# Extracting Structured Information from User Queries with Semi-Supervised Conditional Random Fields

Xiao Li, Ye-Yi Wang, Alex Acero
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{xiaol,yeyiwang,alexac}@microsoft.com

## ABSTRACT

When search is against structured documents, it is beneficial to extract information from user queries in a format that is consistent with the backend data structure. As one step toward this goal, we study the problem of *query tagging* which is to assign each query term to a pre-defined category. Our problem could be approached by learning a conditional random field (CRF) model (or other statistical models) in a supervised fashion, but this would require substantial human-annotation effort. In this work, we focus on a semi-supervised learning method for CRFs that utilizes two data sources: (1) a small amount of manually-labeled queries, and (2) a large amount of queries in which some word tokens have *derived labels*, *i.e.*, label information automatically obtained from additional resources. We present two principled ways of encoding derived label information in a CRF model. Such information is viewed as hard evidence in one setting and as soft evidence in the other. In addition to the general methodology of how to use derived labels in semi-supervised CRFs, we also present a practical method on how to obtain them by leveraging user click data and an in-domain database that contains structured documents. Evaluation on product search queries shows the effectiveness of our approach in improving tagging accuracies.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*; I.5.1 [**Pattern Recognition**]: Models—*Statistical*

## General Terms

Algorithms, Experimentation

## Keywords

Semi-supervised learning, conditional random fields, information extraction, metadata

*SIGIR'09,* July 19–23, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-483-6/09/07 ...$5.00.

## 1. INTRODUCTION

The World Wide Web is a large reservoir of information that is still growing at a rapid rate. Unlike data found in a database, the vast majority of the Web documents are only semi-structured, making information retrieval a challenging task. In recent years, there has been a surge of interest in extracting structured information from Web documents and converting the extracted data into database objects [2, 16, 20, 17]. Moreover, many vertical search engines crawl data directly from a relational database, and their indexes contain highly structured information that is less ambiguous in nature. Such data representations, if appropriately utilized, can greatly help enhance search experience.

When search is against structured data, it is beneficial to extract information from user queries that is explicitly represented in a structured form. In this work, we study the problem of *query tagging* as one step toward this goal. More specifically, we view a query as a sequence of word tokens; given a set of pre-defined fields (or states), our aim is to assign each word token a label indicating which field it belongs to. In particular, we focus our attention on tagging product search queries, since this is one representative domain where structured information can have a substantial influence on search experience. Below is an example showing a product search query annotated with field information. The complete schema of field definition is given in Table 1 with details given in Section 5.

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

For our task, it is possible to construct (from a database) field-dependent lexicons that enumerate possible values for each field. However, a pure lexicon-lookup based approach is insufficient largely due to the presence of ambiguous words. For example, the word "red" is in most cases seen as *Attribute*, but can mean *Brand* in the context of "red hat". Another challenge is that users may formulate queries using out-of-lexicon words. For these reasons, we take a statistical approach to our problem for its known success in sequential labeling tasks. In particular, we use conditional random fields (CRFs) [10] that model probabilistic dependencies between two consecutive fields and between fields and observations. Indeed, despite the fact that user queries are commonly viewed as bags of words, we found that the field ordering in product search queries does display statistically significant patterns that can help sequential labeling.

If queries fully-annotated with field information were avail-

| Fields | Example use in queries |
|---|---|
| *Brand* | **canon** powershot sd850 |
| *Model* | canon **powershot sd800** |
| *Type* | canon **digital cameras** silver |
| *Attribute* | canon digital cameras **silver** |
| *Merchant* | digital cameras at **best buy** |
| *SortOrder* | **best** digital cameras |
| *BuyingIntent* | **buy** canon digital cameras |
| *ResearchIntent* | digital cameras **review** |
| *Other* | digital cameras **at** best buy |

**Table 1: A example schema that defines fields in product search queries. The top and bottom rows contain fields that correspond to product metadata and product-independent keywords respectively.**

able in abundance, our problem could be approached by learning a CRF model in a supervised fashion. However, supervised training data will hardly ever be sufficient in this space as the volume of web search queries is extremely large. Often times, while it is expensive to ask human annotators to label a large amount of queries, it is relatively easy to obtain "labels" for some word tokens from additional resources. We use the term *derived labels* to refer to partial labeling information obtained in this fashion. This should be distinguished from *manual labels* which are acquired via human annotation. Note that derived labels may be available only for a subset of word tokens in a query, and thus cannot be directly used as supervised training data. Moreover, such information is often noisy depending on the data source.

In this work, we investigate a semi-supervised learning method for CRFs that utilizes two sources of information: (1) a small amount of manually-labeled queries, and (2) a large amount of queries with derived labels obtained in an unsupervised fashion. Our goal, then, is to make use of these two data sources to learn a better CRF model. Our contribution is two-fold. First, we assume the availability of the second data source and explore two principled ways of incorporating such information into conditional random fields. In one setting, derived labels are considered as hard evidence on the corresponding state variables. In the second setting, derived labels are used as soft evidence in the form of a feature function, expressing preferences over values of state variables. Both approaches are expressed in a graphical model framework, and their respective optimization objectives are discussed.

