Improving Context-Aware Query Classification via Adaptive Self-training

Minmin Chen[1]*, Jian-Tao Sun[2], Xiaochuan Ni[2], Yixin Chen[1]
[1]Department of Computer Science and Engineering
Washington University in Saint Louis, Saint Louis, MO, USA
[2]Microsoft Research Asia, Beijing, P.R. China
[1]{mc15, chen}@cse.wustl.edu, [2]{jtsun, xini}@microsoft.com

ABSTRACT

Topical classification of user queries is critical for general-purpose web search systems. It is also a challenging task, due to the sparsity of query terms and the lack of labeled queries. On the other hand, search contexts embedded in query sessions and unlabeled queries free on the web have not been fully utilized in most query classification systems. In this work, we leverage these information to improve query classification accuracy.

We first incorporate search contexts into our framework using a Conditional Random Field (CRF) model. Discriminative training of CRFs is favored over the traditional maximum likelihood training because of its robustness to noise. We then adapt self-training with our model to exploit the information in unlabeled queries. By investigating different confidence measurements and model selection strategies, we effectively avoid the error-reinforcing nature of self-training. In extensive experiments on real search logs, we have averaged around 20% improvement in classification accuracy over other state-of-the-art baselines.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous; I.5.2 [**Pattern Recognition**]: Design Methodology – Classifier design and evaluation

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Query classification, User search context, Unlabeled queries

1. INTRODUCTION

Recent years have witnessed the booming of web search and its related services. Commercial web search engines receive hundreds of millions of queries every day. Understanding user intents behind queries is essential in developing search engines. Query classification is a task studied for this purpose. In query classification, user queries are classified into one (or more) predefined target categories. Such category information later can be used for refining web page rankings, triggering the most appropriate vertical searches, and finding matched online advertisements, etc..

Challenges associated with query classification include sparseness of both query features and training data. First, user queries usually contain only several unique terms, thus it is difficult to derive a good feature representation. Previous works focused primarily on tackling this feature sparseness problem, e.g., by augmenting queries with external knowledge such as search engine results. Second, despite colossal volumes of unlabeled queries stored in search engine logs, manually labeled queries are very limited and expensive to obtain. Several works [1, 19] introduced semi-supervised learning methods to explore the information hidden in logs. However, most of the systems take a single query as a unit of search engine interaction, without considering search contexts.

It has been argued forcefully that exploiting search contexts can improve information retrieval systems [10, 13, 26]. Intuitively, introducing search contexts, such as the neighboring queries in the same query session, can help better understand users' search intents and thus improve classification accuracy. For instance, when a user issues a query "giant", it is unclear whether he or she is interested in the San Francisco National League baseball team, the bicycle manufacturer, or something else. However, if we knew that the query "MLB" (Major League Baseball) was issued in the same query session, we could have classified the query as "Sports" rather than "Products". Conversely, if the user issued queries related to other bicycle brands before the query "giant", it would have suggested the query was related to "Products".

In this paper, we utilize search contexts embedded in query sessions, together with the large number of unlabeled queries available free on the web, to improve query classification. To achieve this goal, we need to address the following questions: 1) How to model search contexts? 2) How to incorporate the information in unlabeled queries? 3) How to integrate these two components? 4) How to scale our algorithm to handle web-scale data?

First, we model search contexts using a Conditional Ran-

*This work was done when Minmin Chen was an intern at Microsoft Research Asia.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

Table 1: Two user query sessions with sequences of queries, clicked url titles and user dwell time in seconds ("-" indicates that there is no url clicks associated with the query).

SID	observations				Category
	QID	Query	Url title	Time	
1	1	tires 75067	Lewisville tires | Find tires in Lewisville, TX	18	Compare products or services
	2	Traders Village Grand prairie	Traders Village / A Texas-Size Marketplace	4	Compare products or services
	3	craiger wheels	CRAIGER WHEELS	17	Compare products or services
	4	craiger wheels 75067	-	12	Find contact information
	5	Bing	Bing™ Official Site	1	Find a specific miscellaneous fact
	6	wheels 75067	Listings for Wheels near Texas 75067	14	Find contact information
	7	Colleyville tx	Colleyville tx - Google Maps	53	Plan travel
2	1	discount tire lake jackson tx	-	446	
	2	michelin	-	43	
	2	michelin tires Angleton tx	-	20	

dom Field (CRF) model. Second, we explore the information in unlabeled query sessions by combining them with a few manually labeled ones using self-training (bootstrapping). Third, to integrate these two components, we investigate: 1) different training methods for CRFs; Discriminative training is favored as it is more robust to labeling noise introduced by self-training. 2) different confidence measurements and model selection strategies for self-training to help the CRF model converge faster. We refer to the resulting framework as *Adaptive Self-training with Conditional Random Field (ASCRF)*. Forth, we adopt a softmax approximation during the discriminative training of our CRF model, resulting in a formulation which can be efficiently computed and optimized by a slight variant of existing algorithms. As such, our solution can apply to web-scale data.

The rest of this paper is organized as follows. In Section 2, we introduce notations and problem settings. In Section 3, we briefly introduce the CRF model. We compare two commonly used CRF training methods, and apply an approximation to tailor the training algorithm to deal with web-scale data. In Section 4, we detail the ASCRF framework. A series of experiments are carried out to demonstrate the effectiveness of our proposed framework in Section 5. In Section 6, we review related works. Finally, we conclude our work in Section 7.

2. NOTATIONS AND SETTINGS

Search contexts are usually embodied as interactions between queries and related user behaviors within a query session. Various works [7, 24, 17, 14] proposed different methods for identifying the boundaries of query sessions. In this work, we follow a simple yet effective session segmentation rule; that is, two user queries are separated into different sessions if they were issued longer than 30 minutes apart [11, 6].

