A Semi-Supervised Approach to Modeling Web Search Satisfaction

Ahmed Hassan
Microsoft Research
Redmond, WA
hassanam@microsoft.com

ABSTRACT

Web search is an interactive process that involves actions from Web search users and responses from the search engine. Many research efforts have been made to address the problem of understanding search behavior in general. Some of this work focused on predicting whether a particular user has succeeded in achieving her search goal or not. Most of these studies have faced the problem of the lack of reliable labeled data to learn from. Unlike labeled data, unlabeled data recording behavioral signals in Web search is widely available in search logs. In this work, we study the plausibility of using labeled and unlabeled data to learn better models of user behavior that can be used to predict search success more effectively. We present a semi-supervised approach to modeling Web search satisfaction. The proposed approach can use either labeled data only or both labeled and unlabeled data. We show that the proposed model outperforms previous methods for modeling search success using labeled data. We also show that adding unlabeled data improves the effectiveness of the proposed models and that the proposed method outperforms other strong semi-supervised baselines.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search Process

Keywords

search engine evaluation, user behavior models, semi-supervised learning

1. INTRODUCTION

Finding effective evaluation metrics for information retrieval systems has always received plenty of attention. The need for good evaluation metrics increases as information retrieval systems advance leaving smaller rooms for improvements. Traditionally, per-query metrics such as Mean Average Precision (MAP), Precision @k, and Normalized Discounted Cumulative Gain (NDCG) have been widely used. Recently, researchers started using behavioral signals for evaluating information retrieval systems. Some of this work still considered query-level evaluation [25], while other looked at goal success [12]. Goal level success prediction is related to another body of work that tries to understand different patterns of user behavior in Web search like understanding the behavioral differences between novice and expert users [27], and studying how behavior changes in difficult search tasks [4].

A major problem facing these research efforts is the lack of labeled data to train effective learning algorithms. Hence, most of this work has either been limited to small-scale studies [8], or large-scale analysis of unlabeled data [27, 4]. Labeled data consists of all behavioral signals collected from a user in a search goal and a label indicating whether the user's information need was satisfied or not.

Some approaches have been proposed to overcome the limited availability of labeled data by building frameworks for collecting realistic labeled data. Hassan et al. [13] created a toolbar that monitors search behavior and collects explicit satisfaction ratings from users when they normally use Web search. The toolbar asks users to rate their search experience after every search goal. The main problem with this approach is the low adoption rates of users and the difficulty of keeping users engaged with the toolbar. Ageev et al. [1] suggested using a game like strategy where study participants are competing to find answers to questions using Web search in a given amount of time. Like the first approach, this approach suffers from the low adoption rates. It also creates artificial information needs, rather than monitoring the information needs of real user; which may affect user behavior. Human judges were used to collect labeled search goals in [12]. This process is noisy because judges have no information about the actual intent underlying the information need. Moreover, it is expensive and time consuming. Hence, it does not scale when a large amount of annotated data is needed.

Unlike labeled data, unlabeled data of behavioral signals is readily available in search logs. In unlabeled data, we have access to several signal describing the user behavior, but we do not know whether the user succeeded in satisfying her information goal or not. This suggests that algorithms that can learn from both labeled and unlabeled data could be very effective in modeling search success.

In this work, we look into ways of predicting success in search by modeling user behavior. We introduce a new ap-

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

proach that can use either labeled data only, or labeled and unlabeled data in a semi-supervised setting. We compare the proposed method to previous work on Web search success prediction using labeled data. We also compare it to strong baselines that apply other semi-supervised learning frameworks to the same problem. We discuss the advantages and limitations of the proposed method, and highlight the performance gains the proposed method achieves both in the supervised and semi-supervised settings. We also show empirically that adding unlabeled data improves prediction accuracy.

The rest of this paper will proceed as follows. In Section 2, we discuss related work. We provide a formal definition of our problem is Section 3. In Section 4, we describe a generative model for predicting success and show how it can be extended to the semi-supervised case. Finally we present experiments in Section 6, discussions in Section 7 and we conclude in Section 8.

2. RELATED WORK

Evaluating information retrieval is a challenging problem that received a lot of attention. Traditionally, search engines have been evaluated using classical methodologies that use query sets and relevance judgments. One of the most frequently used metrics for evaluating ad hoc information retrieval is Mean Average Precision (MAP). MAP is optimized to cover both recall and precision while taking the entire search result ranking into consideration. State of the art measurements of query result relevance use discounted cumulative gain (DCG) [18]. The intuition behind DCG is that users are only interested in the first few results and hence high early precision is desirable [14]. DCG metrics lend themselves to a user model of scanning a ranked list of results to some depth. DCG can be calculated using manual query URL relevance judgments, or estimated query URL relevance [2, 21]. Later, DCG evolved into the Normalized Discounted Cumulative Gain (NDCG) [17]. NDCG normalizes DCG across queries by sorting documents of a result set by relevance, computing the ideal DCG, and normalizing DCG with with the ideal DCG. DCG is additive in nature and assumes that documents are independent. Expected Reciprocal Rank [6] is another metric inspired by the cascade model that alleviates these problems. It is defined as the expected reciprocal length of time that the user will take to find a relevant document. Other metrics have been also used like Precision @k.

All those methods are query based metrics. They evaluate an information retrieval system by evaluating individual queries. However, a single user information need may result in one or more queries, and the same query may have different underlying information needs. Hence, individual query URL relevance may not always correlate with user satisfaction. Moreover, all these metrics use human relevance judgments which is expensive to collect and may be noisy because human judges do not know the exact intent behind an information need.

Mining search logs has been extensively studied by various researchers. For example, Jansen et al. [16] studied user query distribution, information needs, and sessions. Beitzel et al. [5] studied the change in popularity of topically categorized queries. White et al. [28] studied search trails including queries and page navigation from toolbar logs.

