Catching the Drift: Learning Broad Matches from Clickthrough Data

Sonal Gupta*
Dept. of Computer Sciences
The University of Texas
Austin, TX, USA
sonaluta@cs.utexas.edu

Mikhail Bilenko
Microsoft Research
One Microsoft Way
Redmond, WA, USA
mbilenko@microsoft.com

Matthew Richardson
Microsoft Research
One Microsoft Way
Redmond, WA, USA
mattri@microsoft.com

ABSTRACT

Identifying similar keywords, known as broad matches, is an important task in online advertising that has become a standard feature on all major keyword advertising platforms. Effective broad matching leads to improvements in both relevance and monetization, while increasing advertisers' reach and making campaign management easier. In this paper, we present a learning-based approach to broad matching that is based on exploiting implicit feedback in the form of advertisement clickthrough logs. Our method can utilize arbitrary similarity functions by incorporating them as features. We present an online learning algorithm, Amnesiac Averaged Perceptron, that is highly efficient yet able to quickly adjust to the rapidly-changing distributions of bidded keywords, advertisements and user behavior. Experimental results obtained from (1) historical logs and (2) live trials on a large-scale advertising platform demonstrate the effectiveness of the proposed algorithm and the overall success of our approach in identifying high-quality broad match mappings.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.5 [**Online Information Services**]: Commercial Services

General Terms

Algorithms, Experimentation

Keywords

Keyword-based advertising, online learning, keyword similarity

1. INTRODUCTION

In pay-per-click keyword advertising, advertisements are submitted as bids on specific phrases, corresponding to the maximum amount an advertiser is willing to pay for a user's click on the advertisement. Whenever the bidded keywords are found in the delivery context, advertisements are selected via real-time auctions that take into account the bids and estimated probability of click (also known as clickthrough rate). The delivery context is defined by the advertising type: in search advertising, it is comprised of the query submitted by a user to a search engine, while in contextual advertising, the context includes various properties of the viewed web page, such as its text, anchor text of incoming links, or search queries for which the page is among top-ranked results. Keyword-based advertising has been shown to have higher user satisfaction and response rates compared to display advertising [35], and has increased its share of the overall online advertising revenue to 44% for the first six months of 2008, up from 41% in 2007 and 40% in 2006 [1].

Identifying all phrases relevant to a campaign is a difficult task for advertisers, and to address this challenge, all major keyword advertising platforms currently offer the option of submitting *broad match* bids on keywords. Broad match, also known as advanced match, identifies keywords related to those found in the delivery context, and allows advertisements bid on related keywords to compete in the auction. Figure 1 provides a high-level diagram of broad match's role in advertisement delivery, and Table 1 illustrates several examples of keywords and their broad match mappings. Broad match is beneficial for both the advertising platform and the advertisers. With broad match, advertisers obtain increased coverage and reach a larger (yet targeted) audience, while the burden of campaign management is decreased since there is no need to identify all keywords that are relevant to a campaign. For the advertising platform, broad match leads to increased competition in auctions and higher monetization of traffic. Overall, it maximizes the utilization of both advertisement and content inventory, removing a core inefficiency from keyword-based pay-per-click Internet monetization.

In this paper, we propose a machine learning approach to combining various existing similarity measures for broad match keyword identification that relies on implicit feedback: training is based on previously shown broad match advertisement impressions, with the clicks (or lack thereof) providing the weak supervision. Clickthrough data is available in abundance from advertising system logs, while human judgments are costly and difficult to collect in large quantity. Although clickthrough data is noisy, noise is also

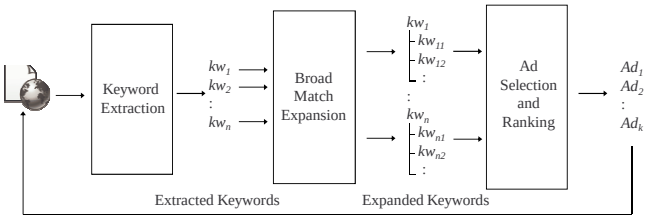

Figure 1: Broad matching in keyword-based advertising

*Work performed while at Microsoft Research

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'09, June 28–July 1, 2009, Paris, France.
Copyright 2009 ACM 978-1-60558-495-9/09/06 ...$5.00.

Table 1: Examples of broad match mappings

Context keyword	Matched Keywords
electric cars	*toyota prius, hybrid, cheap car, golf carts*
las vegas	*las vegas shows, vegas, las vegas hotels*
rihanna	*ashanti, justin timberlake, mariah carey*

known to be an issue for human relevance [33], while much wider availability of clickthrough data makes it a competitive source of supervision.

A key problem for learning in any online advertising task is adaptation to the rapidly changing distributions of both advertisements and the context in which they are shown. As both campaigns and inventory fluctuate significantly, it is critical for the learning algorithm to continuously adapt to the drift in the underlying keyword, advertisement and user distributions. This challenge is made more difficult by the very large traffic loads on advertising systems, making computational efficiency a critical requirement for any approach that could be deployed in a production system.

We describe a new online algorithm, Amnesiac Averaged Perceptron, that addresses all of these challenges. It is based on a highly efficient discriminative online learner, which we modify to allow quick adaptation to the changes in market dynamics and users' behavior, leading to strong accuracy improvements while maintaining computational efficiency, as shown by comparisons with batch learning at varying time intervals. The method continuously leverages weak supervision obtained from the ad delivery system without the need to obtain labeled data from human judges, and is suitable for deployment in high-throughput advertising platforms.

We evaluate the proposed approach to learning broad match mappings on historical data as well as on live traffic on a large-scale contextual advertising platform. First, we utilize a large dataset of contextual advertising logs that are "replayed" to provide realistic train-test experimentation. Results demonstrate that incorporating amnesia to overcome distribution drift leads to dramatic improvements in accuracy, with online learning significantly outperforming batch-learning configurations. Second, we present empirical results from deploying our algorithm on a live contextual advertising system alongside several alternative approaches, and compare their performance in terms of both clickthrough and monetization.