Our second contribution is to present a practical method that generates derived labels for a large amount of queries with minimum human supervision. We demonstrate our approach in the product search domain. Suppose that we have access to user click data in the form of (*query*, *product title*) pairs, and to a product database that contains product titles and their corresponding field data. Then we are able to associate queries with their relevant field data. This enables us to automatically attach labels to some word tokens in queries, and to use them as derived labels in the proposed semi-supervised learning paradigm.

We evaluate our approach in the task of tagging product search queries, while our method is general enough to be applied to other domains (such as local search and sports search). Experiments show that our proposed approach can significantly increase tagging performance on practical data.

## 2. RELATED WORK

There are not many works on query tagging that we are aware of. The only slightly relevant work is by [3] on part-of-speech tagging of queries. Therefore, we review related works mostly from a machine learning perspective.

A good number of semi-supervised methods for CRFs have been published in recent years. One school of approaches makes the standard assumption that, in addition to a small amount of labeled data, there exists a large amount of unlabeled data. In this setting, it is natural to apply *self-training* [18] which trains a seed model using the labeled data, and iteratively uses high-confidence predictions on the unlabeled data to expand the training set. This approach, however, lacks a theoretical justification for optimality, unless certain non-trivial conditions are satisfied [1].

A common challenge to applying semi-supervised learning to CRFs is that the entire state sequences of unlabeled data are hidden. Since CRFs have a maximum conditional-likelihood objective, the Expectation-Maximization (EM) used in generative models [8] is not directly applicable. To solve this problem, *entropy minimization*, originally proposed by [7] and extended to CRFs by [9], aims to maximize the conditional likelihood of labeled data while minimizing the conditional entropy of unlabeled data. Another approach, referred to as *JESS-CM* [15], embeds a generative model (HMMs) into the CRF framework with an objective that can be iteratively optimized.

A second group of works makes an additional assumption that alternative label resources can be utilized in learning; and our work falls into this category. When unlabeled data have partial labeling information, it becomes possible to optimize a CRF model using the EM algorithm. This is essentially the first approach we explore in this work that treats derived labels as hard evidence. The idea is also akin to the methodology proposed by [14] to integrate hidden variables in CRFs. Another work, *generalized expectation* [6, 12], uses aggregated "derived label" information to regularize the conditional distributions of a state variable given **individual** features. Our second approach bears a resemblance to their idea in that derived labels are viewed as soft evidence to bias the values of state variables. The key difference is that we use such information in the context of an input sequence to provide local, as opposed to global, preferences over state hypotheses. In this regard, our method is analogous to the use of *virtual evidence* in directed graphical models [4]. But our goal is to optimize a discriminative model, instead of a generative model, that incorporates such evidence.

Additionally, there are a fairly large amount of works on using additional resources for semi- or un-supervised information extraction in general. Here we only mention the most relevant few. In [5], a database is used to create artificially-annotated training data or to train a language model for an HMM-based sequence labeler. A similar approach is used by [19] for CRF-based text segmentation. Both works resort to relational tables to create field labels for text segments. In our work, we additionally make use of click data to maximally reduce ambiguity in this process.

## 3. CONDITIONAL RANDOM FIELDS

Linear-chain CRFs have been widely used in sequential labeling tasks such as part-of-speech tagging and information extraction [10, 13]. We choose to apply CRFs to our task for its ability of incorporating arbitrary features functions on observations without complicating the training. Formally, we let $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ denote an input query of $T$ terms, and $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ the corresponding state (field) sequence. Each $y_t$ can take on a pre-defined categorical value. We further augment a state sequence with two special states: *Start* and *End*, represented by $y_0$ and $y_{T+1}$ respectively. The conditional probability $p(\mathbf{y}|\mathbf{x})$ is given by

$$p(\mathbf{y}|\mathbf{x}; \Lambda) = \frac{1}{Z(\mathbf{x}; \Lambda)} \exp \left\{ \sum_k \lambda_k \sum_{t=1}^{T+1} f_k(y_{t-1}, y_t, \mathbf{x}, t) \right\} \quad (1)$$

The partition function $Z(\mathbf{x}; \Lambda)$ normalizes the exponential form to be a probability distribution. $f_k(y_{t-1}, y_t, \mathbf{x})$ are feature functions, and $\Lambda = \{\lambda_k\}$ are their corresponding weights. There are typically two types of features used in first-order, linear-chain CRFs: *transition features* and *emission features*. A transition feature is a binary function that indicates whether a transition $(y_{t-1} = i, y_t = j)$ occurs, *i.e.*,

$$f_{i,j}^{TR}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(y_{t-1} = i)\delta(y_t = j) \quad (2)$$

An emission feature is a binary function that indicates whether a observation-dependent feature co-occurs with state $j$. For example, a *unigram feature* function is defined as

$$f_{w,j}^{UG}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(x_t = w)\delta(y_t = j) \quad (3)$$

where $w$ represents a unigram. In a more general form, $\delta(x_t = w)$ can be replaced with an arbitrary function on $\mathbf{x}$. Different forms of this function would express different characteristics of the input query.