More specifically, other research has looked at mining search logs for measuring search quality. Fox et al. [9] found that a strong correlation exists between search log features and user satisfaction. They used Bayesian modeling and decision trees to model explicit satisfaction ratings using several kinds of features. They used clickthrough rate features, dwell time features, and features describing how users end their search sessions.

Huffman and Hochster [15] studied the correlation between user satisfaction and simple relevance metrics. They reported a relatively strong correlation between user satisfaction and linear models encompassing the query URL relevance of the first three results for the first query in the search task.

Radlinski et al. [26] showed that interleaving the results of two ranking functions and presenting the interleaved results to users is a good predictor of relative search engine performance. They also looked at using metrics including abandonment rate, reformulation rate, and time to first click to predict relative performance. They found out that those metrics do not perform as well as interleaving results. Using aggregated features extracted from user behavior is certainly correlated with satisfaction. However, it has been shown that modeling the transitions between actions performed by users during search is a stronger predictor of satisfaction [12].

White and Morris [27] tried to understand the behavioral differences between novice and expert users. They define expert users as those who use advanced operators. They show that the behavior of expert users is different compared to non-expert ones. Aula et al. [4] studied how user behavior changes in difficult search tasks. They performed a user study to understand how users behave with difficult search tasks. They found out that when faced with a difficult search tasks, users tend to use more diverse queries, use advanced operators, and spend more time on the search results page. These studies provide many interesting observations about user behavior. However, they do not try to model or predict success due to the lack of labeled data about goals and success.

Hassan et al. [12] showed that modeling action sequences representing user behavior is better than models based on the query URL relevance of the first three results for the first query. The main reasons are that different information needs sometimes underlie the same query, and that the first query does not tell the complete picture in terms of the entire search task. A user study where firsthand satisfaction ratings were directly collected from users, who opted in to use a special toolbar, was presented in [13].

Piwowarski et al. [24] presented a model that uses Bayesian Networks to predict document relevance in the absence of document content models. They presented a Bayesian Network approach to holistically model user behavior interactions, and used it to identify distinct patterns of search behaviors. These patterns, along with a list of custom features, were used to predict the relevance of a set of query document pairs.

Ageev et al. [1] proposes an extension of the Markov Model approach, described in [12], by augmenting the Markov Model with additional search features. They also used a game like strategy for collecting labeled data where they ask participants to find answers to questions using Web search. None of those methods has considered using unlabeled data to improve the effectiveness of the success prediction models. They all faced the problem of the lack of labeled data and

came up with different solutions: human judges [12], toolbars [13], and games [1]. The amount of data collected is still relatively small and does not allow further developments targeting specific types of information needs, verticals, or users.

Semi-supervised learning is a special form of machine learning where both labeled and unlabeled data are used in the learning process. Labeled data is usually expensive, and hard to obtain. This resulted in a huge interest in learning techniques that can use unlabeled data as well as labeled data. Semi-supervised learning is a very well established area. Some of the earliest semi-supervised methods in the literature is the self training wrapper algorithm which retrains a supervised model using unlabeled data that have been labeled by the same model with high confidence [3, 10]. The closely related concept of transductive inference has been studied in [11]. A very good introduction and survey of semi-supervised learning techniques could be found in [31, 7, 29]. We will show in this work how semi-supervised learning techniques can improve the effectiveness of success prediction algorithms by using labeled and unlabeled data.

3. PROBLEM DEFINITION

We start by defining some terms that will be used throughout the paper:

Definition: A *search goal/task* is a single information need that may result in one or more queries [20, 12].

Definition: *A search trail* is an ordered sequence of actions performed by the user during a search goal.

Definition: A *labeled dataset* is a collection of search goals associated with success labels. Every goal in the labeled data is labeled as successful or unsuccessful either by a human judge or by collecting firsthand labels from users conducting the search.

Definition: An *unlabeled dataset* is a collection of search goals without success labels. This data is easier to collect and does not need any human judges , or any special software to collect labels (i.e. browser plugins).

Assume we have a stream of queries submitted to a search engine. In response to each query, the engine returns a search results page. The user may decide to interact with this page by clicking on results, other search engine features, or by reformulating the query. Every query is a part of a *search goal* with the objective of satisfying a particular information need. Every search goal is represented with a search trail. A search trail is represented by an ordered sequence of user actions. Actions include all types of interactions performed by the users while using the search engine. Following [12], we use the set of actions described in Table 1. Examples of a successful and an unsuccessful goals are shown in Tables 2 and 3 respectively. In the successful goal, the user started with the query "*van gogh self portrait*", and quickly decided to click on an image results. She spent some time examining the result then she successfully ended her search. In the unsuccessful goal, the user typed a query, and briefly clicked on two search results. After that, she decided to switch to the images vertical, but the image vertical still did not have any relevant results so she reformulated her query. Apparently, that still did not satisfy her information need, and hence she gave up and ended her search.

The labeled data was collected during a user study where search behavioral signals and explicit success ratings were collected directly from users. This will be described in details in Section 5. This is better than asking human annotators to judge whether a user was successful with his search or not because judges do not have access to information about the actual user intent.

Action	Description
START	the user starts a new goal
Q	a query submission
RL	a related Search Click
SP	a spelling Suggestion Click
SR	a search result click
AD	a sponsored search result click
TB	a switch to a different tab (search vertical)
END	the user ends the search goal

Table 1: A list of possible user actions

	Action
Query	van gogh self portrait
IMG CLICK	url1
END	

Table 2: An Example of a Successful Goal

So given a search goal, our objective is to predict whether that goal ended up being successful or not. We adopt a semi-supervised learning approach to address this problem. Our training data X can be divided into X_l, for which labels Y_l are provided, and X_u, for which no labels are provided.

