 0$ or $y_j = 0$, we have $P(z_{i,j} = 1|p_{i,j}, y_i, y_j) = 0$. This means when either of the two list items is not a facet term, the two items can never appear in a query facet together. When both of the t_i and t_j are facet terms, $P(z_{i,j} = 1|p_{i,j}, y_i, y_j)$ becomes a conventional logistic function, which models the probability of t_i and t_j being grouped together in a query facet, given the condition that both t_i and t_j are facet term.

The joint conditional probability for the graphical model is calculated as

$$P(Y, Z|T_{\mathcal{L}}, P_{\mathcal{L}}) = \prod_{y_i \in Y} P(y_i|t_i) \prod_{z_{i,j} \in Z} P(z_{i,j}|p_{i,j}, y_i, y_j) \quad (3)$$

where the CPDs are defined in Equation 1 and Equation 2.

5.1.3 Parameter Estimation

The training set for the graphical model can be denoted as $\{T_{\mathcal{L}}, P_{\mathcal{L}}, Y^*, Z^*\}$, where Y^*, Z^* are the ground truth labels for the list items $T_{\mathcal{L}}$ and item pairs $P_{\mathcal{L}}$. The conditional probability of the training set can be calculated according to Equation 4.

$$P(\lambda, \mu) = \prod_{T_{\mathcal{L}}, P_{\mathcal{L}}} P(Y^*, Z^*|T_{\mathcal{L}}, P_{\mathcal{L}}) \quad (4)$$

The log-likelihood $l(\lambda, \mu)$, can be calculated as follows,

$$l(\lambda, \mu) = l_t(\lambda) + l_p(\mu) \quad (5)$$

$$l_t(\lambda) = \sum_{T_{\mathcal{L}}} \sum_{y_i \in Y} \log P(y_i|t_i) - \frac{\sum_k \lambda_k^2}{2\sigma^2} \quad (6)$$

$$l_p(\mu) = \sum_{T_{\mathcal{L}}} \sum_{z_{i,j} \in Z} \log P(z_{i,j}|p_{i,j}, y_i, y_j) - \frac{\sum_k \mu_k^2}{2\gamma^2} \quad (7)$$

$l(\lambda, \mu)$ is separated into two parts, $l_t(\lambda)$ and $l_p(\mu)$. The last terms of Equation 6 and Equation 7 are served as regularizers which penalize large values of λ, μ. σ and γ are regularization parameters that control the strength of penalty. Notice that, in the train set, for those item pairs $p_{i,j}$ with any of its list item not being a facet term, their labels $z_{i,j} = 0$. According to Equation 2, for those item pairs, $\log P(z_{i,j}|p_{i,j}, y_i, y_j) = 0$, which makes no contribution to $l_p(\mu)$, and thus $l_p(\mu)$ can be simplified as

$$l_p(\mu) = \sum_{T_{\mathcal{L}}} \sum_{z_{i,j} \in Z'} \log P(z_{i,j}|p_{i,j}, y_i, y_j) - \frac{\sum_k \mu_k^2}{2\gamma^2} \quad (8)$$

where Z' is a subset of Z, which contains only the labels for item pairs with both of its list items being facet terms.

We can see that Equations 6 and 8 are exactly the same as log-likelihoods for two separated logistic regressions. In fact, Equation 6 learns a logistic regression model for whether a list item is a facet term, and Equation 8 learns a logistic regression model for whether two facet terms should be grouped together. The parameter λ and μ can be learned by maximizing the log-likelihood using gradient descent, exactly same as in logistic regression.

5.2 Inference

When given a new labeling task, we could perform maximum a posteriori inference - compute the most likely labels Y^*, Z^* by maximizing the joint conditional probability $P(Y, Z|T_{\mathcal{L}}, P_{\mathcal{L}})$. After that, the query facet set $\mathcal{F}$ can be easily induced from the labeling Y^*, Z^*. (Collect list items with $y_i = 1$ as facet terms, and group any two of them into

a query facet if the corresponding $z_{i,j} = 1$.) Note that the graphical model we designed does not enforce the labeling to produce strict partitioning for facet terms. For example, when $Z_{1,2} = 1$, $Z_{2,3} = 1$, we may have $Z_{1,3} = 0$. Therefore, an optimal labeling results may induce an overlapping clustering. To simplify the problem, we add the strict partitioning constraint that each facet term belongs to exactly one query facet. Also, to directly produce the query facets, instead of inducing them after predicting labels, we rephrase the optimization problem as follows. First, we use the following notations for log-likelihoods,

$$\begin{aligned} s_t(t_i) &= \log P(y_i = 1|t_i) \\ \overline{s_t}(t_i) &= \log(1 - P(y_i = 1|t_i)) \\ s_p(t_i, t_j) &= \log P(p_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1) \\ \overline{s_p}(t_i, t_j) &= \log(1 - P(p_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1)) \end{aligned}$$

