Click-Through Prediction for News Queries

Arnd Christian König
Microsoft Research
One Microsoft Way
Redmond, WA 98052
chrisko@microsoft.com

Michael Gamon
Microsoft Research
One Microsoft Way
Redmond, WA 98052
mgamon@microsoft.com

Qiang Wu
Microsoft Research
One Microsoft Way
Redmond, WA 98052
qiangwu@microsoft.com

ABSTRACT

A growing trend in commercial search engines is the display of specialized content such as news, products, etc. interleaved with web search results. Ideally, this content should be displayed only when it is highly relevant to the search query, as it competes for space with "regular" results and advertisements. One measure of the relevance to the search query is the click-through rate the specialized content achieves when displayed; hence, if we can predict this click-through rate accurately, we can use this as the basis for selecting when to show specialized content.

In this paper, we consider the problem of estimating the click-through rate for dedicated news search results. For queries for which news results have been displayed repeatedly before, the click-through rate can be tracked online; however, the key challenge for which previously unseen queries to display news results remains. In this paper we propose a supervised model that offers accurate prediction of news click-through rates and satisfies the requirement of adapting quickly to emerging news events.

Categories and Subject Descriptors

H.3.1 [**Information Search and Retrieval**]: Indexing Methods; H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Measurement, Performance, Experimentation

1. INTRODUCTION

Searching for online news is one of the most important user activities in the context of the Web. As a consequence, there exist a number of dedicated news search engines and many of the major search portals offer a dedicated news search tab. Moreover, some search engines (such as *Google* or *Live.com*) have started to mix dedicated news search results with the results displayed in the regular search pane (i.e., when the user has not selected the *news* tab). This interleaving of different search results introduces challenges: because news and the "regular" search results compete for the available space, the display of news results should ideally be triggered only for queries with news-intent (i.e., queries for which a significant fraction of the people issuing them are actually looking for related news), but not for other queries, even if they contain keywords frequent in the news corpus. Hence, we require techniques that determine for which queries to surface news results.

The approach we use is to utilize the observed click-through rate (which we will refer to as CTR from here onward) on the news content as measure of whether news content should be displayed. If we can estimate this CTR for an incoming query, we can use this to trigger the display of news. While it is relatively easy to estimate the CTR for queries for which news results have been displayed a number of times already, the main challenge of estimating the CTR for the previously unseen queries remains. In this paper, we address this problem.

1.1 Characteristics of News CTR Prediction

In order to motivate our approach, we will first illustrate some properties of CTR prediction for news. First, it is important to note that this is not simply a task of statically classifying queries into *news* and *non-news* categories. Many queries are inherently ambiguous – for example, earlier in 2008 the search query "*Georgia*" was typcially either a request about information on the country (or the US state), or a request on news results regarding the conflicts on the Georgian border, with the relative frequency of these two query intents changing (sometimes rapidly) over time. This example illustrates the strong dependence between the time a query is issued and the likelihood of it resulting in a click on news results – any technique that cannot quickly adapt to events becoming "news-worthy" is of little value.

Second, we argue that the CTR we observe for a given query (assuming that somewhat relevant news articles exist in our collection) is not only a function of the relevance score of the displayed news article, but also strongly depends on additional factors such as the current "buzz" around a story, the age of the article itself, etc. For example, initial experiments on the correlation between CTR and the *BM25* [12] score of an article showed little correlation; moreover, different articles surfaced for an identical news queries on adjacent days often exhibit very similar CTR, even when the relevance scores of the articles themselves differ significantly. Our approach accounts for this by estimating the CTR at the level of a query (as opposed to a query-document pair), while using the relevance score of the highest-ranking document(s) as an input feature to this estimation. It is important to note that this does not mean that the quality of the returned documents is irrelevant: having a relevant result document is still a pre-requisite to surfacing any news content.

1.2 Applicability of Existing Approaches

In this section we discuss the applicability of several existing approaches to the problem of news CTR prediction.

Web Document Ranking: Techniques used for document ranking in the web context cannot directly be applied to our scenario for a number of reasons. For one, as we illustrated above in the case of BM25, the predicted click-through rate is not a function of the score of the highest-rated document for a given query, ruling out these

SIGIR'09 July 19–23, 2009, Boston, Massachusetts, USA.

classes of ranking functions for our purposes. Also, techniques that leverage explicit relevance judgements (e.g., between a query and a given news article), which are obtained either directly (e.g., [4]) or implicitly via user-behavior (e.g., [2, 20]) are problematic in this scenario, because the underlying articles change so quickly. Moreover, as pointed out in [8], newly posted news articles initially have very few links pointing to them, so techniques that use the linkage between pages (to, e.g., compute *PageRank* of an article) have very limited benefit in the news domain.