Given a set of manually-labeled queries $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, we can estimate model parameters in a supervised fashion. This training paradigm is expressed in the graphical model language in Figure 1(a). Note that the decoding graph for CRFs is the same as Figure 1(a) except that the entire state sequence becomes hidden. In supervised training, we aim to estimate $\Lambda$ that maximizes the conditional likelihood of training data while regularizing model parameters:

$$J_1 = \sum_{i=1}^m \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \Lambda) - \frac{1}{2\sigma^2}\|\Lambda\|^2 \quad (4)$$

The objective can be optimized using stochastic gradient descent, generalized iterative scaling, or other numerical optimization methods.

## 3.1 Application to Query Tagging

When applying CRFs to query tagging and particularly product search query tagging, it is curious to ask if such a model is suitable for solving our problem. A major question one may raise is that CRFs assume a probabilistic dependency between two consecutive states, which may not have a strong presence in queries. If that were the case, an instance-based classifier that tags each word token independently would be more appropriate. We thus randomly selected 4.5K product search queries across different categories, and had them manually labeled based on the schema in Table 1. Interestingly, we found that the distribution of transition features is rather skewed in this dataset. For example, ($y_{t-1}$=*Type*, $y_t$=*End*) occurred in 80% of the queries, meaning that most queries in this dataset end with the field *Type*. This suggests that CRFs could be more effective than instance-based models in tagging product search queries, as will be supported by experimental evidence in Section 6.

In addition to transition features that are implicitly assumed by CRFs, we introduce three types of emission features for our task. First, we use *ngram features* including both unigrams and bigrams. A unigram feature has been defined in Equation (3); and a bigram feature can be defined in a similar way. Specifically, for a given state variable in the linear chain, we use the current word and its preceding word to form a bigram, *i.e.*,

$$f_{w,w',j}^{BG}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(x_{t-1} = w)\delta(x_t = w')\delta(y_t = j) \quad (5)$$

We can also use the bigram that consists of the current word and its following word or use both types of bigrams simultaneously, but did not observe significant performance difference with these alternatives. The use of bigrams offers contextual information that is helpful in word disambiguation. Consider the examples in Table 1. The word "buy" is typically seen as *BuyingIntent*, but most likely means *Merchant* in the context of "best buy" (same with the earlier example "red" and "red hat"). When used in a CRF model, each ngram feature is assigned a separate weight, providing fine-grained information for sequential labeling. However, this can cause overfitting if the training data is sparse.

To improve the generalization ability of our model, we introduce a second type of features referred to as *regular expression* (regex) *features*:

$$f_{r,j}^{REGEX}(y_{t-1}, y_t, \mathbf{x}, t) = \delta(x_t \sim r)\delta(y_t = j) \quad (6)$$

where $x_t \sim r$ means that $x_t$ matches the regular expression $r$. For example, *sd700*, *sd800* and *sd850* all match the regular expression "[a-z]+[0-9]+" (in the pattern matching language). This can be useful in representing word tokens that correspond to fields like *Model* and *Attribute*. Furthermore, we introduce *lexicon features* which are given by

$$f_{L,j}^{LEX}(y_{t-1}, y_t, \mathbf{x}) = \delta(x_t \in L)\delta(y_t = j) \quad (7)$$

Here $L$ denotes a lexicon of words or phrases, and this feature is activated if $x_t$ occurs in that lexicon. Field-dependent lexicons, *e.g.*, a *Brand* lexicon, can be extracted from a product database, enumerating possible values for each field. The advantage of using such features is that they generalize to words that do not occur in the training data.

# 4. SEMI-SUPERVISED LEARNING WITH DERIVED LABELS

The last section gave an overview of CRFs in a supervised learning paradigm. While it is not always feasible to have a large amount of manually-labeled data, it is often easy to automatically obtain derived labels for some word tokens. Here we make the following distinction between manual and derived labels:

- *Manual labels*, denoted by $\mathbf{y} = (y_1, y_2, \ldots, y_T)$, are obtained via human annotation.
- *Derived labels*, denoted by $\mathbf{z} = (z_1, z_2, \ldots, z_T)$, are obtained automatically from additional resources. They can be available only for a subset of word tokens. We use $z_t$ = *null* to represent missing labels.

As one can imagine, derived labels can be valuable to learning since they may offer information, often in a vast amount, complementary to that provided by manual labels. We will leave the discussion of how to obtain derived labels

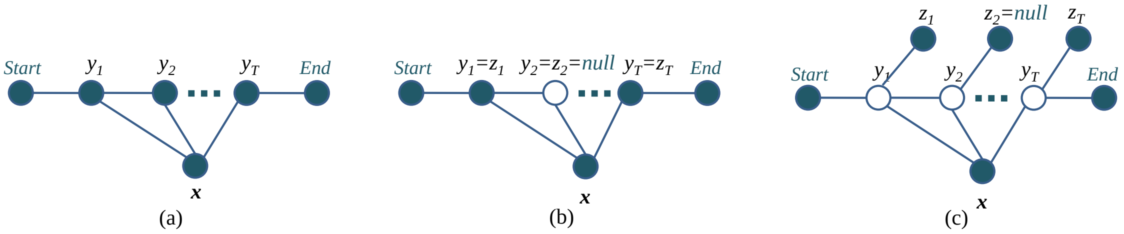


**Figure 1: Graphical model representation of CRFs for three different *training* settings: (a) supervised CRF; (b) semi-supervised CRF with derived labels as hard evidence; (c) semi-supervised CRF with derived labels as soft evidence. Solid and empty nodes denote observed and hidden variables respectively.**

for a specific domain in Section 5. In this section, we present a general methodology of incorporating derived labels in CRFs for semi-supervised learning. For this we assume the availability of two data sources: (1) a set of manually-labeled samples, denoted by $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$, and (2) a set of samples with derived labels, denoted by $\{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=m+1}^{n}$. Our goal, then, is to learn a CRF model leveraging these two sources of information.