Using the notations above, the log-likelihood $l(\mathcal{F})$ for a particular query facet set $\mathcal{F}$ formed from $\mathcal{L}$ can be written as

$$\begin{aligned} l(\mathcal{F}) &= l_t(\mathcal{F}) + l_p(\mathcal{F}) \\ l_t(\mathcal{F}) &= \sum_{t \in T_{\mathcal{F}}} s_t(t_i) + \sum_{t \notin T_{\mathcal{F}}} \overline{s_t}(t_i) \\ l_p(\mathcal{F}) &= \sum_{F \in \mathcal{F}} \sum_{t_i, t_j \in F} s_p(t_i, t_j) + \sum_{\substack{F, F' \\ \in \mathcal{F}}} \sum_{\substack{t_i \in F, \\ t_j \in F'}} \overline{s_p}(t_i, t_j) \quad (9) \end{aligned}$$

In the right hand side of Equation 9, the first term is the intra-facet score, which sums up $s_p(\cdot, \cdot)$ for all the item pairs in each query facet. The second term is the inter-facet score, which sums up the $\overline{s_p}(\cdot, \cdot)$ for each item pair that appears in different query facets. Then the optimization target becomes $\mathcal{F} = \arg\max_{\mathcal{F} \in \mathfrak{F}} l(\mathcal{F})$, where $\mathfrak{F}$ is the set of all possible query facet sets that can be generated from $\mathcal{L}$ with the strict partitioning constraint.

This optimization problem is NP-hard, which can be proved by a reduction from the Multiway Cut problem [3]. Therefore, we propose two algorithms, **QF-I** and **QF-J**, to approximate the results.

5.2.1 QF-I

QF-I approximates the results by predicting whether a list item is a facet term and whether two list items should be grouped in a query facet independently, which is accomplished two phases. In the first phase, QF-I selects a set of list items as facet terms according to $P(y_i|t_i)$. In this way, the algorithm predicts whether a list item t_i is a facet term independently, ignoring the dependences between y_i and its connected variables in Z. In our implementation, we simply select list items t_i with $P(t_i) > w_{min}$ as facet terms. (For convenience, we use $P(t_i)$ to denote $P(y_i = 1|t_i)$.) In the second phase, the algorithm clusters the facet terms $T_{\mathcal{F}}$ selected in the first phase into query facets, according to $P(t_i, t_j)$. ($P(t_i, t_j)$ is used to denote $P(z_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1)$). Many clustering algorithm can be applied here, using $P(t_i, t_j)$ as the distance measure. For our implementation, we use a cluster algorithm based on WQT [7], because it considers the importance of nodes while clustering. We use $P(t_i)$ as the measure for facet term importance, and $d_t(t_i, t_j) = 1 - P(t_i, t_j)$ as the distance measure for facet terms. The distance between a cluster and a facet term is computed using complete linkage distance, $d_f(F, t) = max_{t' \in F} d(t, t')$, and the diameter of a cluster can be calculated as $dia(F) = \max_{t_i, t_j \in F} d_t(t_i, t_j)$. The algorithm is summarized in Algorithm 1. It processes the facet terms in decreasing order of $P(t)$. For each facet term remaining in the pool, it builds a cluster by iteratively including the facet term that is closest to the cluster, until the diameter of the cluster surpasses the threshold d_{max}.

Algorithm 1 WQT for clustering facet term used in QF-I

Input: $T_{\mathcal{F}}, P(t), d_f(F, t), dia(F), d_{max}$
Output: $\mathcal{F} = \{F\}$
1: $T_{pool} \leftarrow \mathcal{F}$
2: **repeat**
3: $t \leftarrow \arg\max_{t \in T_{pool}} P(t)$
4: $F \leftarrow \{t\}$
5: iteratively include facet term $t' \in T_{pool}$ that is closest to F, according to $d_f(F, t')$, until the diameter of the cluster, $dia(F)$, surpasses the threshold d_{max}.
6: $\mathcal{F} \leftarrow \mathcal{F} \cup \{F\}$, $T_{pool} \leftarrow T_{pool} - F$
7: **until** T_{pool} is empty
8: **return** $\mathcal{F}$

5.2.2 QF-J

QF-I finds query facets based on the graphical model by performing inference of y_i and $z_{i,j}$ independently. The second algorithm, QF-J, instead tries to perform joint inference by approximately maximizing our target $l(\mathcal{F})$ with respect to y_i and $z_{i,j}$ iteratively. The algorithm first guesses a set of list items as facet terms. Then it clusters those facet terms by approximately maximizing $l_p(\mathcal{F})$, using a greedy approach. After clustering, the algorithm checks whether each facet term "fits" in its cluster, and removes those that do not fit. Using the remaining facet terms, the algorithm repeats the process (clustering and removing outliers) until convergence.