4. APPROACH

Generative models specify a joint probability distribution over observations and labels. Generative models are often used in machine learning for classification. A conditional distribution is formed from a generative model by use of Baye's rule. In this section, we define a generative model for user behavior and describe how it can be used for both supervised and semi-supervised learning.

4.1 A Generative Model for User Behavior

We assume that search trails are generated by a mixture model. The mixture model has two mixture components corresponding to *success* and *failure*. Every trail is generated using a probability distribution defined by the parameters of the mixture mode, denoted θ. Let there be a set of classes $C = [c_s, c_f]$ corresponding to the *success* and *failure* cases.

Actions in a search trail x_i are generated using the selected mixture component. Every trail x_i is generated according to a distribution $P(x_i|c, \theta)$. Hence the likelihood of seeing a trail x_i is the sum of its probability over the *success* and *fail* mixture components:

$$P(x_i|\theta) = \sum_{c \in [c_s, c_f]} P(c|\theta) P(x_i|c; \theta) \quad (1)$$

Every search trail x_i consists of a set of actions. For every action we chose the next action by drawing from the mixture distributions. Assuming every trail consists of a set of actions: $x = \{a_1, a_2, ..., a_n\}$ and that every action is dependent only on the previous action and independent of the trail length, then we can write:

$$P(x|c, \theta) \propto \prod_{j=1}^{n} P(a_j|a_{j-1}; c; \theta) \quad (2)$$

Action	
Query	steven colbert painting
CLICK	url1
CLICK	url2
TAB SWITCH	IMAGES
Query	steven colbert recursive painting
END	

Table 3: An Example of an Unsuccessful Goal

This leaves us with a set of parameters $\theta_{a,a',c} \equiv P(a'|a;c;\theta)$ for every pair of actions a and a', and every class c. In addition we have two more parameters θ_{c_s} and θ_{c_f} corresponding to the prior class probabilities $P(c_s|\theta)$ and $P(c_f\theta)$. Hence, the probability of generating any search trail x is:

$$P(x|\theta) \propto \sum_{c\in\{c_s,cf\}} P(c|\theta)\prod_{j=1}^{n} P(a_j|a_{j-1};c;\theta) \quad (3)$$

where x is a search trail consisting of a set of actions $x = \langle a_1, a_2, \ldots, a_n, \rangle$, n is the number of actions in trail x, and a_j is action number j in x . In our experiments, we model the trail length by introducing one multinomial distribution for each possible length. Alternatively, we can assume that trail length is identically distributed to reduce the number of parameters.

4.2 Supervised Classification

To use this model for classification, we need to learn an estimate $\hat{\theta}$ of the model parameters. To estimate these parameters, we may use maximum likelihood estimation (MLE) or maximum a posterior (MAP) estimates. One potential problem with MLE is that it may result in zero estimates of some parameters if the corresponding transition has not been seen in the training data. This problem is usually solved by smoothing. Alternatively, we can use MAP estimates with Dirichlet priors, which is identical to MLE with Laplace smoothing. Hence, the action transition parameters are estimated as:

$$\hat{\theta}_{a_i,a_j,c} \equiv P(a_i|a_{i-1};c;\hat{\theta}) = \frac{1+N_{a_i,a_j,c}}{|A|+N_{a_i,c}} \quad (4)$$

where $N_{a_i,a_j,c}$ is the empirical count of transitions from a_i to a_j in trails in class c, $N_{a_i,c}$ is the empirical count of a_i in the same trails, and $|A|$ is the total number of actions. Similarly, the class probabilities are estimated as:

$$\hat{\theta}_c \equiv P(c|\hat{\theta}) = \frac{1+N_c}{2+N} \quad (5)$$

where N_c is the fraction of trails in the training data with class c, and N is the size of the training set.

Given these parameters we can derive a conditional probability distribution to classify new instances given the generative model:

$$P(c|x_i,\hat{\theta}) = \frac{P(c|\hat{\theta})P(x_i|c;\hat{\theta})}{P(x_i|\hat{\theta})} \quad (6)$$

To classify a new trail we just need to compute:

$$PRED(x_i) = arg\,max_{c\in\{c_s,c_f\}} P(c|x_i,\hat{\theta}) \quad (7)$$

This model has three advantages compared the one presented in [12]: (1) the model proposes a more principled way of modeling user behavior, (2) the Markov model presented in [12] makes decisions based on trail likelihood ($P(x|c;\theta)$), while the model we present here uses posterior class probabilities instead ($P(c|x,\theta)$). We will show later the advantages of taking the prior distribution into consideration, and (3) this model can be extended to the semi-supervised case as will be shown in the next sections.

4.3 Modeling Time

We tried two different approaches to modeling time. In the first approach, we treat transition time as a continuous variable drawn from a Gamma distribution with parameters k, and θ. We compute the distribution parameters for every class (i.e. success or failure), and every transition. Then, the probability of some value given a class, $P(t = v|c)$, can be computed by plugging v into the equation for the Gamma distribution parameterized by the parameters we computed earlier.

In the second approach, we replace every click action (e.g. algorithmic result click, sponsored result click, etc.) with one of three different actions based on the click dwell time. For example instead of the action "*SR*", which denote a search result click, we have "*SR-short*" , "*SR*", and "*SR-long*". Long clicks are defined to have a dwell time for longer than 30 seconds and have been shown to correlate with satisfaction, as in previous work [9]. Similarly, short clicks are clicks with a quick bounce back to the SERP and have dwell time on the landing page of under 15 seconds [9]. SERP view actions (e.g. query submission, related search click, etc.) are replaced with one of two actions: short if dwell time is less than 20 seconds, and long otherwise. Short time to first click, which is the difference between the timestamp from serving up the page and the timestamp of the first user click on the page, has been previously shown to be correlated with satisfaction [26].