Table 1 shows two query sessions extracted from real log, one labeled and the other one not. They both consist of a sequence of queries and related user behaviors. Due to space limitations, the urls clicked are not listed. We can make two observations: 1) search contexts can help clarify ambiguous queries; The query "michelin" in Session 2 can either refer to the tire brand, or the Michelin guide book, or something else. However, considering the search context within the session can help correctly classify the query; 2) unlabeled query sessions can help relief the demand for a large number of labeled data. Intuitively, we can infer the category label of the first query in Session 2 based on its similarity to the queries in Session 1, and obtain the labels for the other queries based on context information. By doing so, we can expand the small labeled set with unlabeled queries to improve performance.

Motivated by these observations, we propose to build a semi-supervised context-aware query classification framework that takes advantage of both search contexts and unlabeled query sessions to improve query classification.

Let $\mathbf{x} = \langle \mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_T \rangle$ denote a query session, where each observation $\mathrm{x}_t, t = 1, \cdots, T$ includes a query q_t issued by the user, and sometimes related user behaviors, such as clicked urls, url titles and user dwell time. For any query $q_t (1 \le t \le T)$, the observations $\mathrm{x}_1, \cdots, \mathrm{x}_{t-1}$ are referred to as the search context of q_t [6].

Let $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ be a set of labeled query sessions, where each $\mathbf{x}^{(n)}$ is a query session as defined, and $\mathbf{y}^{(n)}$ is its corresponding labeling sequence. Let $\mathcal{U} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$ be a set of unlabeled query sessions. We have, $N \ll M$. Our goal is to learn a classifier $h(\mathbf{x}; \mathbf{w})$ based on both the labeled and unlabeled query sessions, so that given a new query q_t, it can correctly classify q_t into one of the predefined categories, taking into account the search context of q_t.

3. MODELING SEARCH CONTEXT BY CRFS

Inspired by Cao's work [6], we model search contexts embedded in each query sessions with a linear-chain CRF model.

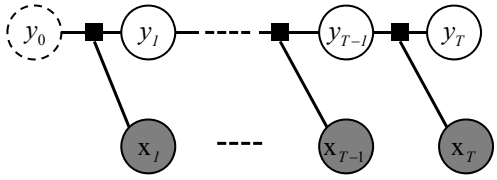


Figure 1: A linear-chain CRF modeling search context.

Let $\mathbf{x} = \langle \mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_T \rangle$ be a user query session, and $\mathbf{y} = \langle y_1, y_2, \cdots, y_T \rangle$ be its corresponding category labeling sequence. As shown in Figure 1, a CRF models the conditional distribution of the labeling sequence $\mathbf{y}$, given the observation sequence $\mathbf{x}$, as [18]

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_k \omega_k f_k(y_{t-1}, y_t, \mathbf{x})\right) = \frac{e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y})}}{Z(\mathbf{x})},$$

where $\mathbf{f} = \{f_k\}$ is a set of predefined *feature* functions, and $Z(\mathbf{x}) = \sum_{\mathbf{y}'} e^{\mathbf{w}^\top \mathbf{f}_\mathbf{x}(\mathbf{y}')}$ is a normalization function.

Given a set of trained parameters $\mathbf{w}^*$, a new query session $\mathbf{x}$ is classified as

$$\hat{\mathbf{y}} = h(\mathbf{x}; \mathbf{w}^*) = \arg\max_{\mathbf{y}'} \mathbf{w}^{*\top} \mathbf{f}_\mathbf{x}(\mathbf{y}'). \quad (1)$$

3.1 Training CRFs

In this work, we investigate two commonly used training methods for CRF models, i.e., maximum likelihood training vs. margin maximization training.

3.1.1 Maximum Likelihood Training.

Given a set of i.i.d training data $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$, maximum likelihood training finds the parameters $\mathbf{w} = \{\omega_k\}$ such that the logarithm of the likelihood,

$$\ell(\mathcal{D}; \mathbf{w}) = \sum_{n=1}^{N} \log p(\mathbf{y}^{(n)} | \mathbf{x}^{(n)})$$

is maximized.

The method has been widely used for CRF training because both of its training and inference can be carried out efficiently using standard dynamic programming algorithms [25], making it scalable to large scale problems. However, as the approach explicitly considers only the probability of the exact training parses, it can easily overfit the training data without proper regularization [22, 9].

3.1.2 Margin Maximization Training.

Margin maximization training of CRF models is a popular alternative. Given a set of training data $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$, it finds a hyperplane that not only separates the training data, but also maximizes the score difference between the true label and the others, i.e. [27, 28],

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_n \quad (2) \\ \text{s.t.} \quad & -\mathbf{w}^\top \mathbf{f}_{\mathbf{x}^{(n)}}(\mathbf{y}^{(n)}) + \max_{\mathbf{y}} \left(\mathbf{w}^\top \mathbf{f}_{\mathbf{x}^{(n)}}(\mathbf{y}) + \Delta_{\mathbf{y}^{(n)}}(\mathbf{y}) \right) \\ & \leq \xi_n, \forall n \end{aligned}$$

where $\Delta_{\mathbf{y}^{(n)}}(\mathbf{y}) = I(\mathbf{y}_t^{(n)} \neq \mathbf{y}_t)$ denotes the hamming distance between $\mathbf{y}^{(n)}$ and $\mathbf{y}$.

