Semi-Supervised Learning of Semantic Classes for Query Understanding – from the Web and for the Web

Ye-Yi Wang
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA
yeyiwang@microsoft.com

Raphael Hoffmann
Dept. of Computer Science
University of Washington
Seattle, WA 98195, USA
raphaelh@cs.washington.edu

Xiao Li and Jakub Szymanski
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA
{xiaol,jakubs}@microsoft.com

ABSTRACT

Understanding intents from search queries can improve a user's search experience and boost a site's advertising profits. Query tagging via statistical sequential labeling models has been shown to perform well, but annotating the training set for supervised learning requires substantial human effort. Domain-specific knowledge, such as semantic class lexicons, reduces the amount of needed manual annotations, but much human effort is still required to maintain these as search topics evolve over time.

This paper investigates semi-supervised learning algorithms that leverage structured data (HTML lists) from the Web to *automatically* generate semantic-class lexicons, which are used to improve query tagging performance – even with far less training data. We focus our study on understanding the correct objectives for the semi-supervised lexicon learning algorithms that are crucial for the success of query tagging. Prior work on lexicon acquisition has largely focused on the precision of the lexicons, but we show that precision is not important if the lexicons are used for query tagging. A more adequate criterion should emphasize a trade-off between *maximizing the recall* of semantic class instances in the data, and *minimizing the confusability*. This ensures that the similar levels of precision and recall are observed on both training and test set, hence prevents over-fitting the lexicon features. Experimental results on retail product queries show that enhancing a query tagger with lexicons learned with this objective reduces word level tagging errors by up to 25% compared to the baseline tagger that does not use any lexicon features. In contrast, lexicons obtained through a precision-centric learning algorithm even degrade the performance of a tagger compared to the baseline. Furthermore, the proposed method outperforms one in which semantic class lexicons have been extracted from a database.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information search and retrieval—*Search process*; I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge Acquisition*; I.5.1 [**Pattern Recognition**]: Models—*Statistical*

General Terms

Algorithms, Theory, Experimentation, Performance

Keywords

Lexicon acquisition/set expansion, query understanding/semantic tagging, HTML lists, conditional random fields, semi-supervised learning, information extraction, model over-fitting

1. INTRODUCTION

One can greatly improve a user's search experience and boost a site's advertising profits by better understanding the user's intent. Query tagging – assigning a pre-defined semantic label to each query term – is an attempt towards this goal. For example, a product query may be tagged as follows:

canon	**powershot**	**sd850**	**digital**	**camera**	**silver**
|	|	|	|	|	|
Brand	*Model*	*Model*	*Type*	*Type*	*Attribute*

By understanding what users are looking for, one can provide the exact information they need. Indeed, much of this information may be found in the contents of relational databases, which are now indexed by most search engines. By automatically tagging queries with field information, one can use the index to directly retrieve the precise information a user needs. Furthermore, an improved understanding of query intent enables better decisions in the selection of contextual ads. For example, relevant local ads may be selected when query tagging identifies a city in the following query:

hotels	**in**	**white**	**swan**	**washington**
|	|	|	|	|
Category	*Other*	*City*	*City*	*State*

Query tagging [10] is a specialized form of information extraction (IE) or named entity recognition (NER), which is often performed by training statistical sequential labeling models, such as Conditional Random Fields (CRFs) [8]. CRFs generally outperform rule based systems by large margins, and usually rules can be incorporated as features into CRFs, so in practice CRFs reach at least similar performance as a rule-based system. Additionally, CRFs are better at ambiguity resolution by modeling context dependencies with a principled data-driven approach. Ambiguity is one of the most difficult problems in query tagging, as illustrated by the example above – "white swan" could denote an animal, arts and crafts or a city; "washington" could refer to a person, city, county or state. However, while supervised learning generates statistical models which disambiguate effectively, substantial human effort is required to create an annotated training set for learning. For product query tagging, this problem is exacerbated by the fact that queries change over time due to the emergence of new kinds of products, new brands, new models, new merchants, or new product features. Domain specific knowledge, such as phrase lists for concepts like *Brand* or *Model* (henceforth "semantic class lexicons"), has been shown to improve the generalization capability of statistical models, reducing the need for annotated data[21]. However, such knowledge is often not available and usually requires substantial human effort to compile and maintain, which may be too costly as information changes over time.

In this work, we compare two semi-supervised learning algorithms that leverage structured data (HTML lists) from the Web to generate semantic-class lexicons. The algorithms are based on the assumption that many HTML lists contain instances of a single concept. For example, if "powershot sd850" is labeled as *Model* in the training data and it is observed to co-occur with "rebel xsi" in many HTML lists, then "rebel xsi" is likely to be a *Model* as well. In our case, instance phrases of semantic classes (*Brand*, *Model*, etc.) are first extracted from annotated queries; the instances are then used as seeds in a graph learning algorithm to obtain additional instances of the same semantic classes. These sets of instances, or lexicons, enable statistical models to rely on less labeled training data – in fact we will show later that with only a few hundred of labeled queries plus the lexicons that are automatically obtained from the proposed algorithm, a query tagger achieves better performance than a model trained with tens of thousands of labeled examples without any lexicons. Because the lists on the ever-changing Web reflect the evolving nature of the contents of semantic classes, and the context patterns are relatively fixed over time – e.g., it is always more likely that a model follows a brand ("canon rebel") than a brand follows a model ("rebel canon"), incorporating lexicons learned from the Web into a statistical model also makes this technique adaptive to the query content shift as it occurs over time.

The algorithms discussed in this paper are not specific to queries; they can also be applied to other information extraction (IE) tasks (in fact, we have applied them to extract from natural language text). One requirement, however, is that the entities of interest are well represented on the Web. For product query understanding, the domain the present work focuses on, this is generally the case. The tag set contains nine labels illustrated in Table 1. We attempt to acquire lexicons for the first five classes that occur frequently in queries and Web lists.

We adapt two graph-learning algorithms to the task of lexicon learning and observe opposite impacts on query tagging accuracy. To understand the difference, we carry out a series of experiments, and find that traditional precision-centric lexicon learning is inadequate for query tagging – it causes a mismatch of lexicon coverage between the training and test data, and consequently results in a severe overfitting of the lexicon features in the statistical sequential labeling model. We show that a more adequate criterion should emphasize the *coverage* of the acquired lexicons while maintaining a *low level of confusability*. Here we prefer to have more entries acquired by the algorithm to cover the instances in the query tagging data (i.e., broad coverage). In the mean time, low level of confusability requires that an unambiguous phrase e has a high posterior probability $P(c\,|\,e)$ for the corresponding semantic class c, while the probability is lower if e is ambiguously labeled with different semantic tags in the query tagging data. If e does not occur often in the domain of interest, the value of the probability has little impact on the final query tagging accuracy – hence precision should not be the primary criterion of lexicon acquisition for query tagging. By increasing coverage while maintaining low level of confusability, similar precision and recall levels can be observed on both training and test data, which effectively avoids model over-fitting.