4.4 Semi-Supervised Classification

The model we described in the previous subsections uses labeled data only to estimate the model parameters. Due to the high cost of obtaining labeled data for this task, and the wide availability of unlabeled data, we would like to extend the model such that it uses labeled and unlabeled data for learning. The model parameters, $\hat{\theta}$ have been estimated using MAP estimations. This cannot directly handle unlabeled data. However, we can do so by using the expectation maximization (EM) algorithm. Semi-supervised classification with EM has been successfully applied to different domains before, including text categorization [23].

The EM algorithm is an iterative algorithms which can be used to find the MAP estimates of parameters in statistical modeling. The algorithm has two main steps: the *Expectation* step and the *Maximization* step. The Expectation (E) step of the algorithm computes the expectation of the missing values; in this case the class membership distribution of the unlabeled data. The Maximization (M) step re-estimates the parameters by maximizing the likelihood of the parameters using the previously computed expectations. The algorithm works as follows:

- Estimate the parameters $\hat{\theta}$ from labeled data only
- Loop while the parameters improve:
  - **E**: Use the current classifier to estimate $P(c_s|x_i;\hat{\theta})$, and $P(c_f|x_i;\hat{\theta})$ for all unlabeled trails.

– **M**: Re-estimate model parameters, $\hat{\theta}$, using the labeled data and the component memberships of the unlabeled.

To take the component membership of the unlabeled data into consideration during training, we modify Equation 4 to:

$$\hat{\theta}_{a_i,a_j,c} \equiv P(a_i|a_{i-1};c;\hat{\theta}) = \frac{1+\sum_{x_i \in X} \delta(x_i,c)N(x_i,a,a')}{|A|+\sum_{x_i \in X} \delta(x_i,c)N(x_i,a)} \quad (8)$$

where X is the set of training data, $N(x_i,a,a')$ is the number of transitions from a to a' in x_i, and $N(x_i,a)$ is the number of occurrences of a in x_i. For the labeled data, $\delta(x_i,c)$ is 1 if x_i is in class c and 0 otherwise. For the unlabeled data, $\delta(x_i,c)$ is the component membership probabilities calculated in the previous E step.

5. DATA

Our data includes search behavioral signals and explicit success ratings collected directly from users. This data was collected during a user study where firsthand success ratings were collected from users. This is better than traditional ways of collecting data that use annotators to judge whether a user has been successful or not [12]. The latter method introduces more noise in the labels because judges lack information about the actual intent behind the search goal.

Study participants were instructed to install a toolbar. The toolbar detects when a user submits a query to Google, Bing, or Yahoo search engines. It then monitors and records all actions performed by the user during search. Participants were instructed to submit a binary success rating at the end of every search goal. Participants were also allowed to ignore a search goal, in which case, no data was collected about that goal.

Every log entry contained a user identifier, a time-stamp for every page view, and the URL of the visited page. Page views included query submission, search result clicks, navigation beyond the search results page originating from clicks on links in a search result, and clicks on other search engine features (e.g. spelling corrections, related searches, etc.). Secure (https) URLs have not been collected and any identifiable information was removed from the data prior to any analysis.

More than 10k unique searches were collected a long with a label indicating whether the search was successful or not. Every record included a search trail, and a success label. A search trail originates with the submission of a query to a search engine and contains all queries and post-query navigation trails [27]. A search trail always begins with a query and ends when the information seeking activity stops. A search trail is represented by an ordered sequence of user actions as described earlier.

6. EXPERIMENTS

6.1 The Supervised Setting

6.1.1 Baselines

We compare the proposed approach to three baselines in the supervised setting. The first baseline poses the problem as a classic machine learning problem where a set of features adopted from previous work, [26, 12], are used to train a logistic regression classifier . The set of features we used are summarized in Table 4

Features
Number of queries
Number of clicks of all types
Number of algorithmic results clicks
Number of sponsored results clicks
Number of answer results clicks
Number of pagination clicks
Number of spell suggestion clicks
Number of related queries clicks
Average time between clicks
Time span of goal
Average time to first click
Average dwell time

Table 4: Static features for predicting success.

The second baseline is the Markov model approach described in [12]. In this approach, two Markov models are trained to characterize the behavior of users in successful and failed goals. Features are described by a sequence of actions as described earlier. A new goal x is classified as successful if $P(x|success) > P(x|failure)$.

The last baseline uses Conditional Random Fields to model success [1]. We ran our approach on the data used in [1]; which was made publicly avaiable. We will report the results from [1] and the results obtained by our approach using the same data.

6.1.2 Experimental setup

We perform experiments using the data described in Section 5. We used four supervised methods; the generative model presented in this paper and three baselines. To compare supervised methods, we report accuracy using 10-fold cross-validation. Folds were created based on user ids. We used the entire dataset described in Section 5. Users reported success in approximately 80% of the goals in this dataset and the rest reported failure. A different dataset was used for the comparison against the CRF method, presented in [1], as will be described later.

6.1.3 Results

We start by comparing the two approaches we described for modeling time and found that the difference in performance was not statistically significant at the 0.05 level according to the Wicoxon *p-value*. We will use the approach based on time discretization in all following experiments.