Query Classification: Techniques commonly used in query intent classification cannot directly be re-used either. For one, classifiers that use the words appearing in the query's text as main features (which have been shown to perform well for other query intent classification tasks, e.g. [15]) are problematic given that news changes so rapidly. As illustrated in the case of the query '*Georgia*' above, the intent of a set of keywords can change quite rapidly, which in turn would require constant re-training of the underlying classifier.

Moreover, the acquisition of training data itself would be difficult – humans may be good judges of whether or not a query is related to news, but they can not estimate a query's CTR well. To illustrate this, we use query logs and click-through data obtained from *Live.com* for the time interval from November 13-15 2008. Consider the queries '*Caylee Anthony*', '*voter registration*' and '*oil price*', all of which are closely related to events receiving significant news-coverage during the days the logs were generated, meaning that a reasonable human annotator might label them as "*news-intent*". However, the true click-through rates seen in the log files with respect to news results differ significantly between queries: while the query '*Caylee Anthony*' results in clicks on news articles at rates between 63%-69% (depending on the individual day), the click-trough rate for the query '*oil price*' lies between 22%-29% and for '*voter registration*' between 1.6%-5%.

To illustrate the variations in CTR for different news queries, we have plotted the rates observed for a sample of queries for which news-results were surfaced on November 7th, 2008 in Figure 1. Here, the x-axis corresponds to the number of times news results were displayed for the query in question (in logarithmic scale), and the y-axis to the observed CTR. The circles in the plot show the overall distribution, with a number of them labeled with the corresponding query text. Note that not only do the click-through rates for different news queries vary significantly, but also do some queries (such as "*facebook*") which – judging from the query terms – do not appear to have news intent actually receive some clicks on news results.

Online Models: Tracking CTR in an online manner via time series models is a well-studied area (e.g. [25, 1, 9]); however, applying any of these methods in our problem context would require collecting initial click-through data for queries by displaying news results for them, which – given that the vast majority of web queries do not have news-intent – would mean large numbers of irrelevant search results being displayed. Instead, our aim is to only display news results for a query when we have high confidence that this will lead to clicks on the news results.

1.3 Our Approach

The characteristics of the problem scenario described above mean that any solution has to satisfy a number of properties:

1. Fast adaptation to changing news events.
2. Prediction of actual CTR-values (as opposed to binary *news/no-news* classification).
3. Prediction of CTR for queries for which no news results have been displayed in the past.

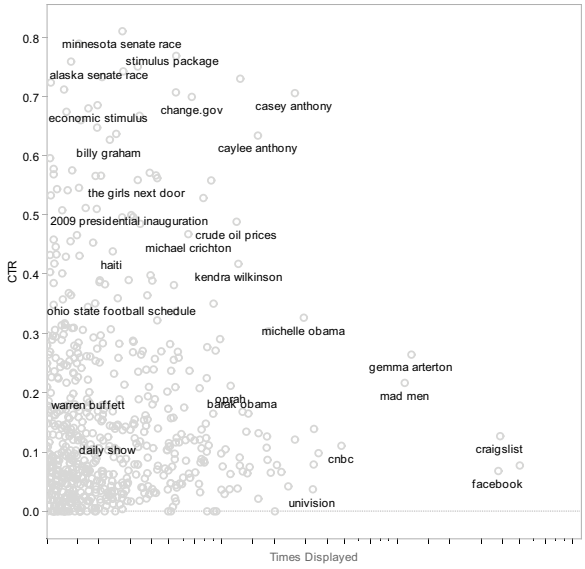

Figure 1: CTR of different news queries

4. Because the CTR has to be predicted online, as queries arrive at the search engine, the generation of features has to be fast.

Our approach adopts a supervised learning framework to address these challenges. To generate features for the learning algorithm, we leverage the following observation: keywords that are associated with news events are typically reflected in current corpora of news and blog articles. For example, when a news event occurs, keywords describing it are frequently found in titles of news articles published around that time. The same holds – to a lesser degree – for the word combinations in the initial paragraphs of news or their full text. Moreover, the change in the frequency of keywords in these corpora over time in itself is of interest; as we will show later, news events are often associated with spikes in keyword frequency. Finally, the context these words occur in within these corpora can itself give us hints as to whether a query characterizes a very specific news event or only contains keywords that happen to occur across many different news stories.

Hence, we will generate the features we use to predict click-through rates by analyzing the frequency and location of keywords in large corpora of news articles, blog postings, etc. By keeping these corpora current, we are able to adopt to breaking news events and changing levels of interest in older ones. For this purpose, we leverage dedicated news and blog crawling engines [18]. The corresponding corpora are ordered by time (using the date the individual articles were posted or crawled), allowing us to detect temporal trends in the frequency with which query terms occur. Because we keep only a small window of historic news and blog articles (7 days in our experiments), the underlying text corpora and required index structures are small enough to fit into the main memory of a single server, allowing for fast in-memory feature extraction as queries are processed.