QF-J is outlined in Algorithm 2. The input to the algorithm are the candidate list item set $T_{\mathcal{L}}$, and the log-likelihoods $l(\mathcal{F})$, $l_p(\mathcal{F})$. In the first step, we select top n list items according to $s_t(t)$ as the initial facet terms, because it is less sensitive to the absolute value of the log-likelihood. In our experiment, n is set to 1000 to make sure most of the correct facet terms are included. Then, the algorithm improves $l(\mathcal{F})$ by iteratively performing functions Cluster and RemoveOutliers. Cluster performs clustering over a given set of facet terms. In step 10 to 12, it puts each facet terms into a query facet by greedily choosing the best facet, or creates a singleton for the list item, according to the resulting log-likelihood, $l_p(\mathcal{F})$. We choose to process these list items in decreasing order of $s_t(t)$, because it is more likely to form a good query facet in the beginning by doing so. RemoveOutliers removes facet terms according to the joint log-likelihood $l(\mathcal{F})$. In step 20 to 22, it checks each facet term to see if it fits in the facet, and removes outliers. F' is the set of facet terms the algorithm selected when processing each facet F.

5.2.3 Ranking Query Facets

The output of QF-I and QF-J is a query facet set $\mathcal{F}$. To produce ranking results, we defined a score for a query facet as $score_F(F) = \sum_{t \in F} P(t)$, and rank the query facets according to this scoring, in order to present more facet terms in the top. Facet terms within a query facet are ranked according to $score_t(t) = P(t)$.

Algorithm 2 QF-J

Input: $T_{\mathcal{L}} = \{t\}, l, l_p$
Output: $\mathcal{F} = \{F\}$

$T_{\mathcal{F}} \leftarrow$ top n list items from $T_{\mathcal{L}}$ according to $s_t(\cdot)$
repeat
  $\mathcal{F} \leftarrow$ CLUSTER$(T_{\mathcal{F}}, l_p)$
  $T_{\mathcal{F}} \leftarrow$ REMOVEOUTLIERS$(\mathcal{F}, l)$
until converge
return $\mathcal{F}$

function CLUSTER$(T_{\mathcal{F}}, l_p)$
  $\mathcal{F} \leftarrow \emptyset$
  for each $t \in T_{\mathcal{F}}$ in decreasing order of $s_t(t)$ **do**
    Choose to put t into the best facet in $\mathcal{F}$ or add t as a singleton into $\mathcal{F}$, whichever that has the highest resulting $l_p(\mathcal{F})$.
  end for
  return $\mathcal{F}$
end function

function REMOVEOUTLIERS$(\mathcal{F}, l)$
  $T_{\mathcal{F}} \leftarrow$ all facet terms in $\mathcal{F}$
  for each $F \in \mathcal{F}$ **do**
    $F' = \emptyset$
    for each $t \in F$ in decreasing order of $s_t(\cdot)$ **do**
      choose to add t into F' or not, whichever has the highest resulting $l(\{F'\})$
      if not, $T_{\mathcal{F}} \leftarrow \mathcal{F} - \{t\}$
    end for
  end for
  return $T_{\mathcal{F}}$
end function
return $\mathcal{F}$

5.3 Features

There are two types of features used in our graphical model, summarized in Table 4.

Item features, $f_k(t)$ in the graphical model, characterize a single list item. To capture the relevance of item t to the query, we use some TF/IDF-based features extracted from the top k search results, $\mathcal{D}$. For example, *snippetDF* is the number of snippets in top k search results that contain item t. *snippetDF* and other frequency-based features are normalized using $log(frequency + 1)$. To capture how likely item t is to be an instance of a semantic class, we use features extracted from candidate lists. For example, *listTF* is the frequency of t in the candidate lists extracted from $\mathcal{D}$. Some list items occur frequently in candidate lists across different queries, such as *home*, *contact us* and *privacy policy*. They are treated as stopwords, and removed from the candidate lists. We also use *listIDF* to cope with this problem. *listIDF* is the IDF of a list item in a general collection of candidate lists we extracted (see Section 7.1). It is calculated as $listIDF(t) = \log \frac{N - N_t + 0.5}{N_t + 0.5}$, where N is the total number of lists in the collection, N_t is the number of lists contain t. The same form is used for *clueIDF*, IDF in ClueWeb09[2] collection.