## 4.1 Derived Labels as Hard Evidence

One natural solution is to treat derived labels the same as manual labels, and use them as hard evidence on state variables. Specifically, for queries in the second data source, we assume that a state variable is observed with value $y_t = z_t$, if $z_t \neq null$, and is hidden otherwise. This training setting is depicted in the graphical model in Figure 1(b).

We let $\mathbf{y}_o^{(i)}$ denote the set of observed state variables, and let $\mathbf{y}_h^{(i)}$ denote the complement set of state variables that are hidden. Our goal is to maximize the conditional likelihood of the *incomplete data*, *i.e.*, $\log p(\mathbf{y}_o|\mathbf{x})$, which is not directly optimizable. However, we can apply the EM algorithm that iteratively maximizes its low bound. Our learning objective, therefore, is given by

$$J_1 + \sum_{i=m+1}^{n} \mathrm{E}_{p(\mathbf{y}_h^{(i)}|\mathbf{x}^{(i)},\mathbf{y}_o^{(i)};\Lambda^g)} \left[\log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)};\Lambda)\right] \tag{8}$$

The first term is the same as Equation (4). The second term denotes the expected conditional likelihood of auto-labeled data. This is akin to the optimization objective of hidden-state CRFs [14]. In the E-step, we compute the posterior probability $p(\mathbf{y}_h^{(i)}|\mathbf{x}^{(i)}, \mathbf{y}_o^{(i)}; \Lambda^g)$, for $i = m+1, \ldots, n$, based on the current model $\Lambda^g$. This can be efficiently computed using the Forward-Backward algorithm. In both the forward and backward paths, the values of the observed state variables are committed to their derived labels. In the M-step, we fix the posteriors and update $\Lambda$ that maximizes Equation (8). This step can be solved using stochastic gradient descent. The gradient has a similar form as that of $J_1$ except for an additional marginalization over $\mathbf{y}_h$.

There are a number of implementation issues worth attention. First, our semi-supervised learning objective is no longer convex due to the existence of hidden variable. The initialization of model parameters, therefore, becomes critical to learning performance. Here we initialize the model by performing supervised learning on manually-labeled data, but we extract emission features from both data sources. Secondly, since queries are typically short, computation is in general not an impediment to exact inference. As for the stopping criterion, we empirically found that running 2-3 epochs of the EM algorithm gives reasonably good results.

## 4.2 Derived Labels as Soft Evidence

By taking derived labels as hard evidence, we are facing the risk that some state variables may take on erroneous values. The second solution we explore in this work is to use derived label information as *soft evidence*. For queries in the second data source, we view the *entire* state sequence as hidden variables, but use derived labels to provide extra evidence in inference. This is achieved by creating a sequence of soft evidence nodes $z_t$, $t = 1, 2, \ldots, T$, in parallel to hidden state nodes $y_t$. This training setting is encoded in the graphical model in Figure 1(c).

Since all state variables are hidden, the learning objective in Equation (8) is no longer applicable. Instead, we propose to optimize $\log p(\mathbf{z}|\mathbf{x})$. In this case, we can apply the EM algorithm in the same fashion as in Section 4.1 which iteratively optimizes an expected conditional likelihood:

$$J_1 + \sum_{i=m+1}^{n} \mathrm{E}_{p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)},\mathbf{z}^{(i)};\Lambda^g)} \left[\log p(\mathbf{y}^{(i)}, \mathbf{z}^{(i)}|\mathbf{x}^{(i)};\Lambda)\right] \tag{9}$$

where the conditional probability in the second term is defined as

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}; \Lambda) = \frac{1}{Z'(\mathbf{x};\Lambda)} \exp\left\{\sum_k \lambda_k \sum_{t=1}^{T+1} f_k(y_{t-1}, y_t, \mathbf{x}) + \omega \sum_{t=1}^{T} s(y_t, z_t)\right\} \tag{10}$$

Here $Z'(\mathbf{x}; \Lambda)$ is a normalization function (obtained by summing the numerator over both $\mathbf{y}$ and $\mathbf{z}$); $s(y_t, z_t)$ is a *soft evidence feature* with a pre-defined weight $\omega$. The information of derived labels is thus incorporated in the model via this feature function. To use $z_t$ as a “prior” of $y_t$, we choose the following function form,

$$s(y_t, z_t) = \begin{cases} 0 & \text{if } z_t = null \\ 1 & \text{else if } y_t = z_t \\ -1 & \text{else} \end{cases} \tag{11}$$