To simplify the optimization, we replace the "hard" max margin in (2) with a softmax $\log \sum \exp$ approximation as proposed in [23, 12]. Let

$$\rho_n(\mathbf{w}) = \max \left\{ 0, \begin{array}{l} -\mathbf{w}^\top \mathbf{f}_{\mathbf{x}^{(n)}}(\mathbf{y}^{(n)}) + \\ \log \left(\sum_{\mathbf{y} \neq \mathbf{y}^{(n)}} e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}^{(n)}}(\mathbf{y}) + \Delta_{\mathbf{y}^{(n)}}(\mathbf{y})} \right) \end{array} \right\},$$

we can now eliminate the slack variables in (2) and rewrite our objective function as follows:

$$\textbf{SMMCRF:} \quad \min \ell(\mathcal{D}; \mathbf{w}, C) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_n \rho_n(\mathbf{w}). \quad (3)$$

We refer to the resulting model as SoftMax Margin Conditional Random Field (SMMCRF). The objective is convex, and can be computed efficiently with a slight variation of the standard forward-backward algorithm used in maximum likelihood training [23]. We present an analysis of the generalization performance of SMMCRF as follows. The bound provides a natural way to bridge SMMCRF with semi-supervised learning to utilize unlabeled data to improve query classification.

3.1.3 Generalization Performance of SMMCRF.

Given a set of i.i.d. training data $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\}_N$, Vapnik [29] showed that with probability $1 - \eta$, the expected risk

$$R(\mathbf{w}) \leq R_{emp}(\mathbf{w}) + \Omega(N, d, \eta), \quad (4)$$

where $R_{emp}(\mathbf{w}) = \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \Delta(\mathbf{y}, h_\mathbf{w}(\mathbf{x}))$ is the empirical risk on the training set, and the VC-confidence $\Omega(N, d, \eta)$ is a term depending on the number of training examples N and the VC-dimension d of the hypothesis class H.

PROPOSITION 1. *Let $\mathbf{w}^*$ be the optimal solution to (3). Then $\frac{1}{|\mathcal{D}|}\sum_n \rho_n(\mathbf{w}^*)$ upper bounds the empirical risk.*

The proof can be easily generalized from Tsochantaridis's proof for the generalization bound of structured SVM. We refer interested readers to [28].

It is known that the VC-dimension of margin-based classifiers grows inversely with the size of the margin [5]. In other words, the margin control factor C in (3) controls the tradeoff between the training error and the complexity of the classifiers. As $C \to \infty$, the empirical risk goes to zero, at the cost of increasing the complexity of the hypothesis class. Contrarily, as C decreases, the size of the margin increases, resulting in a hypothesis class of lower complexity. However, the empirical risk would be increased. In the next section, we will detail how we effectively control C to help avoid the error-reinforcing nature [32] of self-training.

4. INCLUDING UNLABELED QUERIES VIA ADAPTIVE SELF-TRAINING

Now we have a model that incorporates search contexts, in this section, we will be focusing on how to use semi-supervised learning to explore the information contained in unlabeled query sessions to further improve query classification.

4.1 Self-training

The general idea behind semi-supervised learning is to use a large amount of unlabeled data, together with a few labeled data, to build better classifiers. A variety of semi-supervised learning methods have been proposed [30, 4, 2, 16, 20, 15]. However, only a few of them work with graphical models, like CRFs. Self-training [30] is a very commonly used algorithm to wrap complex models for semi-supervised learning. It works as follows:

1. Train a base classifier using the small labeled set;
2. Infer the labels for the unlabeled data;
3. Add confident predictions to the training set;
4. Repeat the process.

Note that the classifier uses its own predictions to teach itself. The procedure is therefore also called self-teaching or bootstrapping.

The advantage of self-training lies in its simplicity and the ability to combine with complex models. On the other hand, the drawbacks of this approach are: 1) erroneous labels are introduced into the training set; and 2) errors are

reinforced at each iteration. In other words, wrong predictions are strengthened during the self-teaching procedure, especially when combined with complex models. Nevertheless, the analysis of the generalization performance of the SMMCRF model in Section 3.1.3 provides heuristics on how to best integrate our model with self-training to avoid error-reinforcing.

4.2 Adaptive Self-training

In this section, we detail the adaptations we made to the self-training algorithm, based on the characteristics of the SMMCRF model.

4.2.1 Overall Framework

Algorithm 1 listed the details of the overall framework. We call our framework ASCRF, *Adaptive Self-training with Conditional Random Field*. We start a SMMCRF model trained on the small set of labeled data. A common practice to set the margin controlling factor C in (3) is through cross-validation. However, since manually labeled data is very limited in this case (in our experiments, only a few hundreds of the query sessions are labeled), it is not practical to do so. Structural risk minimization tells us that we get the smallest bound on the test error if we select a class of hypotheses H that minimizes the right hand side of (4). We start with a relatively large C, so that we can enforce zero empirical risk on the training set. As we gradually decrease C, the margin of our classifier is increased, so that the second term of the bound (4) is decreased. As proved in Proposition 1, $\sum \rho(\mathbf{w})$ upper bounds the empirical risk. We stop when the bound exceeds 0.

We then label query sessions in the unlabeled set $\mathcal{U}$ with the trained base classifier. That is, $\forall \mathbf{x} \in \mathcal{U}$, a prediction $\hat{\mathbf{y}} = h(\mathbf{x}, \mathbf{w})$ is assigned using the trained parameters $\mathbf{w}$. Then, num, the number of most confident predictions $(\mathbf{x}, \hat{\mathbf{y}})$ are picked out and added to the labeled set $\mathcal{D}'$. The model is then retrained with the augmented labeled data set $\mathcal{D} \cup \mathcal{D}'$. The process repeats until we can no longer find confident predictions to aid our learning. To further help our algorithm avoid stuck with wrong predictions in $\mathcal{D}'$, we remove predictions that our model deems unconfident during this process.

To complete our algorithm, we investigated 1) different confidence measurements for selecting unlabeled queries to expand the training set; 2) different model selection strategies (margin control schemes) to balance the capacity of the SMMCRF model to fit the training data, and the complexity of the models.