We then compared the performance of all supervised methods using 10-fold cross validation over the entire dataset. Figure 1 shows the performance of the proposed method and the two baselines. We notice that the proposed approach outperforms the two baselines and the difference is statistically significant at the 0.05 level. We notice that the static features baseline has the worst performance. This agrees with findings reported in previous work because the static features describe an aggregate view of user behavior and misses many important pieces of information. We also see performance gains compared to the Markov model approach. Modeling the posterior probability of search goals ($P(c|x,\theta)$ yields better results that modeling the likelihood ($P(x|c,\theta)$ especially because the dataset is unbalanced with respect to the proportion of successful and failed goals This

imbalance is typical in any random sample of query impressions because most users achieve their search goals and dissatisfaction is usually the exception. For example, Ageev et al. [1] performed a study where users were asked to find answers to specific questions using Web search. In 87% of the tasks, users reported success by finding and reporting an answer. Hassan et al. [12] asked judges to label search goals as successful or not and reported that more than 70% of the goals were labeled as successful. In our dataset, where first-hand success labels were collected from users, approximately 80% of the goals ended with users reporting that their information need was satisfied. When we had a closer look at the performance of the proposed generative model and the Markov model of [12], we noticed that the performance slightly improved for the majority class (satisfied goals) and significantly improved by over for the minority class (dissatisfied goals).

We also compared the performance of our approach to the CRF approach presented in [1]. We used the same dataset and the same experimental setup described in [1]. They used a game like strategy where study participants are competing to find answers to questions using Web search in a given amount of time. The data was collected in 4 different game rounds. We used 4-fold cross-validation where the data from three games was used for training and the data from the fourth was used for testing. Ageev et al. [1] defined four different success criteria: 1) the user submitted an answer and the answer was correct, 2) the user submitted an answer that could be correct or not, 3) the user visited a good URL that has relevant information to the question, and 4) the user visited a good URL and it was last in the search session. Like [1], we measured accuracy and macro-averaged F-measure over the successful and unsuccessful results. For the Markov model and the generative model approaches. We modeled time using the discrete approach described in Section 4.3. The results are shown in Table 5. The table shows that both the CRF model and the generative model outperform the Markov model. The gap between the performance of the generative model and the Markov model increases as the data imbalance increases as expected. The performance of the CRF and the generative model depends on the success definition. The former is better for some definitions and the latter is better for others. The difference in performance is very small in all cases.

6.2 The semi-supervised setting

We compare the proposed approach to two baselines. The first baseline is a transductive discriminative model trained on features describing the transition between different actions. The second baseline is a graph based algorithm based on random fields and harmonic functions. In the rest of this section, we describe the baselines, experimental setup and results.

6.2.1 Baseline 1: Transductive SVM

In this baseline, we can use the action transitions (i.e. transition from a to a' in search trails) as features and train a discriminative model. The objective of this model is to classify each search trail into one of two categories (success or failure). The first step in this process is to transform search trails into a representation suitable for the classification task. One way to approach this problem is to compute a set of aggregate features to describe the user behavior from the search trail (e.g. number of clicks, number of queries, average dwell time, etc.). Hassan et al. [12] looked at this approach and showed that directly modeling transitions is better than using aggregate features. They also showed that transitions subsume information in aggregate features.

Instead of using aggregate features, we can derive features that directly represents the search trail without losing much information. We can use the transition between every two consecutive actions as a feature. For example, given the search trail: *"Q Q SR END"* where *Q*, *SR*, and *END* correspond to query submission, search result click, and goal termination respectively, we can extract the following three features: *Q-Q*, *Q-SR*, and *SR-END*.

Generally, we will have $|A|^2$ features where A is the set of possible actions. Every feature corresponds to a possible transition between two states. Every trail will be represented using those features. Feature values are simply the number of times every transition has been observed in the trail. The feature vector is expected to be sparse with most of the features having a zero value. Transductive support vector machines [19] extend SVMs in that they could handle both labeled and unlabeled data in a semi-supervised setting. SVMs use a set of labeled data, X_l, and a set of corresponding labels Y_l. SVM training algorithm builds a model that assigns new examples into one category or the other. In that, SVMs try to induce a general decision function for the learning task. Transductive SVMs use an unlabeled set X_u along with X_l, and Y_l. TSVMs take into account a particular set of unlabeled data, X_u, and try to minimize misclassification of this set of unlabeled data.

Joachims [19] used TSVMs for text classification and proposed an algorithm for training TSVMs efficiently. Joachims [19] argues that TSVMs are suitable for text classification because of the high dimensional input space and the feature vector sparseness. Although our feature space is smaller than typical feature spaces in text classification, we believe TSVMs is a strong baseline for this problem.

6.2.2 Baseline 2: Harmonic Functions

Another popular class of semi-supervised learning algorithms is the graph based methods. Graph based learning algorithms rely on building a graph representation of labeled and unlabeled data points. Edges in the graph encode pairwise similarities between points. Graph based algorithms have been successfully applied to a range of problems in natural language processing, computer vision, and computational biology. Graph based methods are discriminative and transductive in nature.

We construct a graph $G(V, E)$, where V is the list of labeled and unlabeled trails. E is a set of edges connecting similar points. Inspired by the bag of words model for representing text documents, we represent every trail using a bag of transitions. Every transition has a count associated with it. Each transition corresponds to a dimension in high dimensional space, and every trail is a vector in that space. The frequency of every transition is used as its weight. Let A be the set of possible user actions, $|A|$ be the number of such possible actions, and $N(x_i, a, a')$ be the number of transitions from a to a' in x_i. Then every trail can be represented as vector as follows

$$\vec{x_i} = (N(x_i, a_1, a_1), N(x_i, a_1, a_2), \ldots, N(x_i, a_n, a_n)) \quad (9)$$

We use the popular Cosine similarity measure to compute similarity between user behavior in different goals. The vector representation of trails allows us to use the Cosine similarity measure to compute similarity between any two given trails. The Cosine metric measures the similarity by computing the cosine of the angle between the two vectors representing the search trails. Formally, this can be written as:

$$Cos(\vec{x_a}, \vec{x_b}) = \frac{\vec{x_a} \cdot \vec{x_b}}{|\vec{x_a}| \times |\vec{x_b}|} \quad (10)$$

To construct the graph, we create a node for every search trail. Every pair of nodes is connected by an edge. The weight of edges comes from one of the similarity measure described above. The graph could be also represented by an $n \times n$ symmetric matrix, where n is the number of search trails. Every element $M(i, j)$ represents the similarity between trail x_i and trail x_j.