The remainder of the paper is organized as follows. Section 2 discusses related work, followed by Section 3 that sets up the task of identifying broad match mappings via prediction of click probability. In Section 4, we explain the proposed learning-based approach to identifying broad match keywords and describe the Amnesiac Averaged Perceptron algorithm. In Section 5, we discuss the data, methodology and results of experimental evaluation. We suggest future research directions and present our conclusions in Section 6 and Section 7, respectively.

2. RELATED WORK

The general problem of identifying related textual entities and estimating their similarity has been studied extensively in the context of many tasks that span areas from bioinformatics to databases to natural language processing. Traditionally, algorithms for computing string similarity have largely relied on the syntactic representation of the strings, most popular of them being variants of edit distance [19] and variants of TF-IDF cosine similarity [11].

In recent years, the problem of identifying related strings has attracted increasing attention in the context of web-related tasks, such as query substitution and suggestion [37], spelling correction [13], and named entity disambiguation [5]. For these tasks, a number of approaches have been proposed that leverage web-based information, such as result pages of search engines [31], successive queries from query logs [22, 6], and co-occurrence in web-based corpora such as Wikipedia [5].

In recent work, Jones et al. [22] and Radlinski et al. [28] introduced the problem of identifying broad match mappings in the context of search advertising, and described alternative learning-based approaches for finding broad matches for popular queries. These methods rely on human-labeled relevance judgements for training models that identify related keywords using several similarity measures as features. The key difference between these approaches and one described in this paper is that our method does not assume availability of human supervision, instead deriving training data from the clickthrough logs, which allows training continuously, while the methods in [22, 28] require offline training. As demonstrated by experiments in Section 5, the ability to quickly adapt to the fast-changing content and advertisement distributions is critical for obtaining significant performance improvements. We also note that the approaches of Jones et al. and Radlinski et al. can be combined with our algorithm, with all algorithms providing candidate broad match pairs.

The general approach of combining multiple similarity functions for computing textual similarity has been previously described in the context of a number of tasks, e.g., record linkage [32, 3]. In web-related tasks, several learning-based approaches have been proposed for obtaining phrase similarity measures based on sources of evidence that include search result pages produced for phrases entered as queries [31], language modeling [26], and combining multiple evidence sources [37]. These approaches are all compatible with our algorithm, since their outputs can be utilized as features.

The key distinction of our approach is that it relies on implicit feedback. While information retrieval community has traditionally relied on explicit relevance ratings produced by human judges for evaluation and training [34], in recent years there has been increasing attention to leveraging implicit feedback contained in logs of users behavior. A variety of features derived from implicit feedback have been shown to be useful for evaluation and training primarily in the context of improving ranking of search results and advertisements, e.g., click behavior [20, 2, 21, 8, 10], query sequences [22, 17], page dwell time [2], and mouse movements [18]. It has also been shown that implicit and explicit feedback measures can be correlated for specific measures, such as DCG vs. clickthrough [7] and binary relevance judgements vs. visitation counts [36].

Advertising clickthrough logs have been central in recent work of Chakrabarti et al. [8] and Ciaramita et al. [10] who relied on them to improve ad ranking. These approaches are complementary to ours, as they tackle a problem at a different stage of the advertisement delivery pipeline shown in Figure 1 – selection and ranking of advertisements selected for provided keywords – while our work focuses on expanding the set of keywords used to fetch advertisements for subsequent ranking. Estimating clickthrough rates for ad ranking is also addressed by the work of Regelson and Fain [29] and Richardson et al. [30], who focus on predicting click probabilities for new advertisements.

3. MOTIVATION

Keyword-based advertising relies on relevance of the displayed advertisements to the informational context in which they are presented, which explains its popularity: several studies have shown that ads which are relevant to goal-oriented users give higher user satisfaction and increased clickthrough rates [35]. Thus, the problem facing advertising systems is to identify all advertisements that are relevant to the current context, even if they were submitted for

Table 2: Phrase Similarity Evidence Sources and Similarity Functions

Evidence source	Similarity functions
Syntactic similarity	Variants of edit distance, cosine similarity [19, 11]
Search engine logs	Similarity based on clickthrough and session co-occurrence [22, 6]
Search engine results	Snippet-based similarity (web kernels) [31, 37]

keywords not found in it. Broad match addresses this need by assuming that advertisements submitted for keywords related to those extracted from context are relevant to the context by transitivity.

A number of similarity functions have been previously used for identifying related keywords; Table 2 summarizes the primary evidence sources that have been proposed in literature along with the corresponding similarity functions that can be of syntactic, geometric and statistical nature. While the various sources and functions provide diverse signals indicating keyword similarity, their suitability to the broad matching task may vary. While two keywords may be related syntactically or semantically, the distributions of advertisements that bid on them may differ considerably due to divergence in underlying commercial intent. Because broad match mapping has the specific goal of identifying keywords for which bidded *advertisements* are most suitable to the original context, our approach aims to obtain broad match mappings based on suitability of corresponding advertisements, which is best reflected by the corresponding clickthrough rates. Focusing on clickthrough is also essential because keyword advertising systems perform final advertisement ranking and pricing based on clickthrough estimates (e.g., by multiplying them with the associated bids to obtain expected monetization for a second-price auction).

Thus, the problem of identifying broad matches for a given keyword is equivalent to predicting the click probability for advertisements that bid on broad match keywords, when they are shown in the context of the original keyword. Formally, this task can be defined as follows. Let the advertising system have a vocabulary of bidded keywords, where for each keyword kw there is a corresponding set of broad-match advertisements with associated bids, $\mathcal{A}(kw) = \{\big(a(kw), bid(a, kw)\big)\}$. We assume that for every keyword kw, a pool of candidate broad matches $\mathcal{C}(kw) = \{kw'\}$ can be identified using off-the-shelf similarity functions such as those described above. The task is then to estimate the probability of click on a broad-match

$$p(c|kw \rightarrow kw') = \mathbb{E}_{a \in \mathcal{A}(kw')}[p(c|a, kw)] \quad (1)$$

for each $kw' \in \mathcal{C}(kw)$, where the expectation is computed over the ads with broad-match bids on kw' that are selected by the auction system when kw is found in context.