Item Pair Features, $g(p_{i,j})$ in the graphical model, are used to capture how likely a pair of list items should be grouped into a query facet, given that the two list item both are facet terms. This can be measured by context similarity [25]. For *textContextSim*, we use window size 25, and represent text context as a vector of TF weights. Cosine similarity is used as the similarity measure. Similarly, we use the candidate lists that contain the list item as its list context, and calculate *listContextSim* in the same way as *textContextSim*.

[2]http://lemurproject.org/clueweb09

Table 4: Two types of features

Item Features for list item t	
length	Number of words in t
clueIDF	IDF of t in ClueWeb09 collection
TF	Term frequency of t in $\mathcal{D}$
DF	Document frequency of t in $\mathcal{D}$
wDF	Weighted DF. Each document count weighted by $1/\sqrt{docRank}$
SF	Site frequency. Number of unique websites in $\mathcal{D}$ that contain t
titleTF	TF of t for the titles of $\mathcal{D}$
titleDF	DF of t for the titles of $\mathcal{D}$
titleSF	SF of t for the titles of $\mathcal{D}$
snippetTF	TF of t for the snippets of $\mathcal{D}$
snippetDF	DF of t for the snippets of $\mathcal{D}$
snippetSF	SF of t for the snippets of $\mathcal{D}$
listTF	Frequency of t in candidate lists extracted from $\mathcal{D}$
listDF	Number of documents that contain t in their candidate lists
listSF	Number of unique websites that contain t in their candidate lists
listIDF	IDF of t in a general candidate list collection
TF.clueIDF	TF $\times$ clueIDF
listTF.listIDF	listTF $\times$ lisIDF
Item Pair Features for item pair $p_{i,j} = (t_i, t_j)$	
lengthDiff	Length difference, $\lvert len(t_i) - len(t_j) \rvert$
listCooccur	Number of candidate lists extracted from $\mathcal{D}$, in which t_i, t_j co-occur
textContextSim	Similarity between text contexts
listContextSim	Similarity between list contexts

6. OTHER APPROACHES

In this section, we describe two alternative approaches for finding query facets from candidate lists. They are used as baselines in our experiments.

6.1 QDMinder

Dou et al. [7] developed QDMiner/QDM for query facet extraction, which appears to be the first work that addressed the problem of query facet extraction. To solve the problem of finding query facets from the noisy candidate lists extracted, they used an unsupervised clustering approach. It first scores each candidate list by combining some TF/IDF-based scores. The candidate lists are then clustered with bias toward important candidate lists, using a variation of the Quality Threshold clustering algorithm [10]. After clustering, clusters are ranked and list items in each clusters are ranked/selected based on some heuristics. Finally, the top k clusters are returned as results. This unsupervised approach does not gain by having human labels available. Also, by clustering lists, they lose the flexibility of breaking a candidate list into different query facets.

6.2 Topic modeling

In semantic class extraction, Zhang et al. [34] proposed to use topic models to find high-quality semantic classes from a large collection of extracted candidate lists. Their assumption is, like documents in the conventional setting, candidate lists are generated by a mixture of hidden topics, which are the query facets in our case. pLSA and LDA are used in their experiments. We find this approach can be directly used for finding query facets from candidate lists. The major change we need to make is that: in semantic class extraction, topic modeling is applied globally on the candidate lists (or a sample of them) from the entire corpus; in query facet extraction, we apply topic modeling only on the top k search results $\mathcal{D}$, assuming the coordinate terms in $\mathcal{D}$ are relevant to the query. Then, the topics are returned as query facets, by using the top n list items in each topic (according to the list item's probability in the topic). Though this topic modeling approach is more theoretically motivated, it does not have the flexibility of adding different features to capture different aspects such as query relevance.

7. EVALUATION

7.1 Data

Queries. We constructed a pool of 232 queries from different sources, including random samples from a query log, TREC 2009 Web Track queries [3], example queries appearing in related publications [32, 29] and queries generated by our annotators. Annotators were asked to select queries that they are familiar with from the pool for annotating. Overall, we collect annotations for 100 queries (see Table 5).

Table 5: Query statistics

Source	#queries collected	#queries annotated
query log	100	30
related publications	20	10
TREC 2009 Web Track	50	20
annotators generated	62	40
sum	232	100

Search results. For each query, we acquire the top 100 search results from a commercial Web search engine. A few search results are skipped due to crawl errors, or if they are not HTML Web pages. For the 232-query set, we crawled 22,909 Web pages, which are used for extracting feature $listIDF$ described in Section 5.3. For the 100 annotated queries, the average number of crawled Web pages is 98.7, the minimum is 79, both the maximum and the median are 100.