The remainder of the paper is organized as follows: we first describe related work in Section 2. We then discuss the specific features, their extraction and the required overall architecture in Section 3. In Section 4 we then describe the model and training algorithm we use. We evaluate our approach in Section 5 and then conclude and describe next steps in Section 6.

2. RELATED WORK

The prior work closest to our approach is [9], which studies the integration of news and web search results by means of click-through prediction. Unlike our approach, [9] proposes a model that leverages both online click feedback as well as "contextual" features quantifying the frequency with which query terms occur in query logs as well as the news corpus over time. Our approach differs in both the features and the machine-learning framework proposed and uses additional corpora (such as a blog crawl and Wikipedia) to infer information about "buzz" around specific news items and background frequencies of query terms.

An area related to our problem scenario is *distributed information retrieval* [5], where a search query is distributed among a number of independent text databases. The retrieval of news content can be thought of as a distributed IR task with only two databases – the web and the news corpus. However, our scenario differs in a number of aspects: for one, as discussed earlier, the CTR we seek to predict is not solely a function of the relevance of the news content, which is the function optimized by current distributed retrieval systems. Moreover, the web and news retrieval systems used in practice are themselves very different in size, architecture and features used for ranking, making it difficult to adopt distributed IR approaches in our scenario.

There has also been considerable work in the area of *event/topic detection and tracking* (TDT) (e.g., [3]); event detection systems identify stories that discuss an event that has not been seen earlier and track occurrences of the events found across stories. While TDT cannot be used for CTR prediction directly, it may be possible to leverage their output for features in CTR predictors.

CTR prediction is an important area of research in computational advertising, in both the areas of *sponsored search* as well as *contextual search*. In sponsored search (e.g. [21, 22]), where CTR is commonly modeled on the level of individual advertisements (unlike our approach, where the estimation is tied to the query), a significant challenge is the CTR estimation and resulting ranking of ads which have been shown only few times or not at all. The authors of [22] adopt a supervised framework to address this challenge; both the proposed learning algorithm as well as the features are very different from our approach.

In contextual search (e.g., [7]) the input to the CTR prediction is the combination of the page an ad is to be displayed on and the ad itself (i.e., there is no notion of query); the prediction is typically made on the basis of word-overlap between the advertisement (i.e., the bid phrases) and the page itself (in addition to supplemental features). The approach taken in [7] is similar to our work in the sense that both ads and pages are divided into different regions (title, body text, abstract, etc.). Unlike [7], however, nearly all of the features we propose are aggregates in the sense that they encode the properties of words/phrases relative to a large corpus of text as opposed to individual documents. Also, as before, the learning algorithm we adopt is very different.

3. FEATURES

In order to determine whether a query is likely to result in clicks on news-results, we designed a number of features to produce evidence for or against the news-relatedness of a query. We differentiate three classes of features. First, a small set of features is computed directly from the query text, without consulting any of the blog or news corpora; we refer to these as *query-only features*. All of the remaining features consult various background corpora to find distributional information about the query terms. We further differentiate between *query-context features* that are based on the textual context the query keywords occur in and *corpus frequency features*, which track the number of times query terms occur in the underlying corpora over time. Finally, we also use the BM25 score for the top-scoring document as a feature in our classifier.

One important consideration for features is the extraction overhead they require. Because the features need to be computed online, as queries are submitted, we restrict our feature set to features we can compute online without resulting in unacceptable latency.

3.1 Text Corpora

When trying to determine the intent of a query, and specifically whether it is about current news and events, it is crucial to have a notion of what the current events are. To capture changes in query-intent resulting from recent news events, we use different background corpora to capture the distributions of news events that are drawing attention at any point in time.

The first corpus we use is a corpus of news articles, extending from the time the query was issued to several days in the past. As a second corpus we use a large set of blog posts crawled over the same period of time. User-generated content in blogs has skyrocketed and continues to grow at a fast pace [23, 24]. By nature, blogs tend to be about current trends and events and provide information about the current "buzz" around topics and events, giving us a better handle on which subset of articles are drawing significant attention. Finally, we use Wikipedia as a "background corpus" that contains relevant information, but is not geared towards current events. This corpus can provide additional evidence when compared to blog and news data: if query terms are salient in Wikipedia, but not salient in recent news and blog posts, the query is more likely to be a general information query than a news-related query.

The news and blog corpora are acquired using dedicated news and blog crawling code; in case of blogs, the crawler uses the ping/feed mechanism of blogs to identify new blog posts in a timely manner, combined with additional parsing and crawling of the permalink found in the feed to deal with partial feeds. A detailed description of the blog post acquisition architecture is given in [18].