It is practically infeasible to express the expectation in Eq. (1) analytically for a real-world auction-based advertisement ranking system given the complexity of production ranking functions, which are typically based on second-price auctions, but also incorporate mechanisms to account for factors such as relevance, fraud, inventory balancing, etc. However, if broad match advertisements are continuously shown by the system, we can formulate the task as approximating $p(c|kw \rightarrow kw')$ for the *empirical* distribution of keywords, broad match mappings and advertisements. Then, for each keyword kw, top broad matches can be identified by ranking each candidate $kw' \in \mathcal{C}(kw)$ based on a utility function that combines predicted $p(c|kw \rightarrow kw')$ with various properties of $\mathcal{A}(kw')$ such as bid amounts, bid density, etc.

Finally, we note that considering monetary bid amounts is critical in ad ranking, and they can be easily incorporated into the broad-match selection problem by scaling the clickthrough predictions correspondingly.

4. APPROACH

4.1 Training Data

Our approach aims to leverage the abundance of user-produced behavioral data that is collected and stored in advertising system logs. Because users' click behavior yields the ultimate measure of relatedness between the bidded keywords in pay-per-click advertising, we use it as supervision for learning to identify broad match keyword substitutions. From advertising system logs, all past ad impressions that were based on broad match substitutions can be extracted, yielding a dataset of training instances that have the form $(f(kw \rightarrow kw'), c)$, where

- $f(kw \rightarrow kw')$ is a feature vector encoding various properties of the impression of an advertisement bid on kw' shown in context containing keyword kw, such as various similarity functions, individual properties for both keywords, features derived from the clickthrough history of the keywords, etc., as described in Section 5.1
- $c \in \{-1, 1\}$ is a binary variable encoding whether a click has occurred.

The data distribution for a dataset constructed from the logs in this manner is completely determined by the broad-match algorithms that produced the related keywords behind the served ad impressions. There are two key implications of this bias. First, learning algorithms that utilize this data should avoid making strong generative assumptions, which invites the use of discriminative approaches. Second, this setup encourages diversification of training data by employing A/B testing (also known as split or control/treatment tests, or parallel flights), where a number of alternative broad-match algorithms are used in parallel in randomized fashion [23]. Employing A/B testing with a diverse set of alternative broad-match algorithms produces a less biased distribution of training data, aiming to capture a broad set of candidate broad-match pairs for evaluation.

It is well-known that clickthrough rates are heavily influenced by the relative position in which an advertisement was displayed. Two approaches can be utilized to normalize for position bias: one is to only consider advertisements displayed in the top position [29], which is what we do in experiments described in Section 5. An alternative approach would be to utilize ad impressions from all positions, correcting for positional bias by discounting impressions at lower positions (see, e.g., [30]), which may be preferable in situations when the amount of available data is more limited (e.g., with a smaller-scale ad platform).

There are several benefits of employing log-based training data compared to human labeling. First, it avoids the costs of employing human judges, while providing data at a much larger scale. Second, it provides *in situ* measurement of the utility of broad-match suggestions: while certain keywords may be judged as related by human judges, the sets of ads bid on them may not be optimal (e.g., due to polysemy). Clickthrough data avoids this pitfall, directly reflecting on the utility of a broad-match substitution in the context of pay-per-click advertising.

4.2 Online Learning with Amnesia

Online advertising is an extremely fluid domain. The distributions of advertisers, advertisements, bidded keywords and contexts from which keywords are extracted (queries or web pages) change very rapidly for a number of reasons: advertisers constantly modify their bids, new advertisers enter the system, query distribution shifts over time, publishers' content changes in contextual advertising, etc. World events (holidays, news, new movies, etc.), seasonality and other factors also significantly affect clickthroughs for certain advertising terms, as well as change the corresponding bidded advertisement distribution.

Learning in this setting requires an algorithm that can effectively respond to the changing environment, modifying the learned hypothesis automatically to reflect the drift in underlying data and click distributions. Online learning algorithms which assume that training instances arrive in a continuous stream are most suited for this domain, since they allow deploying the system that continues to learn from clickthrough data without human intervention, incorporating drift in the behavior of users, advertisers and publishers.

We base our approach by adapting the max-margin voted perceptron algorithm [16] to incorporate moving average weighting, commonly used in time series applications. Max-margin voted perceptron is a well-known discriminative online linear classifier that has been shown to exhibit excellent performance on a number of high-dimensional learning tasks in domains that vary from natural language processing to computer vision. Using averaging instead of voting is a common modification to the algorithm that has been empirically shown not to impact results, while significantly simplifying the computation and removing the high memory cost [12].

While averaged perceptron is a robust, efficient classifier, it does not immediately address the issue of drift: its hypothesis is an average of all weight vectors observed in the past (although with sufficient data, subsequent weight vectors will reflect past distribution changes). We propose modifying the algorithm so that instead of a simple mean of all weight vectors, the hypothesis is a *multiplicatively re-weighted mean*. This effectively corresponds to averaging with an exponential time decay, where the weight vectors observed in the past are "forgotten", while the most recent weight vectors have the most influence on the hypothesis. Moving averages are frequently used in applications that involve prediction over time series data [4], and our approach incorporates the popular Exponentially Weighted Moving Average (EWMA).

The resulting algorithm, Amnesiac Averaged Perceptron, is shown in Figure 2. The algorithm processes training examples as a stream, updating a current hypothesis (weights $\mathbf{w}$) any time a training example is misclassified by it, with the update based on hinge loss. The optimal hypothesis, $\mathbf{w_{avg}}$, is maintained as the exponentially weighted moving average used for outputting actual predictions:

$$\mathbf{w}_{\mathbf{avg}}^{(n)} = \alpha \sum_{i=1..n} (1-\alpha)^{n-i}\mathbf{w}_i = (1-\alpha)\mathbf{w}_{\mathbf{avg}}^{(n-1)} + \alpha\mathbf{w}_i$$

where amnesia rate $\alpha, 0 < \alpha \leq 1$, dictates how much influence recent examples have on the averaged hypothesis compared to past examples (larger α correspond to more weight on recent examples). We note that notation used in Figure 2 assumes that each instance vector $\mathbf{x} = f(kw \rightarrow kw')$ includes a special attribute that always has value 1, which obviates the need for a separate bias term.