Table 1: Semantic classes (CRF labels) in product query tagging task

Semantic classes	Example use in queries
Brand	**canon** powershot sd850
Model	canon **powershot sd850**
Merchant	digital cameras at **best buy**
Type	**digital cameras** at best buy
Attribute	**10 megapixel** digit cameras
SortOrder	**cheapest** digital cameras
BuyingIntent	digital camera **sales**
ResearchIntent	digital camera **reviews**
Other	digital cameras **at** best buy

Although we study these effects in the context of query tagging, we believe that our discoveries generalize beyond our task and provide a guideline for future research on semi-supervised knowledge acquisition for information extraction and named entity recognition.

1.1 Related Work

Semantic class and relation acquisition is a well-studied topic. Much research leverages linguistic patterns to extract semantic classes and relations from free text [6, 2, 11, 14, 7]. In [4], an algorithm is introduced to learn semantic classes and named entity extraction patterns simultaneously. Recently there has been increasing interest in leveraging structured data from the Web to learn semantic classes and relations. In [20], a Web page wrapper induction algorithm is presented that learns language-independent patterns for semantic class lexicon expansion. In [5], both linguistic patterns and wrappers for lists are used to extract semantic class members. In [3], a Web-scale relational database is built by filtering HTML tables with statistical classifiers. The work described in [19] is closely related to ours. Like us, the authors use graph learning to acquire open-domain semantic classes by leveraging structured Web data, in their case, the *HTML tables* reported in [3]. Another closely related work is described in [18], where a context pattern induction algorithm is used to obtain lexicons, which in turn are used by a named entity recognition model.

In contrast to this existing work, which primarily focuses on the precision of the acquired semantic classes[1], we are

[1]Note that most of the related work learns semantic classes for the sake of semantic class acquisition, hence precision is

ambivalent about the precision. In our case, the semantic classes are only used as intermediate information to improve the query tagging accuracy and are hidden from the end users. Therefore, we consider over-generalization acceptable as long as it does not degrade the performance of query tagging. As shown later in the paper, this philosophical difference leads to different graph learning algorithms – with dramatically different results.

Finally, we note that there exist large bodies of work on Conditional Random Fields and graph-based learning. CRFs have been broadly applied to many natural language related tasks, including part-of-speech tagging [8], named entity recognition [12], information extraction [16], and parsing [17]. Graph-based learning [23, 24] has been used for various natural language processing tasks, as evidenced by many publications in the TextGraphs Workshops [1].

2. CONDITIONAL RANDOM FIELDS

CRFs [8] are conditional models with the following log-linear form defined with respect to a set of features $f_k(\mathbf{y}, \mathbf{x})$. The features are functions of the label sequence $\mathbf{y}$ and associated observation $\mathbf{x}$:

$$P(\mathbf{y} \mid \mathbf{x}; \Lambda) = \frac{1}{Z(\mathbf{x}; \Lambda)} \exp\left\{\sum_k \lambda_k f_k(\mathbf{y}, \mathbf{x})\right\} \tag{1}$$

Here $\Lambda = \{\lambda_k\}$ is a set of parameters. The value of λ_k determines the impact of the feature $f_k(\mathbf{y}, \mathbf{x})$ on the conditional probability. $Z(\mathbf{x}; \Lambda) = \sum_{\mathbf{y}} \exp\left\{\sum_k \lambda_k f_k(\mathbf{y}, \mathbf{x})\right\}$ is a partition function that normalizes the distribution. Given a set of m labeled training examples $(\mathbf{x_1}, \mathbf{y_1}) \ldots (\mathbf{x_m}, \mathbf{y_m})$, Λ can be learned by using stochastic gradient decent, L-BFGS or other numeric optimization algorithms to maximize the following objective function:

$$\sum_{i=1}^{m} \log P(\mathbf{y}_i \mid \mathbf{x}_i; \Lambda) - \frac{1}{2\sigma^2} \|\Lambda\|^2 \tag{2}$$

The second term in Eq. (2) regularizes the parameters to keep them from taking extreme values, thus preventing model over-fitting. Note that the objective function is a convex function, so a single global optimum exists.

The CRF in Eq. (1) is unconstrained in the sense that the feature functions are defined on the entire label sequence $\mathbf{y}$. Because the number of all possible label sequences is combinatorial, the model training and inference of an unconstrained CRF is very inefficient. To improve efficiency, it is common to restrict attention to linear-chain CRFs [8, 15], imposing a Markov constraint on the model topology, and restricting the feature functions to depend only on the labels assigned to the current and immediately previous terms, in the form $f(y_{t-1}, y_t, \mathbf{x}, t)$. These allow efficient dynamic programming algorithms for inference and model training – yet still support potentially interdependent features on observations via discriminative training of the conditional model.

3. SEMANTIC LEXICON LEARNING AND ITS APPLICATION IN QUERY TAGGING

Our technique for semi-supervised lexicon acquisition and query tagging proceeds in the following steps:

the most important metric.

Table 2: Examples of seed distributions

Seed	*Brand*	*Model*	*Merc.*	*Type*	*Attr.*	*Neg.*
camera	0.003	0.017	0	0.963	0	0.017
powershot	0	0.230	0	0.770	0	0
office depot	0.289	0	0.711	0	0	0

1. Extracting seed instances with an initial distribution over all semantic classes of interest from labeled training data.

2. Constructing a sub-graph from a phrase-list bipartite graph using the seed instance phrases.

3. Applying graph learning algorithms on the sub-graph to obtain a posterior distribution over all semantic classes of interest for each phrase node in the sub-graph.

4. Constructing stratified lexicons according to the posterior distributions for each semantic class.

5. Training the CRFs with features built on the stratified lexicons.

Step 1, 2, and 4 will be described in this section. Step 3 will be devoted a separate section afterwards. Step 5 will be explained in Section 5.1.