4.2.2 Confidence measurements

First, margin maximization training offers us a straightforward margin-based confidence measurement:

$$c^{(1)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) = \mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}}) - \log \sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}') + \Delta_{\hat{\mathbf{y}}}(\mathbf{y}')}.$$

It measures the gap between the prediction $\hat{\mathbf{y}}$ to any other sequence $\mathbf{y}'$ in the output space. The larger the gap, the more confident our model is about this prediction. We select the predictions with the maximum $c^{(1)}_{\mathbf{w}}$.

Second, as introduced in section 3, a CRF models the conditional distribution of a labeling sequence $\mathbf{y}$ given the observation $\mathbf{x}$. Given a trained model $\mathbf{w}$, we can also measure the confidence in a prediction by the conditional probability,

$$c^{(2)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) = p(\hat{\mathbf{y}}|\mathbf{x}) = \frac{e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}})}}{\sum_{\mathbf{y}'} e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}')}}.$$

Similarly, the larger the conditional probability, the more our model favors the prediction $\hat{\mathbf{y}}$. We select the prediction with the maximum $c^{(2)}_{\mathbf{w}}$

Third, the decision boundary of margin-based classifiers depends on the set of support vectors, i.e., the set of data points with

$$\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}}) - \log \sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}') + \Delta_{\hat{\mathbf{y}}}(\mathbf{y}')} = 0.$$

Therefore, our third strategy is to select support vectors, i.e., the predictions that are closest to the decision boundary.

$$c^{(3)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) = -c^{(1)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) \cdot I\left(c^{(1)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) \geq 0\right)$$

Notice here, to avoid wrong predictions, only the predictions that are right on the margin or outside of the margin are considered. It is plausible to call $c^{(3)}_{\mathbf{w}}$ a confidence measurement since high $c^{(3)}_{\mathbf{w}}$ does not indicate that our model is confident about the correctness of the prediction. However, to be consistent with the rest of the notations, we still refer to it a confidence measurement.

By definition, the confidence measurement becomes less strict from $c^{(1)}$ to $c^{(3)}$. With a simple transformation, we can rewrite $c^{(1)}$ as

$$\exp\left(c^{(1)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}})\right) = \frac{e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}})}}{\sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}') + \Delta_{\hat{\mathbf{y}}}(\mathbf{y}')}}.$$

In other words, in order to obtain a high confidence $c^{(1)}$, a prediction $\hat{\mathbf{y}}$ has to maintain a dominant score even when the other parses $\mathbf{y}'$ is helped by $\Delta_{\hat{\mathbf{y}}}(y')$. In $c^{(2)}$, the hamming loss $\Delta_{\hat{\mathbf{y}}}(y'))$ is dropped,

$$c^{(2)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) = \frac{e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}})}}{\sum_{\mathbf{y}'} e^{\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}')}},$$

resulting in a looser standard. $c^{(3)}$ further relax the requirement by only constraining the predictions to lie on or outside of the decision boundary. Therefore, we would expect that the predictions selected using confidence measurement $c^{(1)}$ contain the least noise, and the amount of noise increases when using $c^{(2)}$, and $c^{(3)}$ is the worst. However, in terms of the usefulness of the new predictions to the convergence of the self-training process, the order is reversed. In [31], Zhang et al. showed that knowing the label of an unlabeled data point, on which the current model has low confidence, has more potential in improving the classification. Intuitively, adding a prediction $(\mathbf{x}, \hat{\mathbf{y}})$ on which our model has already achieved a large margin over the other parses will not add any new constraints to our formulation in (2), since they are already satisfied by the current set of parameters $\mathbf{w}$. Thus, using confidence measurement $c^{(1)}$ leads to slow convergence. In contrast, the unlabeled points lying right on the decision boundary have great influence on the change of $\mathbf{w}$. During our empirical study, we found that, as a trade-off between $c^{(1)}$ and $c^{(3)}$, the confidence measurement $c^{(2)}$ performs the best. It introduces less noise than $c^{(3)}$, and it helps the classifier achieve faster convergence than $c^{(1)}$.

Algorithm 1: Adaptive Self-training with Conditional Random Fields (ASCRF)

Input:

- Labeled query sessions $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\}^N$;
- Unlabeled query sessions $\mathcal{U} = \{\mathbf{x}\}^M$;
- $N \ll M$.

Parameters:

- β: constant for tuning the penalty factor C in (3);
- num: number of unlabeled query sessions to be labeled at each iteration;
- $c_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}})$: confidence measurement of the prediction $(\mathbf{x}, \hat{\mathbf{y}})$.

Output:

- $\mathbf{w}$ the set of optimal parameters

Initialize C to a relatively large number;
repeat
  $\mathbf{w}$ = Train_SMMCRF($\mathcal{D}, C$);
  $C \leftarrow C/\beta$;
until $\sum_n \rho_n(\mathbf{w}) > 0$;
repeat
  Label query sessions in $\mathcal{U}$ with the trained classifier: $\forall \mathbf{x} \in \mathcal{U} \longrightarrow \hat{\mathbf{y}} = h(\mathbf{x}; \mathbf{w})$;
  Extract num predictions $(\mathbf{x}, \hat{\mathbf{y}})$ of the highest confidence $c_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}})$ from $\mathcal{U}$ and add them to the labeled set $\mathcal{D}'$;
  Train a SoftMax Margin CRF with the data in $\mathcal{D} \cup \mathcal{D}'$: $\mathbf{w}$ = Train_SMMCRF($\mathcal{D} \cup \mathcal{D}', C$);
  Remove unconfident instances: $\forall (\mathbf{x}, \hat{\mathbf{y}}) \in \mathcal{D}'$, remove it from $\mathcal{D}'$ if $\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}}) < \max_{\mathbf{y}'} \mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}') + \Delta_{\hat{\mathbf{y}}}(\mathbf{y}')$;
  Estimate the labeling noise ϵ newly introduced to $\mathcal{D}'$ by $\epsilon \approx \sum_{num}(1 - p(\hat{\mathbf{y}}|\mathbf{x}))/num$;
  Increase the margin of the classifier to take into consideration the noise by decreasing C: $C \leftarrow C/(1+\epsilon)$;
until *no more confident predictions can be found*;