Because the algorithm produces uncalibrated predictions of click, we employ sigmoid calibration to convert them to actual probabilities, which has been shown to be an effective method for converting the output of max-margin classifiers to probabilities [27].

Algorithm: AMNESIAC AVERAGED PERCEPTRON
Input: Streaming training examples:
$\{(\mathbf{x}= f(kw \rightarrow kw'), c)\}$, $c \in \{-1, +1\}$
Amnesia rate $0 < \alpha \leq 1$
Output: Weight vector $\mathbf{w_{avg}}$
Algorithm:
Initialize
$\mathbf{w_{avg}} = \mathbf{w} = \mathbf{0}$
For each $(\mathbf{x} = f(kw \rightarrow kw'), c)$
$\hat{c} = \text{sign}(\mathbf{w} \cdot \mathbf{x})$
If $\hat{c} \neq c$
$\mathbf{w} = \mathbf{w} + c\mathbf{x}$
$\mathbf{w_{avg}} = (1-\alpha)\mathbf{w_{avg}} + \alpha\mathbf{w}$

Figure 2: Amnesiac Averaged Perceptron algorithm

4.3 Feature Selection

Given the large number of features and their redundancy, along with the high amount of noise inherent in the data set (a given keyword substitution will sometimes be clicked, but most often not), we expect feature selection to improve the performance of our method.

Feature selection has a long history within the field of machine learning [24]. We choose to use greedy forward selection based on a holdout set, a simple yet powerful technique. Greedy forward selection begins with a set of pre-selected features, S (which is typically initially empty). For each feature f_i not yet in S, a model is trained and evaluated using the feature set $S \cup f_i$. The feature that provides the largest performance gain is added to S, and the process is repeated until no single feature improves performance.

5. EXPERIMENTS AND DISCUSSION

We use the Amnesiac Averaged Perceptron algorithm described in Section 4.2 as the learning algorithm in our experiments to predict the clickthrough rate for ads based on broad-match keywords. The following subsections describe the features, the training data, and the results, as well as additional details about the contributions of each aspect of our approach: using amnesia, running in online mode, and employing feature selection.

5.1 Features

As discussed previously, there are many types of similarity functions between keywords. By including a wide variety of functions, we provide more diverse information to the learner which can be exploited for predicting the clickthrough of broad matches.

The first set of features is based on co-occurrence statistics in user query logs and advertiser bidding behavior [22, 28]. When users submit queries to a search engine, they tend to focus on a single topic within each small time window. We can thus exploit the co-occurrence of terms in queries that co-occur within a search session as a signal of relevance. Analogously, when advertisers specify the campaign for a given ad, they enter bids on multiple keywords that are all related to the topic of the ad, allowing the use of keyword co-occurrences as evidence of relevance between them. We counted user query co-occurrences only if they fell within a window of 10 minutes, which we found to qualitatively improve the similarity measure. For advertiser bid term co-occurrences, we ignored advertisers that had bid on more than 500 keywords, as generally such keywords were unrelated. These features are henceforth called "user-query" and "ad-kw", respectively.

The raw counts of co-occurrences, while calculating a similarity between terms, are not reliable because popular keywords (e.g., *myspace*) frequently co-occur with many unrelated keywords. To

address this issue, we measure the similarity using Pointwise Mutual Information (PMI) [9], defined as:

$$PMI(X,Y) = \log_2 \frac{P(X,Y)}{P(X)P(Y)}$$

In order to avoid spuriously high PMI for rare terms, we employ Dirichlet smoothing for the above probabilities. Only high-PMI term pairs are kept (the threshold was set by qualitatively observing the effect of the threshold on the quality of the terms being included). For both user and advertiser co-occurrences, we included both the PMI and non-PMI versions of the similarity measure in our feature set.

The second set of features is based on 17 pre-existing broad match mappings, which were generated by other Microsoft employees using a variety of techniques such as those described in Table 2, and based on textual features, clickthrough logs, monetization, etc. We will refer to these mappings as BM_i, where i is the index of broad match mapping. For each, we generate a binary feature which indicates whether $kw \rightarrow kw'$ exists in the mapping.

The similarity measure derived from users query logs, advertiser bids, and the pre-existing broad match mappings can be formulated as graphs, where each node is a keyword and the similarity between two keywords is given by the weight of the edge between them. We use the following technique to increase the density of the graph: for each node, we add two-step neighbors (neighbors of neighbors) and set the weight of the edge using the electrical-network (or inverse distance) conversion. This third set of features (called "densified"), partially exploits the local structure of graphs and, as will be seen, improved results in some cases.

The fourth set of features is based on syntactic similarity measures, which include string edit distance, the presence of one keyword as a substring inside the other, etc. We also include features that are a function of either the original or the broad-match keyword in isolation, such as the prior clickthrough rate (CTR) of either keyword calculated from both search and contextual advertising logs. We also add some other derivative features such as the log and log-odds values of the above prior CTRs, and a binary indicator feature encoding whether the prior CTR is larger for the original or substituted keyword. For each of the similarity functions described above, we also add a binary indicator features that takes value 1 if the similarity function value is non-zero, and 0 otherwise.

Finally, we add two additional feature sets, each of dimension equal to the total number of bidded keywords, for the original and broad-match keywords. These "keyword-id" features simply encode the identity of the two keywords in each pair; and are efficiently accommodated in the Amnesiac Averaged Perceptron when encoded sparsely.