3.1.1 The Value of Multiple Corpora

Current events are constantly in flux, and temporal changes in the frequency of terms can indicate the emergence of a news story. We found in our experiments that news and blog corpora complement each other in that some emerging news stories will result in "keyword spikes" in one corpus, but not the other, depending on a given topic, which makes blog posts an interesting and complementary source of information.

To give some examples, consider the charts in Figures 2-4. Figure 2 shows the distribution of the word *Galveston* across news articles, news titles and blog posts. As hurricane Ike approached the Texas coast, there was a strong spike in the frequency of the word *Galveston* (an area of the Texas cost hit by the hurricane) across all three sources. The word *Caylee* (*Caylee Anthony* was a missing child in the news at the time) shows a very pronounced spike in blog posts on Sep 11, while news articles and titles during the same time do not exhibit much fluctuation (Figure 3). Upon closer inspection of the data, a likely explanation is that the news at the time was dominated by hurricane stories, and there were relatively few stories about Casey Anthony's (the child's mother) impending re-arrest. Given the popular interest in the story, however, blog commentary about this piece of news spiked sharply. A reverse example, where there is a strong signal in news but not in blogs can be seen in the distribution of the word *Pakistan* in Figure 4, where an event generated significant coverage in news media, but not so much in blogs.

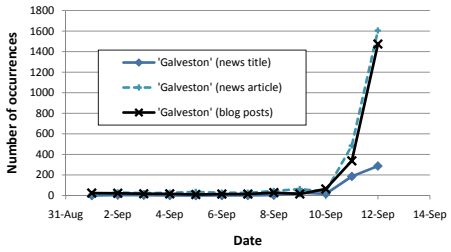

Figure 2: Occurrences of the term "*Galveston*" in different corpora

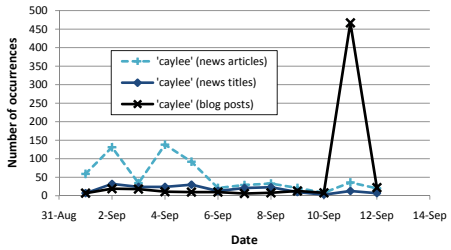

Figure 3: Occurrences of the term "*Caylee*" in different corpora

3.2 Corpus Frequency Features

The first set of features we use quantifies the number of documents that match the query using (a) set containment semantics and (b) phrase semantics in the news and blog corpus (i.e., how many documents in the corpora contain all the words in the query and how many contain the query terms adjacent and in the same order as in the query): for each corpus, we use both of these counts as well as their difference as features. For the news data, we further subdivide each article into the article title and the full text of the article; for each of these regions we collect the counts separately, and use them as distinct features. Finally, each of these counts is broken down by date – we collect separate counts for each date, going back 7 days into the past.

In order to assess the "salience" of query terms in the three corpora, we use simple $tf.idf$-based [17] metrics. The $tf.idf$ value of a token is calculated over the whole corpus as

$$td.idf(token) = \begin{cases} 1 + \log(tf(token)) \frac{N}{df(token)} & : tf(token) \geq 1 \\ 0 & : tf(token) = 0 \end{cases}$$

where $tf(token)$ is the count of the token in the collection, $df(token)$ is the count of documents containing the token, and N the total number of documents. For news articles, we calculate $tf.idf$ based on three distinct parts of the articles: news titles, the first 30 words of the news article, and full body of the news article. The first 30 words are meant to be an approximation of the first paragraph of a news article, which typically contains a synopsis of the content of the article. In one set of features, we average the $tf.idf$ values over all terms in a query and represent it in the following features: *AvgNewsBodyTfIdf, AvgNewsTitleTfIdf, AvgNewsFirstParagraphTfIdf, AvgWikipediaTfIdf*. In a second set of features, we sum the $tf.idf$ values for all terms in the query: *NewsBodyTfIdfSum, NewsTitleTfIdfSum, NewsFirstParagraphTfIdfSum, WikipediaTfIdfSum*.

3.3 Context Features

One indicator of CTR we found in our experiments was the coherence among the documents surfaced by a query. Queries that surface a number of very similar documents are more likely to refer to one specific news event, in turn making high CTRs more likely. On the other hand, query terms that appear in a large number of very different news stories (e.g. "*NY Times*") are less likely to have high CTR. In order to measure the coherence of mentions of query terms in a corpus, we use a modification of the distance measures defined in [6] to model the difficulty of a given retrieval topic: we define a context as a text window of 10 tokens to the right and left of a an occurrence of a query phrase. We then take a sample of 50 contexts per query, and calculate the overall "cohesion" of these contexts. A simple but effective measure for cohesion is to look at the term vectors of the contexts and calculate a distance or similarity metric between the vectors. We use *Jensen-Shannon divergence* [16] and *Cosine Similarity* as distance measures. The features based on this notion are *AvgNewsJS* and *AvgBlogJS* and similarly, *AvgNewsCosine* and *AvgBlogCosine*. To illustrate the effect of this measure measure of coherence, we provide some examples of queries at the extreme ends of high and low AvgNewsJS values in table 1.