4.2.3 Model selection

At each iteration of the self-training procedure, we have to retrain our model based on the set of augmented training data $\mathcal{D} \cup \mathcal{D}'$. As self-training goes on, labeling noise would be introduced to the training set. In this case, instead of forcing the classifier to come up with a set of parameters $\mathbf{w}$ that perfectly classify the training data, we should increase the margin so that the mistakenly labeled training data can be ignored. As the margin increases, the second term of the generalization bound (4) can be reduced to ensure generalization performance. However, blindly enlarging the margin will result in a model with very high bias, increasing both the empirical and expected risk. The tradeoff here is balanced through the margin control factor C. Often the case, cross-validation is used to tune hyperparameters. However, it is prohibitive to do in this setting, due to the sparseness of training data, as well as the relatively long running time of the entire self-training process.

Instead, we came up with an updating rule for the penalty factor C that depends on an estimated error rate of the training data. With a set of trained parameter $\mathbf{w}$, we can estimate the error rate of the newly selected predictions by

$$\epsilon \approx \frac{\sum_{num}(1 - p_{\mathbf{w}}(\hat{\mathbf{y}}|\mathbf{x}))}{num};$$

where $p_{\mathbf{w}}(\hat{\mathbf{y}}|\mathbf{x})$ computes the conditional probability of the true parse being $\hat{\mathbf{y}}$ given the observation $\mathbf{x}$ under current model.

At each iteration, C is decreased by a factor proportional to the estimated error rate, that is,

$$C \leftarrow C/(1+\epsilon).$$

To further avoid error-reinforcement, we also drop the instances that were added into $\mathcal{D}'$ during the early stage of the self-training process, but later were found out to be unconfident ones. That is, a prediction $(\mathbf{x}, \hat{\mathbf{y}})$ is removed from the training set if

$$\tau_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) = \mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\hat{\mathbf{y}}) - \max_{\mathbf{y}'} \left(\mathbf{w}^\top \mathbf{f}_{\mathbf{x}}(\mathbf{y}') + \Delta_{\hat{\mathbf{y}}}(\mathbf{y}') \right) < 0.$$

This term can be efficiently computed with a slight variant of the standard Viterbi algorithm [21]. Note that our first confidence measurement $c^{(1)}$ upper bounds this term, that is, $\forall \mathbf{w}, (\mathbf{x}, \hat{\mathbf{y}})$, $c^{(1)}_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}}) > \tau_{\mathbf{w}}(\mathbf{x}, \hat{\mathbf{y}})$. The reason we did not use $c^{(1)}$ directly is that when it becomes less than zero, the prediction $(\mathbf{x}, \hat{\mathbf{y}})$ is not necessarily within the margin, which might cause correct predictions being removed.

5. EXPERIMENTAL RESULTS

In this section, we validate our proposed methods through systematic empirical comparisons with previous work on two real datasets.

5.1 Datasets

Both datasets were extracted from real search logs. Queries were segmented into different sessions according to the rule described in section 2.

5.1.1 Dataset 1

The first dataset is from previous work [6]. It contains 3,500 query sessions (8,988 queries) extracted from one day's search log of a commercial search engine. Three labelers were invited to label the queries of each session using

the target taxonomy of ACM KDD Cup'05, which is a two-level taxonomy with seven level-1 categories and 67 level-2 categories.

5.1.2 Dataset 2

To make our story complete, we also apply our algorithm to a relatively larger second dataset. The set was extracted and labeled by a product team in 2010. It contains 1,727 query sessions (39,565 queries) in total. Instead of the ACM KDD Cup'05 taxonomy, the queries were labeled using a set of categories more closely related to users' search intents, such as, "Compare products, services, or activities for use", "Learn how to perform a task", and "Plan travel". There are 65 categories in total.

5.2 Evaluation Metric

In our experiments, we used only previous queries in the same session as the search contexts since related user behavior information was missing in Dataset 1. Different from classic sequential learning tasks in which the testing sequences come in batch, queries come in streaming. Hence, the testing phase for query classification was done differently. Given a sequence of queries $q_1, q_2, \ldots, q_t$, we take the query q_t as the test query, and the queries $q_1, q_2, \ldots, q_{t-1}$ as the context of q_t, and find the prediction of the session as in (1). Then the label of query q_t is set to $\hat{y}_t = \hat{\mathbf{y}}_t$. We do this for each time stamp t.

Given a set of queries q_t with true label y_t, and prediction $\hat{y}_t$, we report the accuracy of our trained classifier as

$$Acc = \frac{1}{|T|} \sum_t I(y_t = \hat{y}_t),$$

where $|T|$ is the total number of queries in the testing set.

5.3 Preliminary Experiments

We used the smaller set, Dataset 1, for preliminary study. We first compared different training methods for our CRF model. We then investigate different confidence measurements and model selection strategies for our adaptive self-training framework. Finally we validate our framework on Dataset 2.

5.3.1 Robustness of SMMCRF

In the first experiment, we compare the performance of softmax margin maximization training (denoted as SMMCRF) with maximum likelihood training (denoted as CRF), and maximum likelihood training with regularization $\|\mathbf{w}\|/2\sigma^2$ [22] (denoted as CRF-R) on Dataset 1.