3.1 Extracting Seed Instances

The training set for CRF query tagging contains queries manually tagged with labels like the examples in Section 1. Seed instances for each semantic class of interest are extracted from it. A sequence of terms with the same semantic class labels is combined as an instance phrase. For the example in Section 1, seed instance "canon" can be extracted for *Brand*, "powershot sd850" for *Model*, and "digital camera" for *Type*. Phrases can often ambiguously belong to multiple semantic classes. This may be due to the fact that they are intrinsically ambiguous ("apple" can be a *Brand* or a *Merchant* as the Apple Store) or due to annotation mistakes – the majority of occurrences of "powershot" was mislabeled as *Type* instead of the correct label *Model*. Instead of assigning a single semantic class to a phrase, each phrase is therefore associated with a distribution over the set of classes – for the product query tagging task, the distribution is over the five classes at the top of Table 1, plus an additional "Negative" class representing the union of the remaining four semantic classes, for which we do not learn the separate lexicons to use as features in the CRF. Table 2 shows some examples of the extracted seeds with their distributions.

3.2 Constructing the Sub-Graph

A set of HTML lists (contents of <ol> or <ul> tags) are crawled from the Web and heuristically cleaned to remove the SPAM lists or lists used to format Web page layout. The resulting set includes about 55 million lists, which contain about 61 million unique phrases. From the lists a bipartite graph can be built, as illustrated by Figure 1, where an edge connects a phrase node with a list node if the phrase is an item in the list. The edges have the uniform weight of 1.0.

Given a set of seeds with their class distributions, a sub-graph is constructed. First, all the list nodes are marked with the number of items that exactly match a seed phrase.

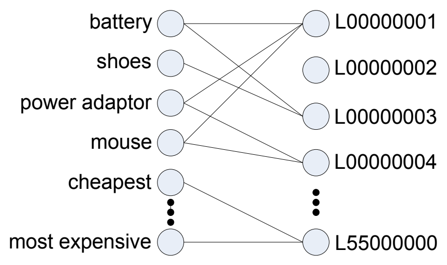

Figure 1: Bipartite list graph. On the left are the unique phrases, on the right are the HTML lists

The list nodes are pruned if their seed count is under a threshold[2]. Then the phrase nodes are marked with the number of the surviving lists they belong to. The same threshold is used to prune out the phrase nodes that do not appear in a sufficient number of lists.

3.3 Constructing Stratified Lexicons

The graph learning algorithms described in the next section produce $P(c \mid v)$, a distribution over semantic classes c for a phrase node v in the graph. While the probabilities can be directly used as continuous features in CRFs, they are not very reliable due to the noisy nature of the lists. Instead, phrases are binned to multiple stratified lexicons for a semantic class c according to their distributions – phrase p (represented by v_p in the graph) is included in the lexicon c_k for class c if $1 - P(c \mid v_p) \in [(k-1)/10, k/10)$. For example, if $P(\textit{Brand} \mid \text{"sony ericsson"}) = 0.85$, the phrase "sony ericsson" will be placed in the lexicon $\textit{Brand}_2$. This binning strategy produces lexicons $c_1, \ldots, c_{10}$ for a semantic class c.

Seed phrases may not exist in the list graph. In this case, they can be either included in the lexicons according to their initial distribution, or not used at all. We will address this in Section 5.

4. GRAPH LEARNING ALGORITHMS

We present two graph learning algorithms here. The first was proposed in [19] and was used for semi-supervised semantic class learning. The second is based on the algorithm for semi-supervised learning of query classification proposed in [9]. While the former can operate on arbitrary weighted graphs, the latter was specifically designed for weighted bipartite graphs. None of the two algorithms was originally used to obtain lexicon features for a sequential tagger. Both algorithms have the same computational complexity – each iteration is temporally and spatially linear to the number of edges in the graph.

4.1 Algorithm I

In [19], a high precision learning algorithm was applied to acquire semantic classes from free text, based on distributional similarity and template patterns. The classes were subsequently expanded based on a bipartite graph constructed from HTML tables. To improve the precision of the semantic class, the algorithm introduces an additional pseudo-class $\perp$, which represents "lack of information for semantic class assignment." All nodes in the graph that are not in the seed set are assigned an initial distribution $L^{\perp}$ with $L^{\perp}(\perp) = 1$.

```
Algorithm I
Input: $G = (V, E, W)$, seeds $V' \subseteq V$,
    seed distributions $\{F_v^0 : \forall v \in V'\}$
Output: Distribution $\{F_v^* : \forall v \in V\}$
1. $F_v^0 = L^{\perp} = (0, 0, ..., 1), \ \forall v \in V - V'$
2. repeat
3.   for all $v \in V$ do
4.     $C_v = \sum_u W(u,v) F_u^{i-1} / \sum_u W(u,v), \quad \forall u \in V$
5.     $F_v^i = p_v^{cont} \times C_v + p_v^{inj} \times F_v^0 + p_v^{abnd} \times L^{\perp}$
6. until the sequences $F_v^i$ converges to $F_v^*, \ \forall v \in V.$
```

The algorithm can be interpreted as as combination of distribution propagation and random graph walks. Here W is a square matrix, $W(u,v)$ stands for the weight on the edge $(u,v) \in E$. C_v is the influence on the distribution of the node v from its adjacent nodes in the graph, which is normalized by the sum of all incoming edges' weights to make it a proper probabilistic distribution. The parameters p_v^{cont}, p_v^{inj} and p_v^{abnd} ($p_v^{cont} + p_v^{inj} + p_v^{abnd} = 1$) respectively represent the probabilities of three different actions that can be taken at v during the random walk – continue the walk to an adjacent node; stay at current node v; or stop the walk. They can be set with heuristics based on a node's fan-out entropy [19].

4.2 Algorithm II

The second graph learning algorithm stems from [23], and was applied to semi-supervised learning of query classification by leveraging query-click graph[9]. Instead of performing learning on generic graphs, this algorithm focuses on bipartite graphs: The nodes of the graph can be partitioned to two sets, I and L, such that there is no edge between any two nodes in the same set. The weight matrix W is $|I| \times |L|$ instead of the square matrix in Algorithm I. The algorithm listed below first "normalizes" W to $B = D^{-1/2}W$. Here D is a diagonal matrix in which $d_{i,i}$ equals the sum of all elements in the i^{th} row (or column) of WW^T. Intuitively, $d_{i,i}$ is the "volume" of all length-of-two paths starting at node v_i. Since the graph is bipartite, this ensures that the propagation of distributions from phrases to lists then back to phrases will not introduce additional probability mass.

```
Algorithm II
Input: $G = (I \cup L, E, W)$, seeds $V \subseteq I$,
    seed distributions $\{F_v^0 : \forall v \in V\}$
Output: Distribution $\{F_v^* : \forall v \in I\}$
1. $F_v^0 = (0, 0, ..., 0), \ \forall v \in I - V$
2. $B = D^{-1/2}W.$
3. repeat
4.   $H_v^i = \sum_{u \in I} B(u,v) F_u^{i-1} \quad \forall v \in L$
5.   normalize $H_v^i, \ \forall v \in L.$
6.   $F_v^i = (1-\alpha) \sum_{u \in L} B(v,u) H_u^i + \alpha F_v^0, \ \forall v \in I$
7.   normalize $F_v^i, \ \forall v \in I.$
8. until the sequences $F_v^i$ converges to $F_v^*, \ \forall v \in I.$
```