To understand the impact of the soft evidence feature on $p(\mathbf{y}|\mathbf{x}, \mathbf{z}; \Lambda^g)$ and hence on training, we re-write this posterior probability using the Bayes rule. It is easy to see that $p(\mathbf{y}|\mathbf{x}, \mathbf{z}; \Lambda^g)$ has the same exponential form as $p(\mathbf{y}, \mathbf{z}|\mathbf{x}; \Lambda^g)$ in Equation (10) except that it has a different normalization function. This means that (a) if $x_t$ does not have an derived label, *i.e.*, $z_t = null$, the soft evidence function assigns equal values (zero) to all state hypotheses and the posterior probability solely depends on transition and emission features $f_k$; (b) when there does exist an derived label, the function assigns a relatively large value to the state hypothesis that agrees with the derived label. In other words, the soft evidence function regularizes a hidden state towards the value of the corresponding derived label. The larger $\omega$ is, the more influence this feature has on the posterior probability. At one extreme where $\omega = 0$, the derived label information is completely ignored in training. At the other extreme where $\omega \to +\infty$, all state hypotheses would have extremely small posterior probabilities except the one consistent with the derived label, which is equivalent to using derived labels as hard evidence.

As one last remark, the soft evidence feature is only used in training. Once trained, the CRF model in Equation (1) is used to predict the state sequences for unseen queries.

# 5. OBTAINING DERIVED LABELS

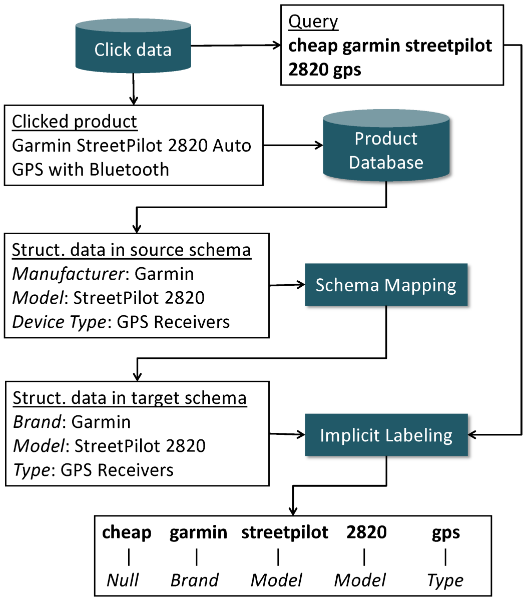
So far we have assumed the availability of derived labels in our proposed semi-supervised learning approaches. In practice, however, the feasibility of acquiring such data in a large amount is critical to the success of our approach. In this work, we give one practical example that automatically obtains such information for product search queries, while our method can be applied to many other domains.

First of all, we define the fields to be extracted from product search queries as in Table 1, and call such a set of fields a *target schema*. Note that there may well be other ways of defining such a schema. The first four fields, *i.e. Brand*, *Model*, *Type*, and *Attribute*, correspond to product metadata we can find from a relational database. The other fields, on the other hand, represent words that frequently occur in product search queries but are not directly related to product metadata. For example, *Mechant* represents a physical or online store that sells products; *SortOrder* represents a way of sorting product listings; *BuyingIntent* and *ResearchIntent* correspond to intents to buy and research products respectively; and *Other* covers everything else. Since the second set of fields do not exist in a product database, they can only be obtained via manual labeing.

Given the target schema, our goal is to obtain derived labels for some word tokens in queries, which are to be used in our proposed semi-supervised CRFs. Briefly speaking, our method leverages product search click-through data in conjunction with a product database. Intuitively, the click data links queries to relevant products, which are then linked to the corresponding metadata via a relational database. With the association between queries and product metadata, it is much easier to predict the field information for some word tokens using simple heuristics. Figure 2 gives a high-level diagram of our approach, while the rest of this section describes each step in detail.

## 5.1 Click data

The click data is extracted from the query log of *Live Search*. When a user clicks on a document after issuing a query, a click event (*query*, *document*) will be recorded in the query log. In particular, we extract click events where the document is known to be a product listing page. To this end, we mine the URL pattern of the product listing pages of an online shopping website http://shopping.msn.com, and select the click events in which their document URLs match such a pattern. Note that the click data could be drastically increased if more shopping websites were considered.



Click data
Query
cheap garmin streetpilot 2820 gps
Clicked product
Garmin StreetPilot 2820 Auto GPS with Bluetooth
Product Database
Struct. data in source schema
Manufacturer: Garmin
Model: StreetPilot 2820
Device Type: GPS Receivers
Schema Mapping
Struct. data in target schema
Brand: Garmin
Model: StreetPilot 2820
Type: GPS Receivers
Implicit Labeling
cheap garmin streetpilot 2820 gps
Null Brand Model Model Type

**Figure 2: Diagram of obtaining derived labels for product search queries**

We further extract product titles from the selected product listing pages. This can be easily achieved by extracting the corresponding field in these pages, which are typically well structured. In this way, a click event is represented in the form of (*query*, *product title*). Alternatively, such pairs can be directly obtained if the query log of a product search engine is readily available.