Query facet annotations. We asked human annotators to construct query facets as ground truth. For each query, we first constructed a pool of terms by aggregating facet terms in the top 10 query facets generated by different models, including two runs from QDM, one run from each of pLSA and LDA using top 10 list items in each query facets, and one run for our graphical model based approach. Then, annotators were asked to group terms in the pool into query facets for each query they selected. Finally, the annotator was asked to give a rating for each constructed query facet, regarding how useful and important the query facet is. The rating scale of good=2/fair=1 is used. Annotation statistics are given in Table 6. There are 50 query facets pooled per query, with 224.8 distinct facet terms per query.

[3]http://trec.nist.gov/data/web/09/wt09.topics.queries-only

Table 6: Annotation statistics

	fair	good	pooled
#terms per query	26.6	55.8	224.8
#facets per query	3.1	4.8	50.0
#terms per facet	8.6	11.6	8.8

7.2 Evaluation Metrics

Query facet extraction can be evaluated from different aspects. We use standard clustering and classification evaluation metrics, as well as metrics designed for this particular task to combine different evaluation aspects.

Notation: we use "$*$" to distinguish between system generated results and human labeled results, which we used as ground truth. For example, $\mathcal{F}$ denotes the system generated query facet set, and $\mathcal{F}^*$ denotes the human labeled query facet set. For convenience, we use T to denote $T_{\mathcal{F}}$ in this section, omitting subscript $\mathcal{F}$. T^* denotes all the facet terms in human labeled query facet set. We use r_{F^*} to denote the rating score for a human labeled facet F^*.

7.2.1 *Effectiveness in finding facet terms*

One aspect of query facet extraction evaluation is how well a system finds facet terms. This can be evaluated using standard classification metrics as follows,

- facet term precision: $P(T, T^*) = \frac{|T \cap T^*|}{|T|}$
- facet term recall: $R(T, T^*) = \frac{|T \cap T^*|}{|T^*|}$
- facet term F1: $FT(T, T^*) = \frac{2|T \cap T^*|}{|T|+|T^*|}$

where facet term F1 is denoted as FT (the T stands for facet *term*) to avoid confusion with a query facet F and clustering F1 defined below. These metrics do not take clustering quality into account.

7.2.2 *Clustering quality*

To evaluate how well a system groups facet terms correctly, similar to Dou et al. [7], we use several existing cluster metrics, namely, Purity, NMI/Normalized Mutual Information and F1 for clustering. To avoid confusion with facet term F1, FT, we call F1 for facet term clustering *facet clustering F1*, and denote it as FP (with P standing for term *pair*).

In our task, we usually have $T \neq T^*$. The facet terms in the system generated and human labeled clustering results might be different: the system might fail to include some human identified facet terms, or it might mistakenly include some "incorrect" facet terms. These standard clustering metrics cannot handle these cases properly. To solve this problem, we adjust $\mathcal{F}$ as if only facet terms in T^* were clustered by the system, since we are only interested in how well the "correct" facet terms are clustered from these metrics. The adjusting is done by removing "incorrect" facet terms ($t \in T - T^*$) from $\mathcal{F}$, and adding each missing facet term ($t^* \in T^* - T$) to $\mathcal{F}$ as singletons. By this adjusting, we do not take into account the effectiveness of finding correct facet terms.

7.2.3 Overall quality

To evaluate the overall quality of query facet extraction, Dou et al. [7] proposed variations of nDCG (Normalized Discounted Cumulative Gain), namely fp-nDCG and rp-nDCG. It first maps each system generated facet F to a human labeled facet F^* that covers the maximum number of terms in F. Then, it assigns the rating r_{F^*} to F, and evaluates $\mathcal{F}$ as a ranked list of query facets using nDCG. The discounted gains are weighted by precision and/or recall of facet terms in F, against its mapped human labeled facet F^*. For fp-nDCG, only precision are used as weight, $\frac{|F^* \cap F|}{|F|}$. For rp-nDCG, precision and recall are multiplied as weight, $\frac{|F^* \cap F|^2}{|F^*||F|}$. However, to the best of our understanding, this metric can be problematic in some cases. When two facets F_1 and F_2 are mapped to a same human labeled facet F^*, only the first facet F_1 is credited and F_2 is simply ignored, even if it is more appropriate to map F_2 to F^* (e.g. F_2 is exactly same as F^*, while F_1 contain only one facet term in F^*).