The algorithm first propagates the distribution of the nodes $u \in I$ (list entries) to the nodes $v \in L$ (HTML lists) to obtain H_v^i, a semantic class distribution for v, then propagates back from the distribution of nodes in L to F_v^i, the distribution of $v \in I$. The parameter α regularizes the graph-learning. Unlike the F_v^i's in Algorithm I, neither H_v^i nor F_v^i is a probabilistic distribution right after the distribution propagation

[2]2 was used in the experiments reported, which yielded better results than without pruning.

(line 4 and 6), so they need to be normalized inside the repeat loop.

Algorithm I and II are similar in the sense that both are learning multiple competing semantic classes simultaneously via distribution propagation. Algorithm II differs from Algorithm in the following three aspects:

1. *No inclusion of $\perp$ as a dummy competing semantic class in Algorithm II.* Because precision is its major performance metric, Algorithm I is more conservative when the evidence for introducing a new instance to a class is not strong. In contrast, Algorithm II is conservative in adding a phrase to a class only if it can potentially be an instance of a competing class. This leads to a difference in the size of acquired lexicons, as we will discuss later.

2. *Weight normalization.* The matrix in Algorithm I is not pre-normalized. The influence of a phrase node on a list node is asymmetric to the influence of the list node on the phrase node. Algorithm II performs matrix pre-normalization in line 2, the influence of a phrase node on a list node is symmetric to the influence of the list node on the phrase node.

3. *Fewer parameters to set for graph-based learning.* Algorithm I uses node specific parameters p_v^{cont}, p_v^{inj} and p_v^{abnd}, whereas Algorithm II has a single regularization weight α.

In practice, we do not learn with Algorithm II till full convergence. We noticed that too many (around 100) iterations make the tagging results significantly worse – this is related to the over-fitting problem that will be discussed in Section 5.3. In the experiments reported in the next section, we stopped at iteration 5 and used 0 for α. Because of the early stop of graph learning, the effect of the value of α is not significant as long as it is not too far away from the small value (0.01) suggested in [22].

5. EXPERIMENTS AND EVALUATIONS

This section compares the effectiveness of the two semi-supervised lexicon learning algorithms for the purpose of query tagging. A series of experiments are then conducted to explain the performance difference between the two algorithms. The experiments lead to the discovery of a more adequate lexicon learning objective that is crucial for effective sequential labeling.

5.1 Data and Experiment Setup

We have conducted experiments with a data set of product search queries logged by a commercial search engine, which was manually labeled by annotators. Since much of the data was labeled using Mechanical Turk, it is very noisy and contains many inconsistent labelings. The test sets have been examined and corrected by an independent expert annotator, while the training sets have not been altered.

We compared tagging accuracy on the test set for three different conditions: using CRFs without lexicon features, using CRFs with lexicon features obtained by Algorithm I, and using CRFs with lexicon features obtained by Algorithm II. For some product categories including Computing and Electronics (C&E) and Clothing and Shoes (C&S), structured databases are available, from which one can directly extract lexicons for each semantic class. In addition to the above conditions, we also compared the performance of a tagger with the semi-automatically acquired lexicons to a tagger with lexicons extracted from these databases on the datasets of these two categories. Table 3 shows summary statistics of these datasets.

Table 3: The size of the training and test data

	Training Set		Test Set	
	# Samples	# Words	# Samples	# Words
Overall	27410	239362	4420	23476
C&E	14843	134362	669	3572
C&S	4122	36774	898	4804

The baseline tagging model is a linear chain CRF. It uses the state transition features $f_{i,j}^{TR}$ to capture the influence of context on the tag assigned to a given word:

$$f_{i,j}^{TR}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(y_{t-1} = i)\delta(y_t = j) \qquad (3)$$

where i, j are states (labels) of the model. It is reported in [10] that the transition features play a very important role in improving the tagging accuracy for the task.

In addition, the baseline tagger uses unigram $f_{w,j}^{UG}$ and bigram $f_{w,w',j}^{BG}$ features to capture the dependency on the identity of the current (and previous) words:

$$\begin{aligned} f_{w,j}^{UG}(y_{t-1}, y_t, \mathbf{x}, t) &= \delta(x_t = w)\delta(y_t = j) \qquad (4) \\ f_{w,w',j}^{BG}(y_{t-1}, y_t, \mathbf{x}, t) &= \delta(x_{t-1} = w)\delta(x_t = w')\delta(y_t = j) \end{aligned}$$

where w, w' are words and j is a model state (label).

In conditions other than the baseline, lexicon features (*a.k.a.* word cluster features) are also included:

$$f_{L,j}^{LEX}(y_{t-1}, y_t, \mathbf{x}, t) = \delta\left(L \Rightarrow [x_t]\right)\delta\left(y_t = j\right) \qquad (5)$$

where L is a lexicon, $L \Rightarrow [x_t]$ means that an entry in L is a substring of $\mathbf{x}$ that covers x_t. j is a label.

The performance of CRF tagging is measured by word level labeling accuracy – the percentage of words that are assigned the correct labels by the model. This measure was chosen since we found that it correlated well with assessments of end-to-end search quality.

5.2 Results

Unless mentioned otherwise, the experiment results in this section were obtained by excluding the seed phrases that do not exist in the bipartite graph from the learned lexicons.