## 5.2 Product metadata

At the second stage, we leverage a relational database that contains structured information of product listings, including product titles and other metadata such as *Manufacturer*, *Color* and *Weight*. There are over a few hundred metadata-related fields in the product database that we have, and this set of fields will be referred to as a *source schema*. With both the click data and the product database, we are able to associate user queries with their relevant metadata via fuzzy match of product titles. Specifically, given a click event (*query*, *product title*), we look for the database entry that has the most similar product title, and attach the corresponding metadata to the query if the similarity is above certain threshold. In this work, we use *tf-idf* based cosine similarity and use an empirically-chosen threshold 0.75 to select (*query*, *metadata*) pairs. The reason we use fuzzy match instead of exact match is to increase the coverage of user queries for which we can obtain metadata.

## 5.3 Schema mapping

Next, we convert the metadata represented in the source schema to that in the target schema defined in Table 1. In fact, Table 1 was created as a simplified version of the source schema in the first place. For example, *Color*, *Weight* and *Dimension* in the database all correspond to *Attribute* in the target schema. As a first attempt to extract structured information from queries, we use the simplified schema (*i.e.*, target schema) in order to reduce human annotation effort.

The mapping is deterministic. Thus it is straightforward to convert the metadata form as shown in Figure 2.

### 5.4 Auto labeling

Given (*query*, *metadata*) pairs where the metadata corresponds to field information represented in the target schema. We apply the following heuristics to generate derived labels for some word tokens in the queries:

- If a word token does not appear in any field or it appears in more than one field of the metadata, it will be labeled as *null*.
- If a word token appears in exactly one field of the metadata, *e.g.*, "streetpilot" in *Model*, that word token will be labeled with the corresponding field name.

Note that none of the fields in the second row of Table 1 exist in our product database. Consequently, the word tokens of these fields are bound to have *null* labels, *e.g.*, the word "cheap" in Figure 2. In this regard, a pure unsupervised approach is inadequate to label such fields correctly.

## 6. EVALUATION

We evaluate our approach on the task of tagging product search queries based on Table 1, where two evaluation metrics are used. (1) *Sentence accuracy* is the percentage of "sentences" (queries in our case) that are correctly tagged, meaning that the entire decoded state sequence has to be correct. (2) *Word accuracy* is the percentage of word tokens that are correctly tagged. Note that there is another widely-used evaluation metric, *entity accuracy*, which was found following a similar trend as the first two metrics and thus was not reported in this work.

### 6.1 Data

Our experiment data was collected from a 3-month query log of www.live.com. We selected queries that clicked on product listing pages of shopping.msn.com. The selected queries (presumably with product search intent) were further classified into 20 product categories. This was done by an automatic query classifier that was built based on [11]. Our evaluation focuses on the largest two categories, *i.e.*, *clothing-shoes* and *computing-electronics*.

Recall that two data sources are needed in our framework: (1) manually-labeled queries and (2) queries with derived labels. To obtain the first source of data as well as to collect test data for evaluation, we randomly sampled product search queries in each category and asked human annotators to label them. Specifically, a user interface was created where each query was presented along with search results from two major search engines. Then a human annotator browsed through both results before assigning the 9 labels in Table 1 to word tokens. We collected 4K queries as supervised training data for *clothing-shoes*, and another 900 queries (2.5K tokens) as test data. For *computing-electronics*, the training and test data were 15K and 700 (2K tokens) respectively. We labeled a drastically larger number of training-set queries for the second category, only to study the impact of the amount of such data on semi-supervised learning performance. Due to resource limit, each query was labeled only once. However, we had 400 and 1300 queries from these two categories labeled by a secondary annotator. The inter-rater agreement is around 80% at query level and 91% at token level for both categories.

Next, we followed our procedure in Section 5 to collect the second source of data. On one hand, we extracted 500K and 250K (*query*, *product title*) pairs for *clothing-shoes* and *computing-electronics* respectively based on Section 5.1. On the other hand, we had a relational database that contains 20M (*product title*, metadata) pairs across different categories. Applying the fuzzy match of product titles and following the auto-labeling heuristics, we obtained 50K and 20K queries with derived labels (at least one word token must be labeled) for these two categories respectively.

### 6.2 Results

We used three types of emission features that have been defined in Section 3. In addition to ngram features that can be automatically extracted from queries, 7 regex features were defined to represent variations of digit-letter combinations. Furthermore, 4 lexicons were extracted from the product database, corresponding to *Brand*, *Model*, *Type* and *Attribute* respectively. For example, for all fields in the source schema that were mapped to *Attribute* in the target schema, their values were aggregated to be the *Attribute* lexicon.