The quality of query facet extraction is intrinsically multi-faceted. Different applications might have different emphasis in the three factors mentioned above - precision of facet terms, recall of facet terms and clustering quality of facet terms. We propose a metric $PRF_{\alpha,\beta}$ to combine the three factors together, using weighted harmonic mean. Let $p = P(T, T^*)$, $r = R(T, T^*)$, $f = FP(\mathcal{F}, \mathcal{F}^*)$, then $PRF_{\alpha,\beta}$ can be expressed as follows,

$$PRF_{\alpha,\beta}(\mathcal{F}, \mathcal{F}^*) = \frac{(\alpha^2 + \beta^2 + 1)prf}{\alpha^2 rf + \beta^2 pf + pr} \quad (10)$$

where α and β are used to adjust the emphasis between the three factors. When $\alpha = \beta = 1$, we omit the subscript part for simplicity, i.e. $PRF \equiv PRF_{1,1}$.

While $PRF_{\alpha,\beta}$ has the flexibility to adjust emphasis between the three factors, it does not take into account the different ratings associated with query facets. To incorporate ratings, we use a weighted version of $P(T, T^*)$, $R(T, T^*)$ and $FP(\mathcal{F}, \mathcal{F}^*)$ in $PRF_{\alpha,\beta}$. We call the new metric $wPRF_{\alpha,\beta}$. The weighted facet term precision, recall and FT are defined as follows

- weighted facet term precision: $wP(T, T^*) = \frac{\sum_{t \in T \cap T^*} w(t)}{\sum_{t \in T} w(t)}$
- weighted facet term recall: $wR(T, T^*) = \frac{\sum_{t \in T \cap T^*} w(t)}{\sum_{t^* \in T^*} w(t^*)}$
- weighted facet term F1: $wFT(T, T^*) = \frac{2wP(T,T^*)wR(T,T^*)}{wP(T,T^*)+wR(T,T^*)}$

where $w(t)$ is the weight for facet term t, and assigned as follows

$$w(t) = \begin{cases} r_{F^*} & \text{if } t \in T^* \\ 1 & \text{otherwise} \end{cases}$$

Similarly, $wFP(\mathcal{F}, \mathcal{F}^*)$ is computed by weighting its pairwise precision and recall in the same fashion as the weighted facet term precision and recall above. Instead of $w(t)$, we need weight for a pair of facet terms $w(t_1, t_2)$ in this calculation. We assign weight for facet term pair $w(t_1, t_2)$ using their sum, $w(t_1) + w(t_2)$.

8. EXPERIMENT RESULTS

8.1 Experiment settings

We compare effectiveness of the five models, QDM, pLSA, LDA and QF-I, QF-J, on the 100-query data set. All the models take the same candidate lists extracted/cleaned (see Section 4.1) as input. We perform 10-fold cross validation for training/testing and parameter tuning in all experiments and for all models (if applicable). When training the graphical model, we standardize features by removing the mean and scaling to unit variance. We set both of the two regularizers σ and γ in Equation 5 to be 1. There are too many negative instances ($y_i = 0$, $z_{i,j} = 0$) in the training data, so we stratify samples by labels with the ratio of positive:negative to be 1:3. For QDM, we tune the two parameters used in the clustering algorithm Dia_{max} (the diameter threshold for a cluster) and W_{min} (the weight threshold for a valid cluster), as well as two parameters used for selecting facet terms in each facet ($S_{t|F} > \alpha |Sites(F)|$ and $S_{t|F} > \beta$). For pLSA and LDA, we tune the number of facet terms in a query facet. For QF-I, we tune the weight threshold for facet terms, w_{min}, and the diameter threshold, d_{max}. For QF-J, there are no parameter need to be tuned. We returned top 10 query facets from all the five models in all evaluation.

8.2 Finding Facet terms

To evaluate effectiveness in finding facet terms, we tune all the models on wFT, which combines both precision and recall, and takes into account facet term weighting.

Table 7: Facet term precision, recall and F1 tuned on wFT

Model	P	wP	R	wR	FT	wFT
pLSA	0.284	0.385	0.562	0.561	0.351	0.430
LDA	0.292	0.394	0.595	0.593	0.364	0.446
QDM	0.407	0.523	0.378	0.388	0.360	0.420
QF-I	0.347	0.458	**0.644**	**0.652**	0.427	**0.514**
QF-J	**0.426**	**0.534**	0.525	0.526	**0.449**	0.511

Table 8: Average number of facet terms in output per query for different models

Model	#terms/query
pLSA	153.1
LDA	154.8
QDM	68.0
QF-I	153.7
QF-J	97.8

Table 7 shows facet term precision, recall and F1 and their weighted version described in Section 7.2. QF-I and QF-J perform relatively well for both precision and recall. Their improvements over the other three models shown are all significant ($p < 0.05$, using paired t-test), except the improvements of QF-J over QDM for P and wP. The two topic model based approaches, pLSA and LDA, have relatively high recall and low precision. Contrarily, QDM has high precision and low recall. This difference can be explain by Table 8, which gives the number of facet terms output per query from each models. QDM only outputs 68 facet terms per query, while pLSA and LDA both output over twice that number. One possible reason for the low precision of pLSA and LDA is that they select facet terms solely according to term probabilities in the learned topics (query facets in our case) and do not explicitly incorporate query relevance. We find most of their facet terms are frequently-occurring list items, which are not necessary relevant to the query.