In the first experiment, we examine the contributions of lexicons at different strata to the tagging accuracy. Figure 2 shows the test set word level query tagging accuracy as different numbers of lexicon strata are included in the model for each semantic class. When only the top stratum (which contains the highest posterior instances) is included, there is little difference between Algorithm I and Algorithm II. As more strata of lexicons are used, the CRF with stratified lexicons learned by Algorithm II achieves improved word level tagging accuracy. The accuracy plateaus after the seven top strata of lexicons are included. In contrast, the performance of the CRF with lexicons learned by Algorithm I does not change much as more strata of lexicons are included. While the number of lexicon strata has little impact on tagging accuracy after the seventh stratum, it does make a practical difference with respect to running time and memory

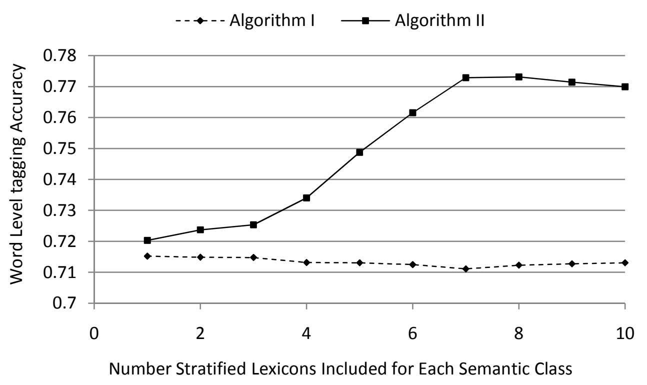

Figure 2: Word level tagging accuracy in relation to the number of strata of lexicons included in the CRF model for each learned semantic class

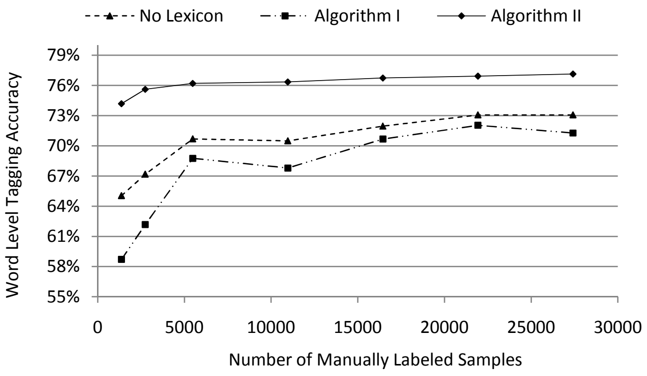

Figure 3: Word level tagging accuracy of the CRFs trained without lexicon features; with lexicon features learned by Algorithm I or Algorithm II

usage. For example, about 1/3 of entries are at the lowest stratum in the lexicons learned by Algorithm II – getting rid of them can significantly reduce the memory footprint. Also, note that the lowest stratum only contains phrases that have been assigned an extremely small posterior of $<$ 0.1, and collectively the ten lexicon strata contain all phrases in the pruned bipartite graph. For these reasons, we drop the lowest stratum from all experiments, and only use the top nine lexicon strata.

Figure 3 compares CRF word level tagging accuracy using no lexicon features, using lexicons learned by Algorithm I, and using lexicons learned by Algorithm II as different numbers of labeled samples are used for obtaining seeds and training the CRF. Lexicons learned by Algorithm II improve the accuracy by 4% to 9% absolutely over the model using no lexicon features (up to 25% relative error reduction); whereas the lexicons learned by Algorithm I degrade the tagging accuracy. Another important observation can be made in Figure 3: trained with only 5% of the training data, the CRF using the lexicon features obtained by Algorithm II has already outperformed the CRF trained with all training data but without using the lexicon features. Algorithm II can therefore greatly reduce the workload of data annotation.

In the next experiment, we compare the performance of the CRFs using lexicons extracted from the structured database and lexicons learned with each of the two graph learning algorithms on the C&E and C&S test data. Figure 4 plots the word level tagging accuracy in relation to the number of training samples used. For the C&E category, the lexicons extracted from the database perform at a similar level as the lexicons learned by Algorithm II. As the training data increases, the model using no lexicon features achieves a similar level of accuracy as the models that use the lexicon features. This is mainly due to the fact that there are enough training samples and the diversity of interests in different products in C&E is not as significant as in other categories. The lexicons learned by Algorithm I, however, degrade performance. For the C&S category, both the CRFs using lexicon features with the lexicons acquired by Algorithm II or extracted from the database achieve significantly improved tagging accuracy over the baseline model. The lexicons learned by Algorithm I, again, hurt accuracy. The models using lexicons from Algorithm II have additional gains over the models using the database lexicons – about 2% absolute in most cases.

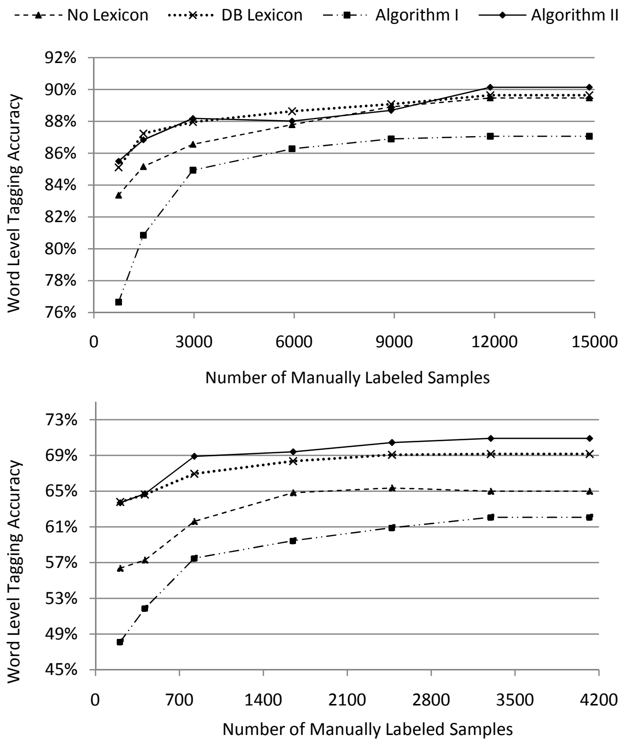

Figure 4: Word level tagging accuracy of the CRFs without lexicon features; with lexicon features, where lexicons were either obtained from structured database, learned by Algorithm I or II. Top chart: C&E. Bottom chart: C&S category

Finally, it is also interesting to note that the machine learning based approaches have superior performance to a rule based system: Using only the entries from the database as rules and a disambiguation strategy based on the prior probability of semantic classes, the word level tagging accuracy reaches 54.8% for the C&E test set and 48.4% for the C&S test set – far less than the 90% and 69% obtained by the CRFs.