5.2 Dataset

Our training and testing data was derived from two months of logs collected by the Microsoft contextual advertising system. As explained in Section 4.1, the training set was constructed by extracting ad impressions that were based on broad match substitutions from existing broad match algorithms used in the system. From each ad impression, an instance $(f(kw \rightarrow kw'), c)$ was created based on the original keyword kw, the substituted keyword kw' (from one of the mappings, BM_i), and an indication whether it was clicked or not, c. To remove positional bias of the ads, we only considered advertisements that were shown in the top position. We subsampled the training data, keeping less than 1% of the non-clicked impressions, to make the class distribution more balanced. Random subsampling does not hurt learning because Amnesiac Averaged Perceptron is a discriminative classifier. The resulting training set contains millions of advertising impressions based on millions of bidded keywords.

5.3 Results

The goal of the work is to accurately predict $p(c|kw \rightarrow kw')$, the probability that an ad shown for a substituted keyword, kw', will be clicked, given the original keyword, kw. Following conventional practice, our base metric for model evaluation is its log-loss over a test dataset $X = \{(kw \rightarrow kw', c)\}$:

$$LogL(X) = \sum_{(kw \rightarrow kw', c) \in X} \log_2 \left(p(c|kw \rightarrow kw')\right)$$

A higher log-loss indicates a more accurate model. Note, however, that the test set is very noisy: any given keyword pair, $kw \rightarrow kw'$, will correspond to many impressions with clicks, and to even more impressions without clicks. Thus, no batch-mode model can achieve a perfect log-loss score because the best it can do is predict the empirical $p(c|kw \rightarrow kw')$ on the test set, suffering non-zero loss for all impressions. In fact, this perfect batch-mode model will achieve a log-loss that is equal to the entropy of the test set distribution $H(c|kw \rightarrow kw')$, which we measured to be 0.5348. We thus report the difference between our model's log-likelihood and the entropy of the test set, which we term "log-likelihood lift" ($LogL\text{-}Lift$). LogL-Lift of zero means the model is equivalent to the perfect, oracle-like batch-mode learner. In the following sections, we present both log-loss (negative log-loss) and LogL-Lift for each model; for both metrics, lower is better. All log-loss and LogL-Lift differences presented in this paper are statistically significant ($p < 0.01$) with the exception of Figure 3, on which we provide error bars.

Training was conducted using the features described in Section 5.1. We used online learning, greedy feature selection, and amnesia, as described in Section 4. The following methodology was used for training: the two-month dataset was divided into one month for training and validation, and one month for testing. The first month was further subdivided in half into a training set (the first 15 days) and a validation set (the last 15 days). Feature selection was performed by training on the training set, and testing on the validation set. Amnesia parameter (α) setting was performed by trying 12 values, ranging from 10^{-6} to 10^{-1}(see Figure 4), training a model on the training set (using the features selected without amnesia), and testing it on the validation set. The best value of α was then used for a new iteration of feature selection that included amnesia as part of its training process (this was found to improve results)[1].

We chose online vs. batch feature selection depending on whether we were training online or batch. Similarly, we ran feature selection using amnesia depending on whether we were training with amnesia or not. For example, if we were training online and with amnesia then feature selection was also done using amnesia and in online mode. This process of feature selection improved performance. Once the parameter and feature selection were complete, the model was trained on the entire first month, and tested on the entire second month (note that the feature selection and parameter tuning were performed only using the data from the first month). Table 3 presents these results.

The row labeled 'Prior' represents a model that simply predicts the same probability of click for all instances in the test set, determined by the average probability of click on the training set. The second row gives the performance of our complete model, which was trained online, and includes feature selection and amnesia. In

[1]We also tried re-tuning α, given these new selected features, but found that in all cases, it remained unchanged.

Table 3: Comparison of Log-Loss and LogL-Lift with Prior (lower score is better). All tabular log-loss and LogL-Lift differences presented in this paper are statistically significant.

Model	*-LogLoss*	*LogL-Lift*
Prior	0.6572	0.1224
Feature Selection (FS) + Online Learning (O) + Amnesia (A)	0.5709	0.0361

the remaining tables, we will abbreviate these as O, FS, and A, respectively. The complete model results in a significant improvement in accuracy, approaching the performance of the hypothetical perfect batch learner, eliminating 71% of the loss between the prior and this (possibly impossible) goal.

In the following subsections, we investigate the contribution of each of the major portions of our model to the overall performance: feature selection, online training, and amnesia.

5.4 Feature Selection

In this section, we investigate the value of performing feature selection. In Table 4, we give the same full model results as before, but with an additional line indicating the performance of the Amnesiac Averaged Perceptron without feature selection ("O+A, All Features").

Table 4: Log-Loss and LogL-Lift and Feature Selection

Model	*-LogLoss*	*LogL-Lift*
O+A, All Features	0.6563	0.1215
O+A, Keyword-id Features	0.5953	0.0605
O+A, Feature Selection	0.5709	0.0361

As can be seen, feature selection provides a dramatic improvement over the standard algorithm that uses all features. Table 5 shows the features selected for the four possible scenarios of (online vs. batch) and (amnesia vs. no amnesia). Interestingly, only a few of the features were selected from the original 68 base (non-keyword-id) features. One of the reasons might be because many features are redundant, particularly the derived features such as the binarized similarity measures (see Section 5.1 for details).

The selected features for different settings contain a mix of feature types, covering historically observed clickthrough rates, textual matching, other broad match mappings and both the user- and advertiser-based co-occurrence similarity measure. Interestingly, the actual PMI-valued features based on co-occurrence graphs were not considered useful, while features discretizing them via thresholds were chosen. Notably, the densified versions of the graphs generated from user query logs and advertiser bids logs were chosen over the non-dense versions, demonstrating the value of the two-step walk densification procedure.

The keyword-id features were selected for all four model settings (Table 5). They perform well in all configurations, but particularly so in the online setting, as they capture the empirical clickthrough rates accumulated continuously, reflecting changes in them due to various drift factors. In fact, a model using only the keyword-id features achieved a log-loss of 0.5953, which represents 72% of the overall improvement from the prior.