While the number of facet terms QF-I outputs is similar to pLSA and LDA, QF-I obtain much higher precision and recall, likely due to the rich set of features used. Table 9 shows the five most important item features according to the absolute values of learned weights. Not surprisingly, list TF/IDF features which are used to capture the likelihood of being a coordinate term have relatively high weights, as well as some features that are used to capture query relevance, e.g. $TF.clueIDF$.

Table 9: Top 5 item features, ranked by absolute weights

Feature	Weight
listTF.listIDF	2.6424
listSF	2.1374
wDF	-1.0754
TF.clueIDF	1.0115
SF	0.6873

From Table 7, we also find the the weighted metrics are usually consistent with their corresponding unweighted metric. One exception is that QF-J performs better than QF-I in FT, but it does slightly worse than QF-J in wFT. This is likely to be caused by the high recall for QF-I, which may include more highly rated facet terms.

8.3 Clustering Facet terms

Table 10 shows clustering performance of the five models, which are tuned on wFP. The improvements of QF-I and QF-J over the other three models shown are all significant ($p < 0.05$, using paired t-test). pLSA and LDA do not perform well in clustering, which could be caused by data sparsity. There are on average 5159 candidate lists per query, but only 3.9 items per list.

Table 10: Facet clustering performance tuned on wFP

Model	Purity	NMI	FP	wFP
pLSA	0.793	0.524	0.230	0.229
LDA	0.773	0.511	0.227	0.226
QDM	0.871	0.565	0.367	0.380
QF-I	0.843	0.606	**0.408**	**0.410**
QF-J	**0.922**	**0.631**	0.352	0.346

Table 11: Weights learned for item pair features

Feature	Weight
listContextSim	1.4944
textContextSim	0.7186
listCooccur	0.0817
lengthDiff	0.0563

The better performance in clustering for QF-I and QF-J can be explained by their incorporating factors other than list item co-occurrence information. In Table 11, we list the weights learned for item pair features. Besides one item co-occurrence related feature, *listContextSim*, we also find that *textContextSim* has a relatively high weight. *textContextSim* is used to capture the similarity of the two list items using their surrounding text, so it can help to group two facet terms together even if they might not co-occur a lot in candidate lists. As an example, for the query *baggage allowance*, we find different airlines do not co-occur a lot in candidate lists, (e.g. *delta* and *jetblue* only co-occur twice), but they tend to have high *textContextSim* (e.g. $textContextSim(delta, jetblue) = 0.81$), and are therefore grouped together by QF-I and QF-J.

8.4 Overall Evaluation

To compare overall effectiveness of the five models, we tune all the models on wPRF, and the results are reported in Table 12.

Table 12: Overall performance tuned on wPRF

Model	wP	wR	wFP	wPRF
pLSA	0.353	0.630	0.229	0.309
LDA	0.358	**0.670**	0.225	0.311
QDM	0.523	0.388	0.253	0.319
QF-I	0.450	0.667	**0.399**	**0.444**
QF-J	**0.534**	0.526	0.346	0.417

Unweighted metrics are very similar to their corresponding weighted metrics in terms of conclusions, and are omitted due to space limitation. Results here are consistent with the results that were tuned on wFT or wFP. pLSA and LDA have high recall, but low precision and FP. QDM has relatively high precision, but low recall and FP. It has on average 68 facet terms per query as output, and fails to improve the overall effectiveness when including more facet terms in its output. QF-I and QF-J are among the best two models according to both PRF and wPRF.

Since wPRF does not account for facet ranking effectiveness, we also report fp-NDCG and rp-NDCG tuned on themselves in Table 13. QF-J gives the best performance for both fp-NDCG and rp-NDCG. The improvements of QF-I and QF-J over the other three models shown in the Table 12 and 13 are all significant ($p < 0.05$, using paired t-test), except the improvements of QF-J over QDM for wP and QF-I over QDM for fp-nDCG.