Experiment Setup. We did 5-fold cross-validation. That is, the entire dataset was randomly split into five folds. Each time four of them were used as training, and the remaining one as testing. The average testing accuracy is reported. In this experiment, all of the training data are labeled, as opposed to the following experiments, in which only fraction of the training data are labeled. For this experiment, the margin control factor C in our loss function (3) and the Gaussian prior σ^2 of the regularized CRF [22] were both tuned on a validation set.

In order to test the robustness of the trained models to noise, we added random labeling noise to our training data. The labeling noise goes from 10% to 40% of the data. To deal with the increasing noise, the margin control factor C of SMMCRF is decreased by a constant factor each time, same for the Gaussian prior σ^2 of CRF-R.

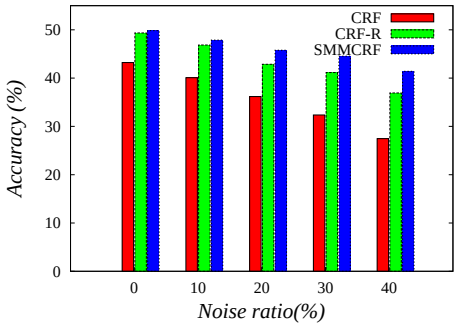


Figure 2: Comparison of softmax margin maximization and maximum likelihood training on dataset 1

Discussion. As shown in Figure 2, the model obtained with maximum likelihood training without regularization is seriously overfitting the training data. At point 0, that is, when the data contains no noise, SMMCRF achieved over 20% improvement over CRF in term of generalization performance. CRF-R did relieve the overfitting problem. However, as more and more labeling noise is added into the dataset, we can clearly see the advantage of margin maximization training. While the performance of CRF and CRF-R deteriorates very quickly as more noise is added, SMMCRF turns out to be much more robust. We can see the performance gap between MMCRF and CRF-R/CRF becomes larger and larger as more noise is added to the training data from Figure 2.

5.3.2 Adaptive Self-training with SMMCRF

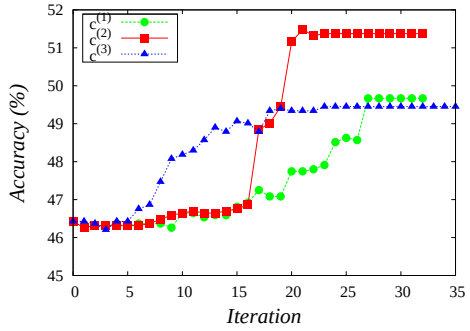
Now that we have a model shown to be robust to labeling noise both theoretically and empirically, in this part, we study the effects of different choices of confidence measurements and model selection strategies on the performance of our framework on Dataset 1.

Experiment Setup. As classical semi-supervised learning setting, we used 80% of the data as training, and the remaining 20% as testing. 10% of the training data are randomly selected to be labeled, and the others are used as unlabeled data. We did 10 runs of these experiments, each time with a different labeled set. Since the size of the training set is small, we used the level-1 categories of the ACM KDD Cup'05 taxonomy, which contains seven categories in total, as the output.

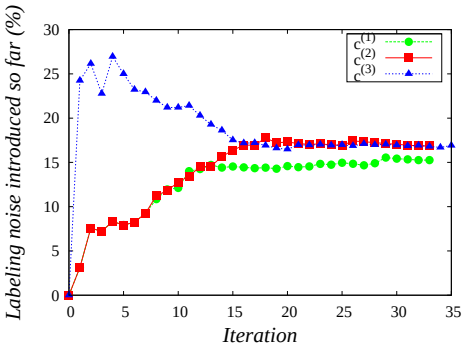
Confidence measurements. In this experiment, we study the effects of various confidence measurements on the performance of our framework. Figure 3(a) shows the progress of ASCRF framework with the three confidence measurements we introduced in Section 4.2.2. Figure 3(b) shows the percentage of noise introduced into our training set when unlabeled query sessions are extracted and added to the set $\mathcal{D}'$ using different confidence measurements.

The results are consistent with our discusion in Section 4.2.2. 1) In terms of noise rate (Figure 3(b)), $c_{\mathbf{w}}^{(1)}$ includes the least amount of noise into the training set, $c_{\mathbf{w}}^{(2)}$ the second, and $c_{\mathbf{w}}^{(3)}$ is the worst. Especially at the very beginning of the process, since our model is trained with a very small labeled set, it is seriously overfitting the training data,

and does not generalize well to unseen data at that point. Therefore, the predictions close to the decision boundary under the current model are very likely to be wrong. 2) In terms of model convergence (Figure 3(a)), the few correct predictions selected by $c_{\mathbf{w}}^{(3)}$ significantly drive up the performance, since they are the data points the original model not sure of. Knowing the labels for these query sessions have great potential in improving the performance. On the other hand, although the predictions selected by $c_{\mathbf{w}}^{(1)}$ are of high accuracy, since they are the data points the current model is very confident about, adding these data points do not help much. 3) In the long run, as the self-training procedure goes on, the wrong predictions selected by $c_{\mathbf{w}}^{(3)}$ start to hurt the learning process; while on the other hand, as the size of the margin increases, the highly accurate predictions selected by $c_{\mathbf{w}}^{(1)}$ start to help the learning process. $c_{\mathbf{w}}^{(2)}$, as a tradeoff between these two, leads our model to the best generalization performance. In the following experiments, we used $c_{\mathbf{w}}^{(2)}$ as our confidence measurement.



(a) Accuracy



(b) Noise

Figure 3: Comparison of different confidence measurements: (a) Accuracy of Adaptive Self-training with different confidence measurements (b) Noise rate in $\mathcal{D}'$ with different confidence measurements.