We note that if online-mode feature selection is computationally too expensive for a given system, it can be performed periodically offline without a significant loss in performance.

5.5 Online vs. Batch

As discussed above, the task of predicting which ad replacement pairs are most likely to be clicked is inherently an online task. In this subsection, we investigate the contribution that online learning made to the overall performance of the learner.

Table 6: Log-Loss and LogL-Lift and Online Updates

Model	*-LogLoss*	*LogL-Lift*
A+FS, Batch	0.6110	0.0762
A+FS, Weekly Batch	0.5948	0.0600
A+FS, Online	0.5709	0.0361

As can be seen from Table 6, online learning contributes significantly to the overall performance of the learner. Recall that when testing batch-mode model performance, feature selection is done by training a model on the training set and testing it in batch mode on the validation set. Likewise, for online feature selection, testing on the validation set is done online. In both cases, we find that matching the feature selection mode to the mode in which the learner would be used leads to increased performance (e.g., online-trained feature selection performs worse than batch-mode-trained feature selection in batch mode training).

Online learning significantly improves the log-loss of the model. One natural question is, what if we re-train the model periodically, rather than keep it updated in an online fashion. In Table 6, we give results for performing such retraining on a weekly basis[2]. Specifically, the model is trained on the training set, and tested on the first week of the test set. Then, it is trained on the training set plus the first week of the test set and tested on the second week of the test set, and so on. As with online learning, the performance is measured as the overall performance on the entire test set. As can be seen from the table, weekly re-training does improve model accuracy, but online training (which retrains after every individual example) is still significantly better.

To further explore the issue of online vs. batch mode training, we plot learning curves (see Figure 3) for batch and online modes (top and bottom curves, respectively). The x-axis indicates the relative data set size, obtained by randomly sampling data instances according to the corresponding probability. The first item to note is that the performance of the batch model asymptotes at around 80%, indicating that the improved performance of the online learner comes from its ability to track a changing distribution, rather than from needing additional data. The results of experimenting with the amnesia rates (presented in next section) also confirm that this is the case by demonstrating that intentionally forgetting distant training examples significantly improves performance.

The learning curves also show that the performance of the online model with only 10% of the data still out-performs that of the batch model. In a live system, it may not be possible to train the model on every ad impression, but as can be seen, even if the training occurs on only one of every ten impressions, the model still significantly out-performs the batch learner. Also, the performance of the online model is continuing to improve at 100% of the training set available to us, indicating the value of using all available data. One advantage of the Amnesiac Averaged Perceptron algorithm is that it can be implemented very efficiently, which allows taking advantage of all incoming data. Our implementation was able to process over 5000 examples per second on a single CPU core on a commodity PC workstation.

Finally, we also plot a curve indicating the performance of the online learner if it is only given a portion of the initial training set,

[2]We tried both batch and online feature selection and report the better of the two.

Table 5: List of selected features using Feature Selection. See section 5.1 for a description of these features.

Online and Amnesia	*Keyword-id*, $log_2(Contextual\ CTR_{kw})$, BM_1, BM_{17}
Online and No Amnesia	*Keyword-id*, $Contextual\ CTR_{kw'}$, $Contextual\ CTR_{kw'} > Contextual\ CTR_{kw}$, BM_{12}, *Binary-Users-query using PMI*, BM_{14}, BM_4
Batch and Amnesia	*Keyword-id*, $log_2(Paid\ CTR_{kw'})$, $log\text{-}odd_2(Paid\ CTR_{kw})$, *Substring*, $Paid\ CTR_{kw'}$
Batch and No Amnesia	*Keyword-id*, $log_2(Paid\ CTR_{kw'})$, $log_2(Paid\ CTR_{kw})$, *Substring*, *Binary-Densified-User-query using PMI*, *Binary-Densified-Ad-kw using PMI*

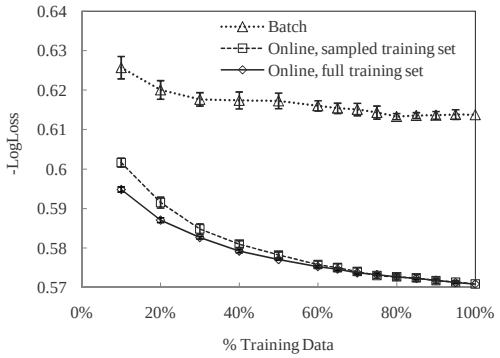

Figure 3: Effect of training data availability on performance

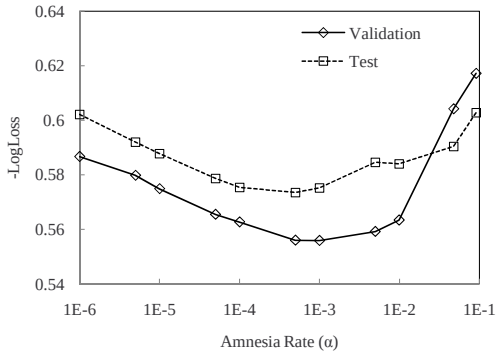

Figure 4: Effect of amnesia parameter on Log-Loss, using online training.

instead of the full training set (middle curve). At approximately 50% of the data, the performance of this curve joins that of the online learner that was initialized with the full training set, indicating that with sufficient online data, the learner does not need much initial training.

5.6 Amnesia

In this subsection, we investigate the value of adding amnesia to the standard online averaged perceptron algorithm. As described in Section 4.2, amnesia is a technique for the model to smoothly forget old training examples as novel examples arrive. This allows it to remain focused on the current instance distribution, which may have changed dramatically from the original one. In Table 7, we give the results with and without amnesia.

Table 7: Log-loss and LogL-Lift with and without amnesia, both for batch and online Mode

Model	*-LogLoss*	*LogL-Lift*
O+FS	0.6033	0.0685
O+FS, with Amnesia	0.5709	0.0361
FS	0.6204	0.0856
FS, with Amnesia	0.6110	0.0762

As can be seen, amnesia is contributes significantly to the model performance. Given the time-varying nature of our task, we expected this, but the fact that adding amnesia nearly halves the LogL-Lift of our model is surprising. This shows that there is significant change occurring in the data stream, and also suggests the potential benefits from using amnesia in other clickthrough prediction tasks, such as modeling user interaction with search engines.