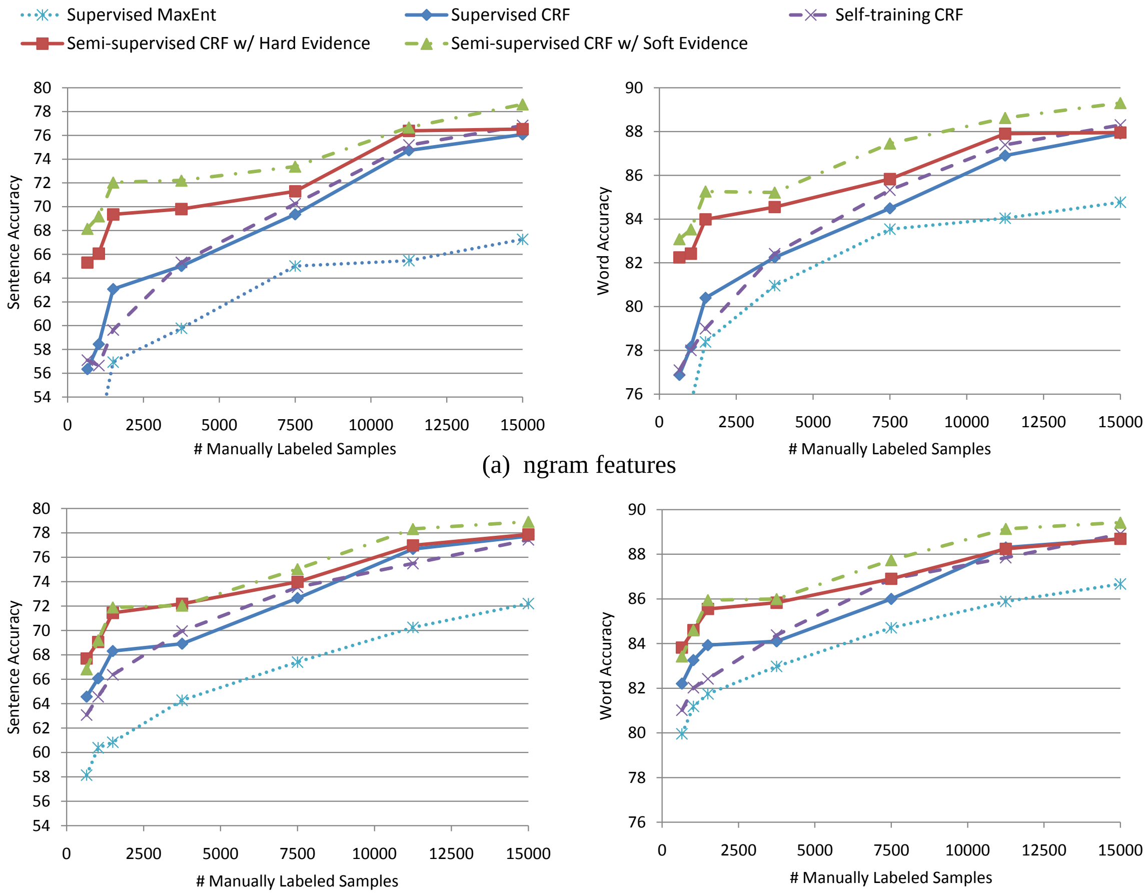
In our evaluation, we use two different feature settings for **all** experiments: (a) ngram features only, including both unigram and bigram features, and (b) ngram + regex + lexicon features. The former setting completely relies on training data, requiring no feature engineering efforts from human. The latter setting has a better generalization ability, but the design of regex features requires some degree of human intervention.

Under the above feature settings, we compare the following methods for query tagging:

- *Supervised MaxEnt.* Use an instance-based maximum entropy model which classifies each word token independently. The model is estimated solely on manually-labeled samples. Note that a *prior feature* is added in this case as a standard MaxEnt feature.
- *Supervised CRF.* Train a linear-chain CRF on manually-labeled samples.
- *Self-training CRF.* Apply self-training [18] on unlabeled queries without using any derived label information. Here the unlabeled queries correspond to the second data source with derived labels removed.
- *Semi-supervised CRF with hard evidence.* Apply the method proposed in Section 4.1.
- *Semi-supervised CRF with soft evidence.* Apply the method proposed in Section 4.2 with $\omega = 1$ (chosen by cross-validation) in Equation (10) for all experiments.

In all models above, the regularization parameter $\sigma^2$ is chosen via cross-validation.

We first inspect tagging performance when we vary the amount of supervised data while maintaining the same set of auto-labeled data. We only experimented with *computing-electronics* since only for this category we obtained a large number of manually-labeled queries. Figure 3 shows the sentence/word accuracies when we gradually increase the number of manually-labeled queries from 500 to 15K and when we use all 20K queries with derived labels. While the



(a) ngram features

(b) ngram + regex + lexicon features

**Figure 3: Tagging accuracies (left: sentence acc.; right: word acc.) for *computing-electronics* with different amounts of manually-labeled queries and a fixed amount of (20K) queries with derived labels.**

first two supervised approaches (MaxEnt and CRF) only use the first data source, the three semi-supervised learning methods additionally use the second data source.

There are several observations from Figure 3: (1) Adding regex and lexicon features works better than using ngram features alone (by comparing the top two plots with the bottom two), especially with a relatively small amount of manually-labeled samples. The difference becomes trivial, though, when the supervised data is significantly increased. This makes sense since regex and lexicon features generalize better, and thus are especially beneficial when supervised data is limited. (2) *Supervised CRF* performs significantly better than *supervised MaxEnt* in both feature settings. This confirms that transition features are helpful in sequential labeling of queries. (3) Semi-supervised CRFs with derived label information greatly improve over supervised training and self-training. The improvement is relatively large when supervised data is limited. Finally, (4) treating derived labels as soft evidence is in general superior to treating them as hard evidence. This is mainly because derived label information is noisy in our task.