Model selection. In this experiment, we test different model selection strategies introduced in Section 4.2.3. We proposed to update the margin control factor C in (3) with an estimate of the noise rate in set $\mathcal{D}'$. We compare this strategy with 1) a constant factor C; 2) a constant update, that is, $C \leftarrow C/\beta$ at each iteration of self-training. All experiments were carried out in the same setting, except with different updating rules for C. The confidence measurement $c_{\mathbf{w}}^{(2)}$ is used.

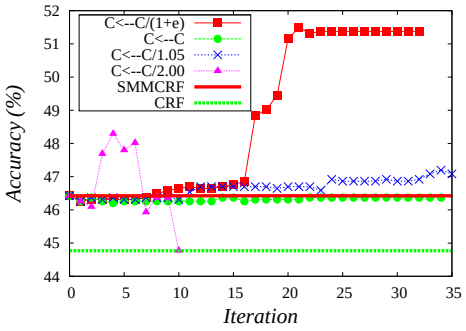


Figure 4: Comparison of different model selection strategies.

For constant update, we tried $\beta = 1.05$ and $\beta = 2$. As shown in Figure 4, self-training with constant C is overwhelmed by the noise introduced, thus shows no progress. For constant update, a small β leads to slow convergence, while a large β results in a model of high bias. As shown in the magenta curve, when $\beta = 2$, the model can not find confident predictions within several iterations, and terminates very quickly. On the other hand, with the proposed updating rule, we achieved over 10% improvement compared to a SMMCRF model trained with only the labeled instances shown as red solid line, and around 20% improvement over a CRF model trained with maximum likelihood training, shown as green dotted line in Figure 4. To make comparison, we also trained a SMMCRF with all the training data labeled, it achieved 58.85% test accuracy (Notice here the number is higher than the one shown in Figure 2 because in this experiment we used only 7 level-1 categories as opposed to 67 level-2 categories in previous one), while our framework achieved 51.38% precesion with only 10% of the training data labeled.

5.4 Overall performance

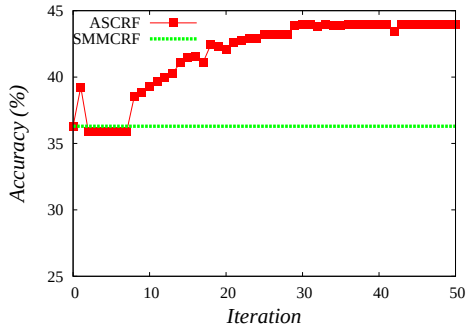
In this experiment, we test our overall framework on Dataset 2. The experiment setting is same as previous, 80% of the data are used as training set, and the remaining 20% as testing. We only used 5% of the training data as labeled ones, and the remaining 95% are unlabeled. Same here, we did 10 runs of this experiments.

Figure 5 shows the progress of our framework on Dataset 2, with proposed model selection strategy and confidence measurement $c_{\mathbf{w}}^{(2)}$. We compare our ASCRF framework to a baseline, which trains a SMMCRF with the labeled query sessions. As shown in the figure, in Run 1, we improved over the baseline for around 20% in accuracy; and in Run 2, we nearly doubled the performance.

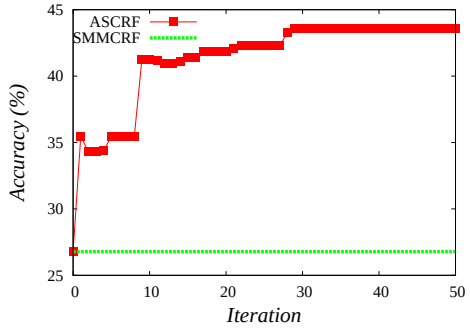
Table 2: Overall performance comparison (Accuracy and Standard Deviation) of ASCRF vs. baseline on Dataset 1 and Dataset 2, with 10 runs each.

Accuracy(Std)	Baseline	ASCRF
Dataset 1	$44.67\%(1.48 \times 10^{0})$	$49.94\%(8.07 \times 10^{-1})$
Dataset 2	$35.25\%(4.95 \times 10^{-2})$	$43.53\%(4.44 \times 10^{-3})$

Table 2 reports the overall experimental results. The results are consistent across the two datasets. We have averaged around 10% improvement on Dataset 1, and over 20%



(a) Run 1



(b) Run 2

Figure 5: Two runs of ASCRF on Dataset 2: 5(a) Run 1; 5(b) Run 2.

on Dataset 2. Our algorithm achieved more significant improvement and smaller deviation on Dataset 2. We believe this is due to the size of the unlabeled set is larger in Dataset 2 than Dataset 1.

5.5 Efficiency

The main computational time of our algorithm comes from the training of SMMCRF. As stated in Section 3.1.2, using the softmax approximation, we can compute the objective and gradient for our formulation efficiently with a slight variant of the forward-backward algorithm used in the standard maximum likelihood training. Hence, the computational time is linear on the sequence length, and quadratic on the number of categories. Our algorithm was implemented on top of the Limited Memory Variable Metric (LMVM) solver in the Toolkit for Advanced Optimization (TAO) [3]. The experiments were run on a machine with Intel(R) Xeon(R) CPU 2.67GHz, and 4G memory. It takes us around 2 minutes to finish the entire adaptive self-training process for Dataset 1, with 7 level-1 categories, and 8,988 queries; and around 4 hours on Dataset 2, with 65 categories, and 39,565 queries. The time cost is acceptable for offline training. Also, the algorithm can run parallel to improve efficiency.

Web users often have very strict requirement on the response time of online applications. Fortunately, the testing phase of our algorithm can be done very efficiently using the standard Viterbi algorithm. During our experiment, it takes less than 1 second to finish the testing phase for Dataset 2, which contains around 7,000 test queries in total.