Zhu et al. [30] presented a semi-supervised graph learning approach based on Gaussian random fields and harmonic functions. The similarity graph we described earlier is used to represent both labeled and unlabeled data. The learning problem is then formulated using the Gaussian random field on this graph, with the mean of this field characterized using harmonic functions [30].

The method assumes n data points, l of which are labeled, and the rest u are unlabeled. Consider the graph $G = (V, E)$ built as described above. We represent the graph using a symmetric $n \times n$ matrix W. The objective of this method is to compute a real valued function $f : V \rightarrow \mathbb{R}$. The values of f are fixed for labeled data and equal the class label. The value of f for unlabeled data will be computed and used to assign labels to unlabeled data. Zhu et al. shows that the function f that minimizes the eneregy function $E(f)$ is harmonic [30]. It is intuitive that the function f is expected to have similar labels to points that are close in the graph. Hence, they used the quadratic energy function:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} (f(i) - f(j))^2 \quad (11)$$

The harmonic solution is computed explicitly as follows:

- Let $D = diag(d_i)$ be the diagonal matrix with entries $d_i = \sum_j w_{ij}$, where $W = [w_{ij}]$ is the weight matrix.
- Because f is harmonic, it can be expressed as $f = Pf$, where $P = D^{-1}W$.
- Split W, and similarily D and P into 4 blocks:

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}$$

- Let $f = \begin{bmatrix} f_l \\ f_u \end{bmatrix}$, where f_u denotes the values of f for the unlabeled data.
- The harmonic solution is given as:

$$f_u = (D_{uu} - W_{uu})^{-1} W_{ul} f_l = (I - P_{uu})^{-1} P_{ul} f_l$$

6.3 Experimental Setup

We perform experiments using the data described in Section 5. We compare the performance of three semi-supervised methods. The first is the proposed approach that uses expectation maximization to extend a generative mode to handle labeled and unlabeled data. The second uses Transductive Support Vector Machines (TSVMs), and the third is a graph based model that uses Gaussian random fields and harmonic functions. We carried out the experiments in the transductive settings, following [7]. In this setting, the test set coincides with the set of unlabeled points (50% of the data). This has two main advantages. First, it is economical in terms of the required data. Second, it allows us to compare different methods including those that are naturally transductive. We compare every semi-supervised method to a corresponding supervised method to assess the benefit of adding the unlabeled data. We also compare the semi-supervised methods to one another. To make sure that the reported results are independent of labeled points selection, we select 10 different sets of labeled data for every number of labeled data points at random and report the average of the 10 runs.

6.4 Results

In this section, we assess the benefit of using unlabeled data for every model. We compare every semi-supervised method (the proposed method and the baselines) to a corresponding supervised method. We compare the supervised and the semi-supervised versions of the proposed approach. We compare the Trasductive SVM baseline to an inductive SVM classifier, and the harmonic functions baseline to a k-neaerest neighbor method using the same graph. The objective of this experiment is to estimate the benefit of using unlabeled data regardless of the semi-supervised model used. We start with SVM and TSVM. Figure 2 compares the performance of both methods while varying the amount of labeled data. We notice a consistent improvement over using labeled data only when unlabeled data is introduced. The improvement is bigger when the number of labeled points is small and gets smaller quickly as more labeled data is introduced.

We notice similar behavior when we compare semi-supervised learning using harmonic functions to the K-nearest neighbor method in Figure 3. Interestingly, we notice that adding the unlabeled data hurts the performance when the number of labeled points is very small. Otherwise, adding labeled data improves performance. Like TSVM, the benefit of adding unlabeled data decreases as more labeled data is introduced. The benefit of using harmonic functions with labeled and unlabeled data over using KNN is considerably bigger than the benefits of using TSVM over SVM.

Figure 4 compares the performance of the semi-supervised generative model that uses EM and the supervised generative mode. We notice similar behavior to the two previous cases where using EM and unlabeled data improves performance. Unlike the graph based model, the improvement is consistent regardless of the amount of labeled data. The improvement is also considerably larger than the TSVM case.

We also compare the performance of the three semi-supervised learning algorithm in Figure 5. The figure compares the performance of the generative model with EM, the Transductive Support Vector Machines (TSVMs) model, and the graph based model that uses Gaussian random fields and harmonic functions using a Cosine similarity graph. We notice that the graph based model performs poorly when the amount of labeled data is limited. The two other methods

Success Definition	Majority		MM		GM		CRF	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Answer Correct	65	40	68	61	67	60	68	60
Answer Sub.	87	47	83	64	88	65	86	64
Good URL	83	45	78	60	83	65	80	57
Good URL Last	60	38	67	66	67	67	68	66

Table 5: The Majority Baseline (MB) vs. the Markov Model (MM) approach vs. the Generative Model (GM) approach vs. the CRF approach using data from [1].

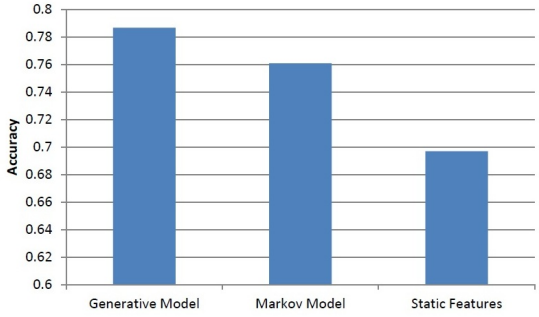

Figure 1: Performance of all supervised methods using 10 fold cross validation.

perform better with the the generative model with EM outperforming all other models with a large margin. As more labeled data is introduced the improvement margin starts to decrease.