Table 4: Number of lexical phrases for different semantic classes for the C&E and C&S categories

		DB Lexi.	Algo. I	Algo. II
C&E	*Brand*	3752	1150	77056
	Merchant	29	67	1679
	Model	91267	1945	43829
	Type	3133	3019	143563
	Attribute	2792	3166	273464
C&S	*Brand*	1069	609	56698
	Model	2827	367	16263
	Type	1070	623	54569
	Attribute	648	1140	122894

5.3 Analyses

What are the key factors that lead to the different behavior of Algorithm I and II? It was a surprise that Algorithm I degraded the accuracy of the baseline CRFs using no lexicon features. In this subsection, as an aftereffect, we compare the lexicons acquired by the two algorithms with the seeds extracted from the entire training sets. The first difference we noticed is the collective size of the top nine strata of lexicons learned by the two algorithms. Table 4 shows the size of the lexicons for different semantic classes for the C&E and C&S categories. Compared to the lexicons extracted from the database or learned by Algorithm I, the lexicons learned by Algorithm II are at least an order of magnitude larger (the only exception is C&E Model for which the database is extremely comprehensive). The semantic lexicons learned with the overall data have similar patterns as with the C&E and C&S categories.

The lexicon size of a semantic class has direct impact on the classifier's performance on the semantic class. Table 5 shows the precision, recall and F1 score on each semantic class, of the model trained with all training data. When the lexicon size is too small (e.g., Merchant), the performance (especially recall) suffers. Table 6 shows the number of semantic class instance occurrences and types (multiple occurrences of the same instances are counted once) in the test set, as well as the coverage of different lexicons for these instances. While coverage explains the superior performance of Algorithm II, it is clearly not the only factor since Algorithm I learned lexicons have similar (or better) coverage as the lexicons extracted from the database for C&E and C&E, yet their performance is much worse than the DB lexicons. Furthermore, the learned lexicons are quite different from the DB lexicons even though they have similar performance as in the case of the DB and Algorithm II learned lexicons for the C&E category – they do not overlap much as illustrated by the last two rows in Table 6. However, the Algorithm II learned lexicons cover almost all the test set instances covered by the DB lexicons, which suggests that Algorithm II is good at acquiring the popular semantic class instances that occurs frequently in the test set.

What are the other factors that affect the tagging performance? Naturally question may be raised about the quality of the learned lexicons, especially the huge one learned by Algorithm II. We believe that a more fundamental question is what metrics the quality should be measured against. The traditional precision-centric metrics may be irrelevant: Since the lexicons are hidden from users, over-generalizing a lexicon may not affect the final tagging accuracy – as long

Table 5: The performance of each semantic class

Semantic classes	**Precision**	**Recall**	**F1**
Brand	0.59	0.84	0.69
Model	0.67	0.54	0.60
Merchant	0.42	0.06	0.10
Type	0.85	0.84	0.84
Attribute	0.66	0.28	0.39

Table 6: Test set semantic class instance coverage

Lexicon	C&E			C&S		
	Size	Coverage		Size	Coverage	
		Type 736	Occ. 1195		Type 945	Occ. 1597
DB	100973	229	570	5614	209	754
Algo I	9347	392	833	2739	205	761
Algo II	539591	606	1028	250424	683	1248
DB∩I	1295	159	498	489	123	624
DB∩II	11241	210	551	2991	203	746

as it does not cause confusion in discriminating among the semantic classes of interest to the current task (here this includes the agglomerated "Negative" class).

If only the lexicons are used for query tagging without other features, they can keep growing as long as their growth brings more benefit than harm to the query tagger. Here, the benefit is the number of semantic class instances in the test set that are correctly covered by the lexicon (true positives, TP). The harm is the mislabeling of a word sequence as a semantic class due to overgeneralization (false positives, FP). Figure 5 shows the FP/TP ratio in relation to the number of stratified lexicons of each semantic class included – the statistics are collected by simply using the lexicons as matching rules in a rule-based system. No CRFs are used in this and the next experiment for precision/recall. We also include the lexicons obtained from the structured database; since these are not stratified, their FP/TP ratio is a horizontal line. The curve for Algorithm I is much flatter than that for Algorithm II. That implies that the precision of the lexicons from Algorithm I is much higher than that from Algorithm II as more strata of lexicons are included. However, the ratio is still below 1 (benefit is greater than harm) for Algorithm II after the 8th stratified lexicon is included (which covered 61% of the total acquired semantic class instances for C&E and 58% for C&S.) This shows that Algorithm II expands the lexicon more aggressively (thus results in higher coverage for instances in the test queries). In the mean time it also maintains a low level of confusion. The statistics here directly correlate with the CRF's behavior in Figure 2 – accuracy keeps improving until around the seventh stratum of lexicons learned by Algorithm II is included in the model.

The capability of maintaining a lower confusion level during lexicon growth can be largely attributed to the fact that the learning algorithms propagate entire class distributions instead of only a single class' probability, as is common in other approaches such as PageRank [13]. Because competing semantic classes in a given task are learned simultaneously, a phrase that occurs in multiple lists containing seed phrases from different semantic classes will not be able to get a high posterior probability for a specific class. Consequently, intrinsically ambiguous phrases are less likely to end up in

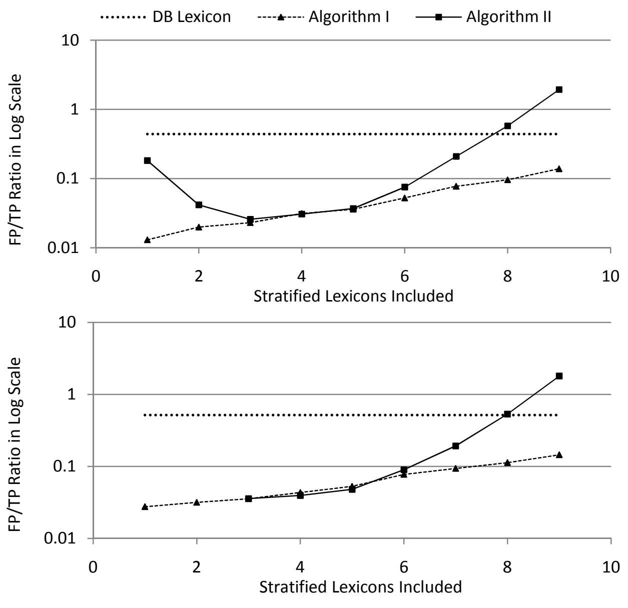

Figure 5: FP/TP ratio on the test set semantic classes for the lexicons extracted from database or learned with different algorithms. Top chart: C&E. Bottom chart: C&S

the top strata of lexicons. This has the advantage that in cases of ambiguity a CRF can put more emphasis on other contextual features, such as state transitions.