The amnesia parameter, α, was set using the validation set. As can be seen in Figure 4, the minimum log-loss on the validation set (in online mode) is achieved at $\alpha = 5 \times 10^{-4}$ for the online setting. In batch mode, the optimal setting is $\alpha = 5 \times 10^{-5}$. In both cases, the minimum log-loss on the test set is found at the same setting as on the validation set. It is interesting to note that the amnesia parameter is larger for online mode, which makes intuitive sense: as the online learner is continuously refreshed with new examples, a larger amnesia setting allows it to react more rapidly to any changes in the distribution.

5.7 Offline CTR Estimates On Real-World Data

While log-loss provides a good information-theoretic measure of the quality of our model, it is difficult to interpret from a practical standpoint. To construct a more concrete evaluation measure, we attempt to estimate the real-world performance of our approach through the following process: (1) Use the learned model to build a new broad match mapping (BM mapping), (2) Estimate the clickthrough rate (CTR) of ads based on the new mapping. The new broad match mapping is built by selecting, for each source keyword, the set of k broad-match keywords with highest predicted clickthrough. These are chosen from all observed broad match replacements in the training data.

Once the top-k replacements are selected for each term, the mapping is evaluated by estimating the CTR that would be obtained if it were used on real user traffic. This is done by measuring the CTR of the subset of the test set that overlaps with the mapping:

$$CTR(BM) = p(c|(kw \rightarrow kw') \in BM)$$

For each test-set instance, if the substituted keyword, kw' is one of the k replacements given by the broad match mapping for the original keyword, kw, it is considered in the CTR computation, otherwise it is ignored. We also report the coverage of the mapping:

$$Coverage(BM) = p((kw \rightarrow kw') \in BM)$$

The purpose of the coverage measure is to verify that a given algorithm is not just providing replacements for a few "cherry-picked" high-CTR keywords. It does not reflect what would happen if the mapping were used on live traffic, because at that point it would have 100% coverage; rather, we use it as a diagnostic metric to ensure that the estimated CTR is based on a significant subset of the test set.

Because our metrics use the test set to evaluate the quality of the broad match mapping, we must restrict ourselves to only using the batch-mode model (the online model uses information from the test set to continue training, which would constitute having knowledge of the future when the mapping was evaluated). Since the online model performs significantly better, we would expect an even higher CTR for it.

To compare to a pre-existing broad match mapping, we follow the same process, but with a model that is trained using only one feature: whether or not the keyword pair exists in that mapping. This ensures a fair comparison between the learned model and the previous broad match mappings by ensuring that (1) they have access to the same set of pairs with which to construct a mapping, and (2) they are both forced to build replacement lists of size k for each original keyword, which helps to ensure they have comparable coverage on the test set.

In Table 8, we give the results of this experiment, for our model (in batch mode) and the three pre-existing broad match mappings that have performed best on live traffic (BM_1, BM_2, and BM_6). In the table, we give the relative CTR as compared to BM_6, the best performing broad match mapping. As can be seen, our model (FS+A) performs significantly better than all of the previous best mappings. The coverage results verify that all three models are being fairly compared, as all cover a sizable fraction of the test data.

Our method could have chosen to use any subset of the 17 different input broad match mappings (BM_1, BM_2, ...) as inputs (though the feature selection often chose only a couple). Thus, we verified that the FS+A model has a higher estimated CTR than all 17 BM mappings. Given these results, we can be reasonably confident that our model will significantly boost CTR if used in a live system with real users.

Table 8: Relative Estimated CTR (relative to BM_6) and Coverage for batch-trained model (FS+A) and the three broad match mappings that had highest real-world performance (see Section 5.8). The FS+A model achieves a higher estimated CTR than all 17 broad match mappings, BM_1-BM_{17} (not shown).

Model	*Relative CTR*	*Coverage*
BM_1	80.0%	59.5%
BM_2	76.7%	59.2%
BM_6	(100.0%)	30.2%
FS+A	117.6%	41.4 %

5.8 Live Test Results

There are limits to the accuracy of offline scoring. In particular, some of the best replacements suggested by our algorithm may never be seen because they are not part of the observed data. This could occur, for example, if a particular keyword replacement has ads that are not very valuable. Though our offline tests suggest the algorithm will perform well, in the end, running it in an actual system serving live traffic is the most trustworthy evaluation. Thus, we employed the method from Section 4 to generate a broad match mapping, setting k to the same number of replacements as were used in the existing broad match mappings. Furthermore, we incorporated monetization into our mapping by adding a preference for terms that had historically been worth more.

In Figure 5, we show the CTR performance and revenue of our algorithm (AAP) as compared to the three other broad match mappings that had highest revenue and CTR. These mappings were run in parallel, meaning each impression entering the system was randomly assigned to one of many broad match mappings (ours and the three best performers are just four of many). Because the broad match mappings are running in parallel, we have a controlled experiment that allows us to directly compare the CTR and revenue results for each mapping. The CTRs and revenues are scaled so that the highest-CTR mapping performance is given as 100%.

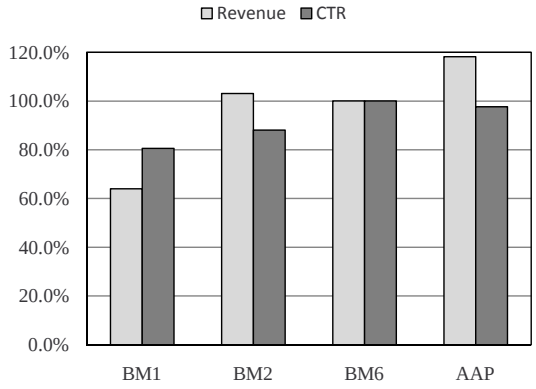

Figure 5: Relative revenue and CTRs for our model (AAP) vs. the three other best-performing broad match mappings

As can be seen, our model increases revenues by approximately 15% over the next-best mapping (BM_2), and 18% over the mapping that has highest CTR. In terms of CTR, our model has a slight (2%) decrease in CTR vs. the best model, but still shows an 11% gain over the model that had the next-highest revenue. By having a good model for clickthrough of substitution terms, our model was able to identify terms that were most likely to lead to higher clickthrough. We were then able to trade-off that improved clickthrough for improved revenue by selecting terms that had a slightly lower clickthrough, but provided more revenue. We hypothesize this is the reason for our model significantly increasing revenues with a negligible effect on clickthrough rates.