6. RELATED WORK

Cao et al.'s work [6] adopted CRF models to include context information in query classification. There are two major differences between our work and theirs: 1) The previous work was carried out under a supervised learning setting. In order to achieve satisfactory performance, a great number of labeling queries are required. In contrast, we propose to combine a small set of labeled instances with the abundant unlabeled queries free on the web to build a good query classifier, reducing human effort in labeling query sessions. 2) The previous work used the traditional maximum likelihood training for CRF models. However, during our study, we found that discriminative training of CRF models is much more robust to labeling noise. Empirical study also supports our claim, SMMCRFs consistently outperform CRFs trained with maximum likelihood training. It provides us not only with guaranteed generalization performance, but also a straightforward mechanism to adapt the popular semi-supervised learning method, self-training, to explore the information contained in unlabeled queries.

The second class of related work is semi-supervised learning [32, 8], which aims at utilizing the information in unlabeled data to improve supervised learning. This can be achieved in various ways [30, 4, 2, 16, 20, 15]. In 2005, Beitzel et al. [1] proposed a semi-supervised learning framework tailored specially for query classification. It uses labeled data to learn logical relationships between query terms and query categories, and unlabeled data to fully exploit the "syntactic" dependencies between query terms, so that more queries can be classified. Li et al. [19] used click graphs to increase the amount of training data. Specifically, class memberships of unlabeled queries are inferred from those of labeled ones according to their proximities in a click graph. However, both works ignored the context information contained in query sessions.

7. CONCLUSION AND FUTURE WORK

Accurate topical classification of user queries can benefit a variety of web applications. A great amount of works have been proposed to study the problem. However, the issue of lacking training data to obtain a high quality classifier has not been well addressed. Inspired by a previous work on context-aware query classification, we built a general framework that exploits the large number of unlabeled queries available free on the web, as well as search contexts embedded in query sessions to improve query classification. Extensive experiments on two datasets extracted from real search log demonstrated the efficacy of our framework.

In our experiments, we used only previous queries in the same session as search contexts. In future work, we plan to include related user behaviors, such as clicked urls, user dwell time, to enrich our feature set and investigate the impact of them. Li et al. [19] found that with a large amount of training data, classifiers using only query words/phrases as features can work remarkably well. It would be interesting to see if the statement still holds in our case.

8. REFERENCES

[1] S. Beitzel, E. Jensen, O. Frieder, D. Lewis, A. Chowdhury, and A. Kołcz. Improving automatic query classification via semi-supervised learning. In *Proc. ICDM*, pages 42–49, 2005.

[2] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. *Learning theory*, pages 624–638, 2004.

[3] S. Benson, L. McInnes, J. Moré, and J. Sarich. TAO user manual (revision 1.9). *Mathematics and Computer Science Division, Argonne National Laboratory, Tech. Rep. ANL/MCS-TM-242*, 2005.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. COLT*, pages 92–100, 1998.

[5] C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[6] H. Cao, D. Hu, D. Shen, D. Jiang, J. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *Proc. SIGIR*, pages 3–10, 2009.

[7] L. Catledge and J. Pitkow. Characterizing browsing strategies in the World-Wide Web. *Computer Networks and ISDN systems*, 27(6):1065–1073, 1995.

[8] O. Chapelle, B. Schölkopf, A. Zien, et al. *Semi-supervised learning*. MIT press Cambridge, MA, 2006.

[9] M. Chen, C. Y., M. Brent, and A. Tenney. Gradient-Based Feature Selection for Conditional Random Fields and Its Applications in Computational Genetics. In *Proc. ICTAI*, pages 750–757, 2009.

[10] B. Croft et al. The role of context and adaptation in user interfaces. *Journal of Man-Machine Studies*, 21(4):283–292, 1984.

[11] H. Cui, J. Wen, J. Nie, and W. Ma. Probabilistic query expansion using query logs. In *Proc. WWW*, pages 325–332, 2002.

[12] K. Gimpel and N. Smith. Softmax-margin crfs: Training log-linear models with cost functions. In *Proc. ACL*, pages 733–736, 2010.

[13] A. Goker. Context learning in Okapi. *Journal of Documentation*, 53(1):80–83, 1997.

[14] B. Jansen, A. Spink, C. Blakely, and S. Koshman. Defining a session on web search engines. *Journal of the American Society for Information Science and Technology*, 58(6):862–871, 2007.

[15] F. Jiao, S. Wang, C. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proc. ACL*, pages 209–216, 2006.

[16] T. Joachims. Learning to classify text using support vector machines: Methods, theory, and algorithms. *Computational Linguistics*, 29(4):656–664, 2002.

[17] R. Jones and K. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proc. CIKM*, pages 699–708, 2008.

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

[19] X. Li, Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *Proc. SIGIR*, pages 339–346, 2008.

[20] G. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. ICML*, pages 593–600. ACM, 2007.

[21] N. Seshadri and C. Sundberg. List Viterbi decoding algorithms with applications. *Communications, IEEE Transactions on*, 42(234):313–323, 2002.

[22] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. Human Language Technology - NAACL*, pages 134–141, 2003.

[23] F. Sha and L. Saul. Large margin hidden Markov models for automatic speech recognition. In *Proc. NIPS*, pages 1249–1256, 2007.

[24] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *ACM SIGIR Forum*, volume 33, pages 6–12, 1999.

[25] C. Sutton and A. McCallum. An Introduction to Conditional Random Fields for Relational Learning. *Introduction to statistical relational learning*, page 93, 2007.

[26] S. Talja, H. Keso, and T. Pietiläinen. The production of 'context' in information seeking research: a metatheoretical view. *Information Processing and Management*, 35(6):751–763, 1999.

[27] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Proc. NIPS*, 2003.

[28] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*, page 104, 2004.

[29] V. Vapnik and V. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

[30] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. ACL*, pages 189–196, 1995.

[31] T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. ICML*, pages 1191–1198, 2000.

[32] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2006.