By varying the number of stratified lexicons included, we plot the training and test set semantic class instance recall/precision curves in Figure 6. The lexicon extracted from the database is represented by a single point in each of the charts for C&E and C&S, since it is not stratified. At the same precision level, the lexicons learned by Algorithm II have much higher recall of the test set semantic class instances than the lexicons extracted from the databases. With more strata included, the precision of the lexicons learned by Algorithm I is much higher. In fact, both precision and recall on test data do not change very much. This explains why tagging accuracy does not change much as more lexicons learned by Algorithm I are included in the CRF, as illustrated in Figure 2. In contrast, Algorithm II has a better recall on test data when more stratified lexicons are included. On the other hand, because the seed data is taken from the training sample, the learned lexicons have a high recall of semantic class instances on the training set. That high recall may not be available on the test set if the learning algorithm is conservative for better precision. This is exactly what we observe in Figure 6: the training data curve of Algorithm I has much larger recall than the corresponding test data curve, whereas the difference is much smaller with the curves for Algorithm II. In fact, in the case of the C&E data, the test data recall is often larger than that of the training data for Algorithm II. In summary, the precision/recall curves of the training data and test data are much more similar for the lexicons learned by Algorithm II than those learned by Algorithm I.

Note that the FP/TR ratios in Figure 5 and the precisions in Figure 6 do not reflect the ability of ambiguity resolution of the lexicons learned by Algorithm II – every match of a

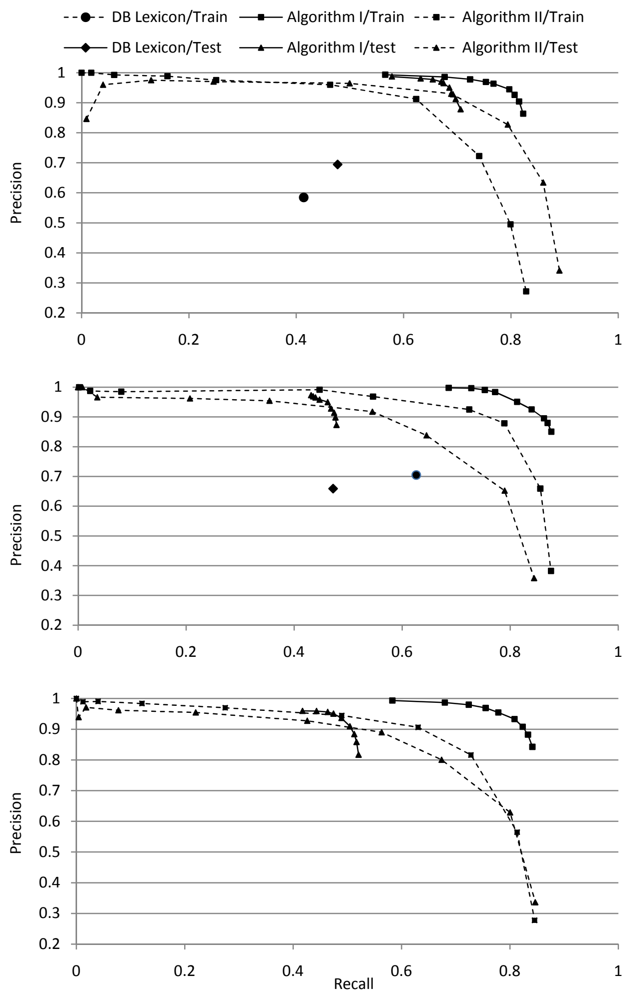

Figure 6: Recall/Precision curves of the lexicons. Top chart: C&E. Middle chart: C&S category. Bottom chart: overall data

test set instance by a lexicon of a wrong class was counted as an error without considering the competition from the lexicons of the correct semantic class. Table 7 compares the stratum of the correct semantic class lexicon covering a test set instance phrase with the highest stratum among the lexicons of competing semantic classes. It reports the number (and percentage) of times that the correct semantic class lexicon has higher, equal or lower rank (hence higher, similar or lower posterior probability) than the lexicons of competing semantic classes. In most cases, the correct semantic class has higher posterior probability. Therefore, the potential benefit of over-generalizing the lexicon is even larger than it seems to be in Figure 5.

The degradation of tagging accuracy with the lexicons learned by Algorithm I can be attributed to model overfitting. Because of the large discrepancy of recall between the training and test set in the case of Algorithm I, as illustrated by Figure 6, the learned models rely too much on the lexicon features and do not generalize well to the test data. The over-fitting of the lexicon features may depress the contribution of other important features to the final tagging decision, such as state transitions features. Table 8

Table 7: Stratum comparison of the ambiguous lexicons that cover a test set instance phrase. The lexicons of the correct semantic classes have higher ranks for 70% or more of test set instance occurrence than those of competing semantic classes

	Higher	Equal	Lower
Overall	4881 (70.5%)	621 (9.0%)	1424 (20.6%)
CE	925 (80.7%)	65 (5.7%)	156 (13.6%)
CS	1025 (69.3%)	157 (10.6%)	297 (20.1%)

compares the average absolute values of the weights for the lexicon features and those for other features learned by the CRF models with the DB lexicons, the lexicons learned by Algorithm I and the lexicons learned by Algorithm II. It is clear that the models based on the lexicons learned by Algorithm I have the highest ratios of lexicon feature weights and other feature weights, while the models based on the lexicons learned by Algorithm II have the lowest such ratios. The ratios correlate well with the tagging accuracy. It clearly indicates that over-fitting the lexicon features is the major problem with the models based on the lexicons learned by Algorithm I.

Table 8: Comparison of the average absolute values of the weights between the lexicon features and other features

		Lexicon	Others	Ratio
Overall	Algorithm I	0.530	0.129	4.109
	Algorithm II	0.430	0.249	1.727
C&E	Algorithm I	0.452	0.127	3.559
	Algorithm II	0.411	0.235	1.749
	DB Lexicon	0.556	0.209	2.660
C&S	Algorithm I	0.360	0.111	3.243
	Algorithm II	0.410	0.190	2.158
	DB Lexicon	0.651	0.226	2.881

We found that more restrictive regularization helped but was not able to bring the accuracy on par with the CRFs using no lexicon features – in fact the word level tagging accuracy we reported earlier was obtained with tuned regularization parameters. Surprisingly, the high precision makes the problem even worse! Since the higher precision is also observed in the training set, the learned model is more confident on relying on the lexicon features. In the extreme case, if all training set instances of semantic classes are covered by lexicons (100% recall) and the lexicon for each semantic class does not cover any training set instances of other semantic classes (100% precision), the learner will treat the lexicon features (match of lexicon entries) as the necessary and sufficient condition for a substring to be labeled as a semantic class – though the condition may not hold at all on the test set. This explains the observation that the CRFs using lexicon features based on the lower precision database lexicons outperform the higher precision lexicons learned by Algorithm I, when the two sets of lexicons have similar level of recall on test data.