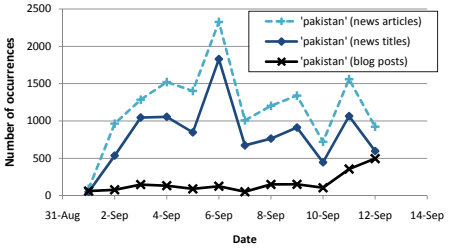

Figure 4: Occurrences of the term "*Pakistan*" in different corpora

"Specific" queries (low *AvgNewsJS*)	"Nonspecific" queries (high *AvgNewsJS*)
hanna montana	att
obama tax plan	boise state
lehman college	corporation
carol mccain	hcl
filing bankruptcy	kitchen appliances
red hot chili peppers	design services
mega millions lottery	long term care
iphone price	hindi
sarah palin action figure	sheikh

Table 1: Examples of phrases with different "context-coherence".

3.4 Query-Only Features

Query-only features capture information about the makeup of the query terms themselves. The first and obvious feature is query length in tokens. The presence of stop words (typically function words) is addressed by a feature measuring the ratio of stop words to query length. Some queries contain non-alphabetic characters, which may indicate a non-news related query. We compute the count of special characters in a query and the ratio of special characters. Navigational queries often consist of URLs or fragments of URLs ("*www.google*"). We therefore determine whether a query is a URL through a series of regular expressions. In addition, we simply detect whether the query contains the string "*www*".

Finally, capitalization of the terms that appear in a query can indicate named entities ("*Casey Anthony*") as opposed to common nouns. However, when entering queries in a search box, users only rarely add capitalization; instead, we test whether or not the terms occurring in a search query are typically capitalized when they occur in the news corpus. This is done as a pre-computation step, i.e. as the news corpus is updated, we pre-compute a list of terms and phrases that are commonly ($\geq 90\%$ of the time) capitalized in the

corpus. This can also be viewed as a very simple approximation to named entity detection. We use the number of words in the query that are typically capitalized, the ratio of such words to the query length, and the presence of a sequence of typically capitalized words as features in our classifier.

3.5 Overhead of Feature Extraction

In general, all data we use for our approach resides in main memory; for feature extraction we retain the text in the corpora itself, inverted indices on the corpora, as well as some word-level statistics (which can be encoded as part of the inverted index itself), plus a small set of pre-computed lists (such as *stop-words*, etc). Over time, the inverted indices are updated through a background process as new documents are crawled. We maintain blog and news corpora for a window of 7 days into the past.

Corpus Frequency Features: Here, we need to differentiate between the *tf.idf*-based statistics for individual words and the word-set/phrase counts we collect for different corpora. Regarding the former, we are able to maintain the required *term frequency* and *document frequency* statistics for each word in the vocabulary as part of the corresponding word's entry in the inverted index.

Computing the word-set counts (for multi-keyword queries) requires answering partial match queries (also known as containment queries), which are inherently expensive (e.g. [11]); in fact, the count features for word-sets or phrases are the only features whose computation induces non-negligible overhead. To compute the counts for sets of query terms, we use the inverted index intersections for the indices corresponding to the query terms. Because in our scenario all required corpora can fit into main memory on a single server, the cost of these queries is orders of magnitude lower than for disk-based indices; however, if the resulting latency is still too large, it can be reduced further by approximating the intersection sizes (see [14]) or simply using a smaller sample of the documents in each corpus. We evaluate the effect of using a subset of documents on the accuracy experimentally in Section 5.4. By encoding positional information together with the postings within the inverted index, we also compute the phrase counts using the inverted index intersections.

Query Context Features: Once we have computed the phrase count features, we can compute a sample of the contexts each phrase occurs in using the result of the inverted index intersection and extracting the context directly from the corresponding articles/posts. Because the corpora are kept in main memory, this extraction does not cause significant additional overhead. Moreover, we only use a small number of contexts (50 in our current implementation) for these features.

Query-Only Features: Most of these features can be derived quickly from the query itself; the only exceptions are the stop-word features for which we match the individual words in the query against a small hash-table of stop-words and the features dealing with capitalization. For the latter, we periodically compute a list of capitalized words or phrases via a background process that iterates over the current corpus. Because the capitalization of terms does not change quickly over time, this process does not need to be triggered frequently and does not constitute a performance issue. Once we have computed all such phrases, we need to match all subsets of words in a query against this phrase table, which corresponds to an common indexing problem in advertisement matching and can be addressed efficiently via the data structures described in [13].