The experiments show that adding unlabeled data considerably increases the prediction power of the models. This gain is achieved regardless of which semi-supervised model is used. When we compare the three semi-supervised mode, we notice that both the EM model and the TSVM model perform better than the other baseline and their performance is pretty close to one another. The EM model has an advantage over the TSVM and the graph based models which are transductive in nature. Transductive models need to be retrained for every new test set. The generative EM model does not suffer from this limitation.

Adding more unlabeled data points is shown to improve performance in Figure 6. In this experiment, we fixed the number of labeled points at 250, varied the number of unlabeled points and observed the performance. The figure indicates that adding more unlabeled data improves performance for all methods.

In summary, we have shown that using a semi-supervised framework improves performance regardless of the algorithm. We have also shown that the EM model and the TSVM model perform better than the graph based model, and that the EM model has an advantage over the other models because it does not have to operate in a transductive setting.

7. DISCUSSION

We studied the problem of Web search success prediction. Our study is different from previous work in that the methods we proposed can learn from both labeled and unlabeled data. The area of using behavioral signals to model success in Web search is rather new. All previous work has posed the problem as a supervised learning problem where labeled data is used to train a model that can be used later for predictions. We have discussed the problems associated with collecting labeled data for search success prediction and how the process is time consuming, expensive, and sometimes not very reliable. In the meantime, unlabeled data could be easily and quickly collected. We showed how learning from labeled and unlabeled data can improve the performance of search success prediction models. We proposed different methods for incorporating unlabeled data in the learning process. We showed that adding unlabeled data significantly improves performance. We also compared the performance of different methodologies for learning from labeled and unlabeled data and highlighted the strength and weakness of every method. In addition, we also proposed a novel supervised learning model for search success prediction. We showed that this model outperforms or achieves a comparable perfromance compared to the state of the art methods for predicting search success.

We performed several experiments comparing the performance of several supervised and semi-supervised models for predicting success in search. Now, we try to summarize our finding for both the supervised and the semi-supervised cases. For the supervised case, the proposed model, and the Markov model [12] perform much better than a classifier that used a set of static features. The static features classifier aggregates features describing the user behavior. It performs poorly compared to the other models because the other models have a more accurate picture of user behavior by modeling action sequences. The proposed model is more

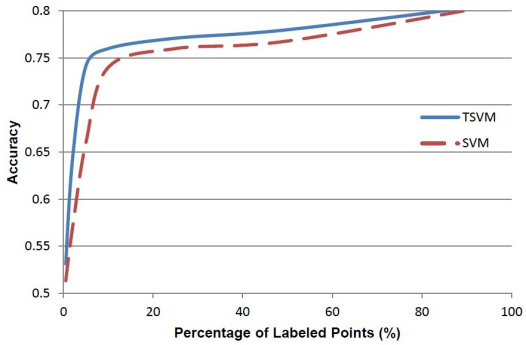

Figure 2: Performance of SVM vs. TSVM for different numbers of labeled data points.

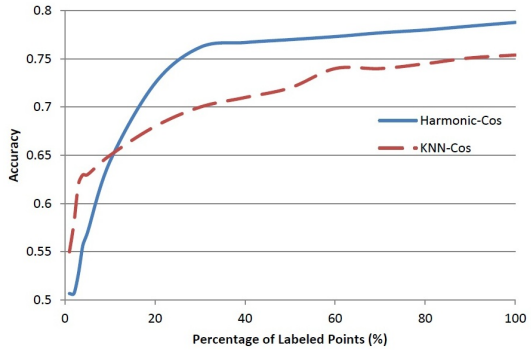

Figure 3: Performance of Harmonic Functions vs. KNN for different numbers of labeled data points.

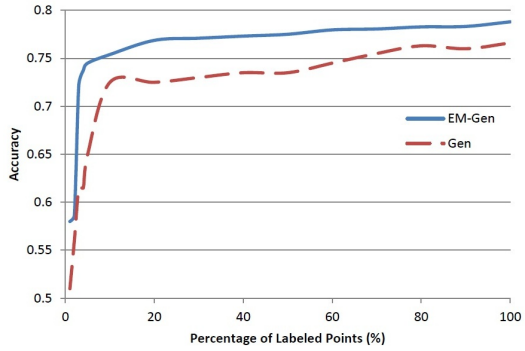

Figure 4: Performance of Generative Model vs. Generative Model with Expectation Maximization for different numbers of labeled data points.

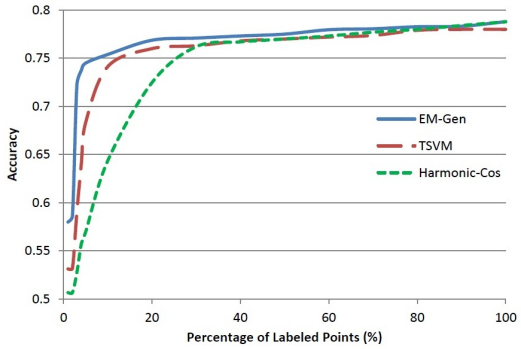

Figure 5: Performance of all semi-supervised models for different numbers of labeled data points.

principled and performs better than the Markov model [12]. The Markov model is intuitive and powerful, but it uses the likelihood of the action sequence representing a search goal, rather than the posterior probability as is the case with the proposed method. The posterior probability takes the prior into consideration which was shown to be important especially because of the inherent imbalance in the data distribution.