We note that live traffic tests were conducted using a preliminary version of our algorithm that did not yet incorporate amnesia or online learning. In terms of log-loss, the complete model performed significantly better than the one that we tested, so we expect the real-world performance of the complete model to show even more significant gains in revenue and possibly gains in CTR.

6. FUTURE WORK

While in this work we only employed similarity functions based on pairwise similarities between keywords, the highly relational nature of data in the keyword advertising domain invites approaches that can exploit higher-order structures for obtaining more accurate similarity estimates. For example, an interesting avenue for future research is investigating the utility of graph-based similarity functions. One such function has been proposed by Fouss et al. [15], which estimates similarity based on a random-walk computation. Another possible approach is to incorporate the work of Zhou et al. [38] who proposed a multiple-graph framework for document recommendation.

The very high dimensionality and sparsity of keyword advertising data makes smoothing an important component for probabilistic similarity functions such as PMI. Improving smoothing using advanced approaches such as that of Mei et al. [25] may produce improved similarity estimates, leading to better overall performance.

Our dataset is obtained from a diverse set of broad match algorithms running on a large ad network. It is interesting and chal-

lenging to explore the algorithms to be used on small networks to sample the data adaptively.

The excellent empirical performance of Amnesiac Averaged Perceptron invites developing principled theoretical understanding of its effectiveness. The recent development of *budgeted* perceptron variants that allow learning-theoretic analysis, e.g., Forgetron [14], lends hope that formal analysis of Amnesiac Averaged Perceptron will provide a fruitful research direction.

7. CONCLUSIONS

In this paper, we propose an approach to identifying broad match keywords via learning to estimate corresponding clickthrough probabilities. The approach utilizes supervision in the form of implicit feedback obtained from logs of past ad impressions based on broad matches, obviating the need for expensive human labeling. We introduce an efficient online algorithm, Amnesiac Averaged Perceptron, which rapidly adapts to changes in the underlying data distribution by multiplicatively down-weighting past weights when averaging the hypotheses. Experimental evaluation demonstrated the effectiveness of our approach both in offline log-based experiments as well as in live tests conducted on contextual advertising traffic.

8. REFERENCES

[1] IAB Internet Advertising Revenue Report conducted by PricewaterhouseCoopers (PWC), November 2008. [online] `http://www.iab.net/insights_research/530422/1357`.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of SIGIR-06*, pages 19–26, 2006.

[3] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of KDD-03*, pages 39–48, 2003.

[4] G. Box, G. M. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley, 2008.

[5] R. C. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL-06*, 2006.

[6] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*, 2008.

[7] B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *Proceedings of NIPS-07*, 2007.

[8] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *Proceedings of WWW-08*, 2008.

[9] K. Church and P. Hanks. Word association norms, mutual information and lexicography. In *Proceedings of ACL-89*, pages 76–83, 1989.

[10] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *Proceedings of WWW-08*, pages 227–236, 2008.

[11] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 73–78, 2003.

[12] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-02*, pages 1–8, 2002.

[13] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP-04*, pages 293–300, 2004.

[14] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008.

[15] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.*, 19(3):355–369, 2007.

[16] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In *Machine Learning*, 1999.

[17] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *Proceedings of WWW-08*, pages 61–70, 2008.

[18] Q. Guo and E. Agichtein. Exploring mouse movements for inferring query intent. In *Proceedings of SIGIR-08*, pages 707–708, 2008.

[19] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, New York, 1997.

[20] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of SIGIR-05*, pages 154–161, 2005.

[21] T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. *IEEE Computer*, 40(8):34–40, 2007.

[22] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of WWW-06*, pages 387–396, 2006.

[23] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of SIGKDD-07*, pages 959–967, 2007.

[24] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[25] Q. Mei, D. Zhang, and C. Zhai. A general optimization framework for smoothing language models on graph structures. In *Proceedings of SIGIR-08*, pages 611–618, 2008.

[26] D. Metzler, S. T. Dumais, and C. Meek. Similarity measures for short segments of text. In *Proceedings of ECIR-07*, 2007.

[27] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 185–208. MIT Press, 1999.

[28] F. Radlinski, A. Z. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, and L. Riedel. Optimizing relevance and revenue in ad search: a query substitution approach. In *Proceedings of SIGIR-08*, pages 403–410, 2008.

[29] M. Regelson and D. Fain. Predicting click-through rate using keyword clusters. In *Proceedings of the Second Workshop on Sponsored Search Auctions*, 2006.

[30] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of WWW-07*, pages 521–530, 2007.

[31] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of WWW-06*, pages 377–386, 2006.

[32] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of KDD-02*, pages 269–278, 2002.

[33] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of KDD-08*, pages 614–622, 2008.

[34] E. M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of SIGIR-98*, pages 315–323, 1998.

[35] C. Wang, P. Zhang, and R. Choi. Understanding consumers attitude toward advertising. In *Eighth Americas Conference on Information Systems*, pages 1143–1148, 2002.

[36] R. White, I. Ruthven, and J. M. Jose. The use of implicit evidence for relevance feedback in web retrieval. In *Proceedings of ECIR-02*, pages 93–109, 2002.

[37] W. T. Yih and C. Meek. Improving similarity measures for short segments of text. In *Proceedings of AAAI-07*, 2007.

[38] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles. Learning multiple graphs for document recommendations. In *Proceedings of WWW-08*, pages 141–150, 2008.