3.5.1 Main Memory Requirements

The number of documents in our news crawl for a single day varies between 103K and 184K documents, with an average of 146K documents. We retain a window of 7 days of news into the recent past and retain only the news text from each article. For the blog corpus, we use blog documents posted on the same range of dates; the number of documents crawled for a single day vary between 31K and 97K, with an average of 75K documents.

Both the blog and news corpora are divided into separate sub-corpora, each of them containing articles/posts that were published/posted on a specific day; the news corpus is further subdivided into headlines, first paragraphs and complete text of the articles. We maintain separate inverted indices for each corpus; in addition, global statistics on word/document counts are maintained on a single-word basis (for the current 7-day window). Given that the news articles average slightly over 3K characters per document (the blog articles are shorter), a 7-day window of articles can easily fit in main memory on a 16GB server. As was shown in [19], *inverted indices* that uses a simple (and standard) gap-encoding scheme require space proportional to the size of the compressed text of the underlying corpora, which implies that the index structures required for feature extraction typically require significantly less space than the text corpora themselves.

4. LEARNING MODEL

In this section we will a give short overview of the learning model used in our approach; a more detailed description can be found in [26]. The learning method we use for CTR-prediction is based on *Multiple Additive Regression-Trees* (*MART*). MART is based on the *Stochastic Gradient Boosting* paradigm described in [10] which performs gradient descent optimization in the functional space. In our experiments, we used the log-likelihood as the loss function (optimization criterion), used the steepest-decent (gradient descent) as the optimization technique, and used binary decision trees as the fitting function - a "nonparametric" approach that applies numerical optimization in functional space.

In an iterative boosting (or residue-learning) paradigm, at the beginning of every iteration, the click probabilities of the training data are computed using the current model. The click prediction is compared with the actual click outcome to derive the errors (or residuals) for the current system, which is then used to fit a residue model – a function that approximates the errors – using *MSE* (Mean Square Error) criteria. In MART, we compute the derivatives of the log-loss for each training data point as the residual and use the regression tree as the approximation function-residual model. A regression tree is a binary decision tree, where each internal node splits the features space into two by comparing the value of a chosen feature with a pre-computed threshold; once a terminal node is reached, an optimal regression value is returned for all the data falling into the region. Finally, the residual model is added back to the existing model so that the overall training error is compensated for and reduced for this iteration. The new model – the current plus the residual model – will be used as the current model for the next boosting/training iteration. The final model after M boosting iterations is the sum of all M regression trees built.

Properties of MART: MART does have a number of important properties that are crucial for our scenario. In addition to providing accuracy superior to all other models we tried, MART is able to handle the diverse sets of features we have proposed in the previous section. For one, it does not require transformations to normalize the inputs into zero mean and unit variance which is essential for other algorithms such as logistic regression or neural nets. More importantly, by its internal use of decision trees, which are able to "break" the domain of each feature arbitrarily, it is able to handle the non-linear dependencies between the feature values and the CTR and without using explicit *binning* as a preprocessing step.

MART also computes the importance of a feature by summing the number of times it is used in splitting decisions weighted by the *MSE* gain this split has achieved. The relative importance of a feature is computed by normalizing its importance by the importance of the largest feature, i.e., the most important feature will have the relative importance of 1 and and other features will have relative importance between 0 and 1. The relative importance of input features makes the model interpretable - helping us gain an understanding of the input variables that are most influential to the final decision and the nature of the dependency on these features.

In addition to MART, we also tested different learning algorithms (using *Logistic Regression* and *Averaged Perceptron* models) on the same data, none of which produced comparable accuracy. We omit the details of this study due to space considerations.

1: $F_0(x) = 0$
2: **for** $m = 1$ to M do **do**
3: **for** $i = 1$ to N do **do**
4: $p_i = 1/\left(1 + e^{-F_{m-1}(x_i)}\right)$
5: $\hat{y}_l = y_i - p_i$
6: $w_i = p_i \cdot (1 - p_i)$
7: **end for**
8: $\{R_{lm}\}_1^L = L -$ Terminal node tree
9: $r_{lm} = \sum_{x_i \in R_{lm}} \hat{y}_l / \sum_{x_i \in R_{lm}} w_i$
10: $F_m(x_i) = F_{m-1}(x_i) + v \sum_l r_{lm} I(x_i \in R_{lm})$
11: **end for**