Table 13: fp-nDCG and rp-nDCG tuned on themselves

Model	fp-nDCG	rp-nDCG
pLSA	0.250	0.071
LDA	0.238	0.063
QDM	0.257	0.093
QF-I	0.290	0.157
QF-J	**0.336**	**0.193**

9. CONCLUSIONS

In this paper, we studied the problem of extracting query facets from search results. We developed a supervised method based on a graphical model to recognize query facets from the noisy facet candidate lists extracted from the top ranked search results. We proposed two algorithms for approximate inference on the graphical model. We designed a new evaluation metric for this task to combine recall and precision of facet terms with grouping quality. Experimental results showed that the supervised method significantly outperforms other unsupervised methods, suggesting that query facet extraction can be effectively learned.

10. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part under subcontract #19-000208 from SRI International, prime contractor to DARPA contract #HR0011-12-C-0016. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

11. REFERENCES

[1] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL-HLT'09*, pages 19–27, 2009.

[2] J. Allan and H. Raghavan. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of SIGIR'02*, pages 307–314, 2002.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *MACHINE LEARNING*, pages 238–247, 2002.

[4] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, July 2009.

[5] V. Dang, X. Xue, and W. B. Croft. Inferring query aspects from reformulations using clustering. In *Proceedings of CIKM '11*, pages 2117–2120, 2011.

[6] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman. Dynamic faceted search for discovery-driven analysis. In *Proceedings of CIKM '08*, pages 3–12, 2008.

[7] Z. Dou, S. Hu, Y. Luo, R. Song, and J.-R. Wen. Finding dimensions for queries. In *Proceedings of CIKM '11*, pages 1311–1320, 2011.

[8] Z. Harris. Distributional structure. *The Philosophy of Linguistics*, 1985.

[9] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING '92*, pages 539–545, 1992.

[10] L. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome research*, 9(11):1106–1115, 1999.

[11] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *Proceedings of SIGIR '12*, pages 305–314, 2012.

[12] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of ACL '03*, pages 423–430, 2003.

[13] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML '01*, pages 282–289, 2001.

[14] D. Lawrie, W. B. Croft, and A. Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of SIGIR '01*, pages 349–357, 2001.

[15] D. J. Lawrie and W. B. Croft. Generating hierarchical summaries for web searches. In *Proceedings of SIGIR '03*, pages 457–458, 2003.

[16] C. G. Nevill-manning, I. H. Witten, and G. W. Paynter. Lexically-generated subject hierarchies for browsing large collections. *International Journal on Digital Libraries*, 2:111–123, 1999.

[17] M. Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proceedings of WWW '07*, pages 101–110, 2007.

[18] M. Paşca and E. Alfonseca. Web-derived resources for web information retrieval: from conceptual hierarchies to attribute hierarchies. In *Proceedings of SIGIR '09*, pages 596–603, 2009.

[19] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP '09*, pages 938–947, 2009.

[20] P. Pantel and D. Lin. Discovering word senses from text. In *Proceedings of KDD '02*, pages 613–619, 2002.

[21] P. Pantel, D. Ravichandran, and E. Hovy. Towards terascale knowledge acquisition. In *Proceedings of COLING '04*, 2004.

[22] M. Pasca. Acquisition of categorized named entities for web search. In *Proceedings of CIKM '04*, pages 137–145, 2004.

[23] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. 1988.

[24] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of SIGIR '11*, pages 1043–1052. ACM, 2011.

[25] S. Shi, H. Zhang, X. Yuan, and J.-R. Wen. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of COLING '10*, pages 993–1001, 2010.

[26] K. Shinzato and K. Torisawa. Acquisition of categorized named entities for web search. In *Proceedings of RANLP '05*.

[27] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, Sept. 1999.

[28] R. Song, M. Zhang, T. Sakai, M. Kato, Y. Liu, M. Sugimoto, Q. Wang, and N. Orii. Overview of the ntcir-9 intent task. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 82–105, 2011.

[29] X. Wang, D. Chakrabarti, and K. Punera. Mining broad latent query aspects from search sessions. In *Proceedings of KDD '09*, pages 867–876, 2009.

[30] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *Proceedings of SIGIR '07*, pages 87–94, 2007.

[31] F. Wu, J. Madhavan, and A. Halevy. Identifying aspects for web-search queries. *J. Artif. Int. Res.*, 40(1):677–700, Jan. 2011.

[32] X. Xue and X. Yin. Topic modeling for named entity queries. In *Proceedings of CIKM '11*, pages 2009–2012, New York, NY, USA, 2011. ACM.

[33] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of WWW '10*, pages 1001–1010, 2010.

[34] H. Zhang, M. Zhu, S. Shi, and J.-R. Wen. Employing topic models for pattern-based semantic class discovery. In *Proceedings of ACL '09*, pages 459–467, 2009.