In a second set of experiments, we investigate the performance of our best approach, *i.e. semi-supervised CRF with soft evidence*, when fixing the amount of supervised data and exponentially increasing the second data source. We conducted this experiment for both *clothing-shoes* and *computing-electronics*. We use all 4K manually-labeled queries for the former category, and 3K queries for the latter category just to be comparable. As shown in Table 2, the accuracy generally improves when more queries with derived labels are used. The gain is especially large for *clothing-shoes*. This is largely because word ambiguity is more severe a problem in this category (which also accounts for its relatively poor baseline performance). Adding auto-labeled training samples, therefore, is more likely to help.

## 7. CONCLUSIONS AND FUTURE WORK

This work presented semi-supervised CRFs that incorporate derived label information from additional resources. Two principled approaches were explored to encode derived labels into CRFs, one as hard evidence and the other as soft evidence expressed by a feature function. Their respective optimization objectives were discussed. Furthermore, we presented a practical method on how to obtain derived labels by leveraging user click data and an in-domain database in the context of tagging product search queries. Evaluation shows that our semi-supervised learning CRFs with derived labels are effective in improving tagging performance compared with both supervised training and self-training. In addition, we found that treating derived labels as soft ev-

| # derived | ngram features | | ngram+regex+lex | |
|---|---|---|---|---|
| queries | Sent. | Word | Sent. | Word |
| 0 | 37.12 | 65.90 | 45.04 | 68.19 |
| 50 | 38.46 | 63.61 | 46.27 | 69.23 |
| 500 | 41.69 | 66.90 | 50.61 | 73.44 |
| 5K | 44.93 | 69.15 | 51.73 | 73.94 |
| 50K | 46.93 | 71.19 | 51.95 | 74.10 |

(a) *Clothing-Shoes* with 4K manual samples

| # derived | ngram features | | ngram+regex+lex | |
|---|---|---|---|---|
| queries | Sent. | Word | Sent. | Word |
| 0 | 65.02 | 82.25 | 68.31 | 84.10 |
| 20 | 65.92 | 81.63 | 68.01 | 84.87 |
| 200 | 68.31 | 83.03 | 69.96 | 85.22 |
| 2K | 69.06 | 84.04 | 72.20 | 86.28 |
| 20K | 71.21 | 84.66 | 72.05 | 86.00 |

(b) *Computing-Electronics* with 3K manual samples

**Table 2: Tagging accuracies of semi-supervised CRFs with soft evidence, where we use the same sets of manually-labeled samples and vary those with derived labels. The first row corresponds to supervised learning results.**

idence performed the best in our task. In the future, we would like to apply the soft evidence approach to manually-labeled data as well, since this data source can also be noisy. It is also important to show the benefit of query tagging to information retrieval.

## 8. REFERENCES

[1] S. Abney. Understanding the Yarowsky algorithm. *Association for Computational Linguistics*, 30(3):365–395, 2004.

[2] A. Arasu and H. Garcia-Molina. Extracting structured data from webpages. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[3] C. Barr, R. Jones, and M. Regelson. The linguistic structure of English web-search queries. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1021–1030, 2008.

[4] J. Bilmes. On soft evidence in Bayesian networks. Technical Report UWEETR-2004-0016, University of Washington, 2004.

[5] S. Canisius and C. Sporleder. Bootstrapping information extraction from field books. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing*, pages 827–836, 2007.

[6] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 595–602, 2008.

[7] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, 2004.

[8] T. Grenager, D. Klein, and C. Manning. Unsupervised learning of field segmentation models for information extraction. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 371–378, 2005.

[9] F. Jiao, S. Wang, C.-H. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 209–216, 2006.

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

[11] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graph. In *SIGIR'08: Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, July 2008.

[12] G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proceedings of Association of Computational Linguistics*, 2008.

[13] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, 2003.

[14] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transaction on Pattern Analysis and Machine Intellegence*, 29(10):1848–1852, 2007.

[15] J. Suzuki and H. Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of the 46th Annual Meeting of the ACL: Human Language Technologies*, pages 665–673, 2008.

[16] P. Viola and M. Narasimhand. Learning to extract information from semi-structured text using a discriminative context free grammar. In *SIGIR'05: Proceedings of the 28th Annual International ACM SIGIR conference on Research and development in information retrieval*, pages 330–337, 2005.

[17] T.-L. Wong, W. Lam, and T.-S. Wong. An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 35–42, 2008.

[18] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[19] C. Zhao, J. Mahmud, and I. Ramakrishnan. Exploiting structured reference data for unsupervised text segmentation with conditional random fields. In *Proceedings of the SIAM International Conference on Data Mining*, 2008.

[20] J. Zhu, B. Zhang, Z. Nie, J.-R. Wen, and H.-W. Hon. Webpage understanding: an integrated approach. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 903–912, 2007.