Precision-centric learning is particularly harmful when the knowledge is acquired from HTML lists, due to the high noise level of the lists – focusing on precision with noisy inputs implies an even more conservative learning strategy

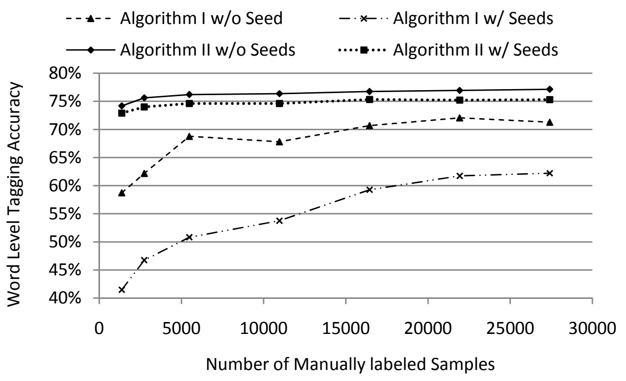

Figure 7: The effect of adding seed phrases not in the bipartite graph into the lexicons, for lexicons learned with Algorithm I and II

hence lower coverage of the test set instances. On the other hand, since the training set instances are used as seeds, they have a high chance to be covered by the learned lexicons. Therefore the mismatch between the training and test data becomes more severe.

The final experiment confirms that over-fitting is the culprit. In the experiment results reported so far, we have excluded the seed phrases that do not exist in the bipartite list graph from the lexicons. If we include those in the lexicons according to their initial distribution, we get 100% coverage of the training data semantic classes. This, on one hand, increases the coverage of the test set semantic classes, on the other hand, makes the over-fitting problem even worse. Figure 7 shows that adding these seed indeed degrades the performance for both algorithms on the entire test set. The degradation is more severe for Algorithm I because the precision of the lexicons is so high that the model almost learned to make decisions solely based on lexicon features.

N-fold cross-learning of lexicons and CRFs may alleviate the over-fitting problem of Algorithm I. However, preliminary results show little improvement for Algorithm II with 10-fold cross learning – over-fitting is not a big issue there.

6. DISCUSSIONS AND CONCLUSIONS

We applied two different semi-supervised graph learning algorithms to acquire semantic class lexicons from Web lists, and used the lexicons as features in CRFs for query tagging. One algorithm resulted in significant improvements in query tagging accuracy, and substantially reduced the human effort needed to manually label training data.

By comparing the behavior of two algorithms, we found that the precision-centric learning algorithms are not suitable for use in sequential labeling tasks, due to the problem of over-fitting. Instead, it is better to over-generalize the learned lexicons to result in a similar recall on the training and the test set, while maintaining a low level of confusion among the semantic classes of interest. This can be achieved by simultaneously learning lexicons of multiple competing classes via distribution propagation. We note that each of the two algorithms discussed was not designed for the purpose of lexicon acquisition for query tagging. Each can be enhanced to increase the recall of semantic class instances. While the present work compares existing algorithms adapted to our task of query tagging, we are planning to develop novel algorithms based on our insights in the future. For example, we can quantify "confusability" and include it in an objective function (together with lexicon coverage), such that new learning algorithms can be designed to directly optimize the objective function.

The major contribution of the present work lies in revealing the key factors in semi-supervised lexicon acquisition that make it successful in a sequential labeling task. While we studied these factors in the context of query tagging, we are confident that the conclusions extend beyond that. In fact, the lessons learned in this work provide a general guideline for future research on semi-supervised knowledge acquisition for sequential labeling tasks. Importantly, this applies to our finding that knowledge acquisition should adopt an objective that ensures that the knowledge acquired through partial-supervision from the training set has similar properties on the training set as well as an independent development set. Only then the sequential labeling model will not over-fit the knowledge obtained in a semi-supervised fashion.

7. ACKNOWLEDGMENTS

The authors would like to thank Alex Acero, Paul Viola and Dan Weld for their helps, comments and feedback on this work.

8. REFERENCES

[1] Textgraphs: Graph-based algorithms for natural language processing. http://www.textgraphs.org.

[2] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *the Proceedings of the 5th ACM Conference on Digital Libraries*, San Antonio, Texas, USA, 2000.

[3] M. J. Cafarella, A. Halevy, Z. D. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the Web. In *the Proceedings of VLDB*, Auckland, New Zealand, 2008.

[4] E. Eiloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *the Proceedings of the 16th National Conference on Artificial Intelligence*, 1999.

[5] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderl, D. S. Weld, and E. Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *the Proceedings of AAAI*, 2004.

[6] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *the Proceedings of the 14th Conference on Computational Linguistics*, 1992.

[7] M. Komachi and H. Suzuki. Minimally supervised learning of semantic knowledge from query logs. In *the Proceedings of IJCNLP*, Hyderabad, India, 2008.

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

[9] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *the Proceedings of the 31st SIGIR Conference*, 2008.

[10] X. Li, Y.-Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *the Proceedings of the 32nd SIGIR Conference*, 2009.

[11] D. Lin and P. Pantel. Concept discovery from text. In *the Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, 2002.

[12] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and Web-enhanced lexicons. In *the Proceedings of the 7th Conference on Natural Language Learning (CoNLL)*, Edmonton, Canada, 2003.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the Web. Technical report, Stanford InfoLab, 1999.

[14] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *the Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, Sydney, Australia, 2006.

[15] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *the Proceedings of Human Language Technology Conference and the Conference of North American Chapter of the Association for Computational Linguistics*, 2004.

[16] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *the Proceedings of Advances in Neural Information Processing Systems*, Vancouver, Canada, 2005.

[17] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *the Proceedings of Human Language Technology Conference and the Conference of the North American Chapter of the Association for Computational Linguistics*, 2003.

[18] P. P. Talukdar, T. Brants, M. Liberman, and F. Pereira. A context pattern induction method for named entity extraction. In *the Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, New York City, 2006.

[19] P. P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *the Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[20] R. C. Wang, N. Schlaefer, W. Cohen, and E. Nyberg. Automatic set expansion for list question answering. In *the Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[21] Y.-Y. Wang, A. Acero, C. Chelba, B. Frey, and L. Wong. Combination of statistical and rule-based approaches for spoken language understanding. In *the Proceedings of the International Conference on Speech and Language Processing*, Denver, Colorado, 2002.

[22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, volume 16, pages 321–328, 2004.

[23] D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems*, 2005.

[24] X. Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005.