In the semi-supervised case, the generative model with EM performs best when the the labeled data is scarce. This is consistent with [22] that states that generative models have superior performance when only a few training examples are available. When the number of labeled data is increased, the performance of all algorithms is similar. One advantage for the generative model with EM over TSVM is that TSVMs are very slow to train and cannot handle large amounts of data. The graph based model is also pretty slow because it involves computing a pairwise similarities for all data points. Another advantage for the generative model is that it is inductive by nature. However, other models may be used in an inductive setting as well but possibly with some performance loss.

Our results show that incorporating unlabeled data in the user satisfaction prediction task improves classification. Previous work on user satisfaction prediction and other related tasks have faced the problem of the limited avaiability of labeled data for supervised learning. Our results suggest that

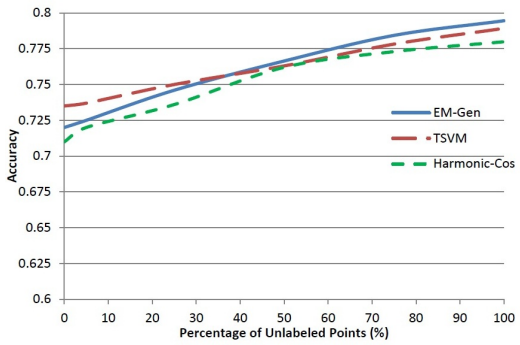

Figure 6: Performance of All Semi-supervised methods for different numbers of unlabeled data points.

semi-supervised methods in general, not only the proposed one, should be seriously considered in the future to predict satisfaction and other similar tasks.

Our findings suggest several lines of future directions for this work. The fact that adding unlabeled data can improve the performance will allow us to build more specific success prediction models that target specific verticals or specific types of queries. Moreover, this may allow us to build personalized success models tailored to the behavior of different users. We also plan to compare the predictive power of different transitions along the search trail. A preliminary study showed that looking at the entire trail is more informative that any parts of it, but we believe that we can develop better models by further studying this problem. One other direction of future research is to try to incorporate other query and SERP features like the ones presented in [1] in the semi-supervised models.

8. CONCLUSIONS

We presented novel methodologies for predicting success in Web search by analyzing user behavior. Collecting large amounts of labeled data for search success prediction is time consuming and sometimes not reliable. We showed that using semi-supervised methods, where both labeled and unlabeled data are used, improves the prediction acuracy. We presented several methods for incorporating unlabeled data and we discussed the advantages nad disadvantages of each. The results suggest that semi-supervised methods in general should be considered in the future for this task and other similar tasks.

9. REFERENCES

[1] M. Ageev, Q. Guo, D. Lagun, and E. Agichtein. Find it if you can: a game for modeling different types of web search success using interaction data. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, SIGIR '11, pages 345–354, New York, NY, USA, 2011. ACM.

[2] E. Agichtein, E. Brill, and S. T. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR 2006: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM.

[3] A. Agrawala. Learning with a probabilistic teacher. *Information Theory, IEEE Transactions on*, 16(4):373 – 379, jul 1970.

[4] A. Aula, R. M. Khan, and Z. Guan. How does search behavior change as search becomes more difficult? In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 35–44, New York, NY, USA, 2010. ACM.

[5] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 321–328, New York, NY, USA, 2004. ACM.

[6] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM.

[7] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[8] H. A. Feild, J. Allan, and R. Jones. Predicting searcher frustration. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 34–41, New York, NY, USA, 2010. ACM.

[9] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23, 2005.

[10] S. Fralick. Learning to recognize patterns without a teacher. *Information Theory, IEEE Transactions on*, 13(1):57 – 64, jan 1967.

[11] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *In Uncertainty in Artificial Intelligence*, pages 148–155. Morgan Kaufmann, 1998.

[12] A. Hassan, R. Jones, and K. L. Klinkner. Beyond dcg: user behavior as a predictor of a successful search. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 221–230, New York, NY, USA, 2010. ACM.

[13] A. Hassan, Y. Song, and L.-w. He. A task level metric for measuring web search satisfaction and its application on improving relevance estimation. In *Proceedings of ACM 20th Conference on Information and Knowledge Management (CIKM 2011)*, 2011.

[14] D. Hawking, N. Craswell, P. Thistlewaite, and D. Harman. Results and challenges in web search evaluation. In *Proc of WWW '99*, pages 1321–1330, 1999.

[15] S. B. Huffman and M. Hochster. How well does result relevance predict session satisfaction? In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 567–574, 2007.

[16] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.*, 36:207–227, January 2000.

[17] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 41–48, New York, NY, USA, 2000. ACM.

[18] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[19] T. Joachims. *Making large-scale support vector machine learning practical*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[20] R. Jones and K. Klinkner. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM 2008)*, 2008.

[21] S. Jung, J. L. Herlocker, and J. Webster. Click data as implicit relevance feedback in web search. *Information Processing and Management (IPM)*, 43(3):791–807, 2007.

[22] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[23] K. Nigam, A. Mccallum, and T. M. Mitchell. *Semi-Supervised Text Classification Using EM*, chapter 3. MIT Press, 2006.

[24] B. Piwowarski, G. Dupret, and R. Jones. Mining user web search activity with layered bayesian networks or how to capture a click in its context. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, 2009.

[25] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 43–52, New York, NY, USA, 2008. ACM.

[26] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In J. G. Shanahan, S. Amer-Yahia, I. Manolescu, Y. Zhang, D. A. Evans, A. Kolcz, K.-S. Choi, and A. Chowdhury, editors, *CIKM*, pages 43–52. ACM, 2008.

[27] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[28] R. W. White and J. Huang. Assessing the scenic route: measuring the value of search trails in web logs. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 587–594, 2010.

[29] X. Zhu. Semi-supervised learning literature survey. University of Wisconsin-Madison - Computer Science TR 1530.

[30] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fie'lds and harmonic functions. In *International Conference on Machine Learning*, 2003.

[31] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009.