Algorithm 1: The MART Algorithm

The MART Algorithm: The above pseudo-code summarizes the MART algorithm; it assumes there are N total query-impressions (i.e., there were a total of N results shown) in our training set and that we wish to train M stages (trees). The training data is a set of input/output pairs $\{x_i, y_i\}, i = 1 \ldots N$, where x_i is the feature vector of a query and y_i is 1 if a query impression resulted in a click on news results and 0 otherwise. M rounds of boosting are performed, and at each boosting iteration, a regression tree is constructed and trained on all queries. Step 1 initializes the functional value of all data points to 0. Step 4 computes the probability of the query being clicked from its functional value. Step 5 calculates the log-likelihood gradients for query. Step 6 calculates the second-order derivative. A regression tree with L terminal nodes is built in step 8, using the Mean Squared Error to determine the best split at any node in the regression tree. The value associated with a given leaf of the trained tree is computed first as the mean of the gradients for the training samples that land at that leaf. Then, since each leaf corresponds to a different mean, a one-dimensional *Newton-Raphson* line step is computed for each leaf (Step 9). Finally, in Step 10, the regression tree is added to the current boosted tree model, weighted by the shrinkage coefficient v, which is chosen to regularize the model. MART has three main parameters: M, the total number of boosting iterations, L, the number of leaf nodes for each regression tree, and v, the "shrinkage coefficient" - the fraction of the optimal line step taken. Using a shrinkage coefficient with value less than one is a form of regularization [10].

We further injected randomness in MART to improve the robustness of the resulting model. In our experiments, the most effective method is to introduce randomness at the node level. Before each node split, a sub-sample of training data and a sub-sample of features are drawn randomly. Then, the two randomly selected subsamples, are used to determine the best split.

5. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of our techniques. First, we describe the characteristics of the training and test data sets we used in our experiment (Section 5.1). Subsequently, we describe the overall accuracy of our approach and the importance of different classes of features (Section 5.2). We assess how well our CTR-prediction generalizes to a corpus of general web queries (as opposed to the corpus of mostly news-related queries for which we were able to observe click-through rates) in Section 5.3. Because the size of the underlying news and blog corpora is the deciding factor in the cost of feature extraction, we analyze the effect of reducing their size through sampling in Section 5.4.

5.1 Training/Test Data

To evaluate our approach, we use CTR data obtained from *Live.com*. The data contains web search queries for which news results were displayed and – for each individual *impression* the information whether a click on the news result occurred or not. By aggregating this data for each query, we can derive the observed CTR for each query and compare it to the estimate given by our technique. Obviously, the value of the "observed" CTR depends on at which granularity we aggregate the data; for our experiments we measure the CTR at the level of days, corresponding to the 24-hour news cycle. We remove any queries for a given day from the data set for which there were not at least 50 impressions during the day, since the small number of impressions makes it impossible to assess the underlying "true" CTR with any degree of confidence. This may bias our initial results towards popular queries; we will discuss experiments on general web traffic in Section 5.3.

The training/test data we collected contains a total of 2.7 million impressions, covering 5 days in September 2008. All of the news results shown for the queries in our data set were shown in the same position within the search engine, i.e., we do not have to adjust for the effect of result position on CTR. Note that this training set exhibits bias towards news queries: because it only contains queries for which the search portal surfaced news results, it does not reflect the overall distribution of web queries. In particular, it contains very few queries that have no news intent (and hence would have zero or almost zero CTR), which, however, is common in normal search traffic. To address this, we adopt two approaches: (1) We use manual analysis of a web search query log to identify a set of queries for which there was no news intent; these were then added to the training and test sets with a CTR of 0 (we will refer to these queries as the 0-CTR queries subsequently). These queries account for a total of 2.6 million impressions, meaning that our training and test data contain about half non-news queries and half queries for which news results were shown by the search engine. (2) In addition, we explicitly evaluate how our estimator performs on a large sample of web-queries obtained from a search query log in Section 5.3.

To avoid any test/training contamination, we keep the test and training sets strictly separate with respect to queries (i.e., no instance of a query/day combination can appear in both). When training MART, we used the parameter settings of the number of iterations M = 600, the number of leaf nodes in an individual decision tree L = 5, and the shrinkage coefficient v = 0.1.

5.2 Accuracy

In this section we evaluate the overall accuracy of CTR estimation and the contributions of individual groups of features. As a measure of performance we will use the *log-loss* on the held-out test set. In addition, to give a better intuition for the quality of our approach in practice, we measured the accuracy of our approach when used to estimate whether a given query's CTR (in the test

set) exceeds a certain threshold t. This is a key metric for a scenario where we only surface news results whose CTR exceeds this threshold. We have plotted the resulting precision/recall curves for $t = 10\%, t = 15\%$ and $t = 20\%$ in Figure 5. For larger target thresholds the precision is very high; for example, for a threshold of $t = 20\%$, the precision over the entire test set is 90.9%; trading off precision against recall, we can achieve 99.1% precision at 61% recall and 99.7% precision at 50% recall. Given that the perceived penalty for a "false positive" (i.e., showing irrelevant news results for a non-news query) is much larger than for a "false negative" (i.e., not showing news results for a query with news-intent), trading off recall for precision is important in practice. Overall, we observe that the proposed technique already achieves sufficient accuracy on the test data to be relevant for practical use. To put these numbers in perspective, the baseline probability for predicting CTR $> 20\%$ is 81.8%, the baseline for predicting CTR $> 15\%$ is 75.9% (our predictor is correct 87.9% of the time) and for predicting CTR $> 10\%$ is 70.1% (our predictor is correct 82% of the time).

We approximate the CTR of 82.5% of all queries within an "error-band" of +/-10% of their true CTR (i.e. the *absolute* difference in true CTR and estimated CTR is less than 0.1) and 59.8% of all queries within +/-5% of their true CTR.

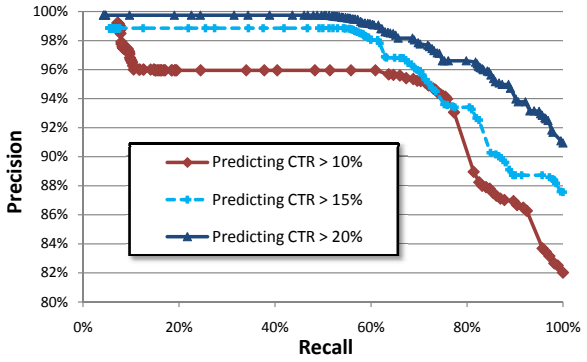

Figure 5: Precision/Recall for predicting different CTR threshholds

To measure the importance of different corpora and sets of features we conducted ablation experiments in which we removed some corpora (and hence the features based on them) from consideration. For this experiment, we considered the overall log-loss on the test data for (a) removing the news-based features, (b) removing the blog-based features, (c) removing the Wikipedia-based features and (d) removing the context features from the full feature set. The results are shown in Figure 6; the Wikipedia-features have the least impact on accuracy, whereas the news and blog features have the most. Moreover, both the news and blog corpus detect different underlying trends and appear to complement each other.

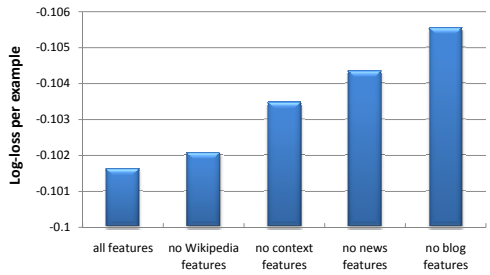

Figure 6: Accuracy for different feature-sets

5.2.1 *Feature Importance*

We also measured the relative importance of different features directly using the measure of feature importance described in Section 4. The top-ranking feature was the binary encoding of whether a query corresponded to a URL; this was mainly due to the fact that our 0-CTR contained a large number of navigational queries. The next-most important features were (in order): the averaged *tf.idf* scores of the query terms relative to the Wikipedia corpus, the BM25 score of the top-ranking document, the sum and average of the *tf.idf* scores of the query terms relative to the news corpus, the number of times the query terms occurred in news titles one and two days prior to the query being issued and finally the Jensen-Shannon divergence of the query term's context in the blog corpus. Features leveraging all 3 corpora (Wikipedia, news and blogs) were present among these top-ranked ones. Also, while the BM25 score is in itself not sufficient to predict click-through rates, it did turn out to be a very important feature when combined with the corpus-based ones. We also found that context-features based on Jensen-Shannon divergence consistently ranked higher than ones based on Cosine similarity.

5.3 Generalization to Web Query set

As we pointed out earlier, the training/test data sets we used in the previous evaluation are biased towards news-related queries (when compared to general web search traffic), even after we manually added the 0-CTR data. However, since our CTR-prediction is aimed at general web search traffic, it is crucial that it also performs well for non-news queries. In order to assess how well our approach generalizes to this type of queries, we first trained a CTR predictor using only the part of training data based on observed click-through data (i.e., without the batch of 0-CTR data). Then we took the 10K search queries most frequently issued against *Live.com* on September 9th, 2008, and ranked the queries by the estimated CTR. The resulting CTR-estimation was then (because of the absence of any observed CTR for nearly all queries) manually evaluated.

To give a quick overview of the result, we have plotted the top and bottom 15 queries in Table 2. All of the top queries refer to an event that generated significant news coverage at the time, whereas none of the bottom 15 do. In fact, every single one of the top 50 queries in the log corresponded to a "news event" or the name of a person generating news coverage on that day. To contrast this with the remainder of the search query distribution we manually analyzed 100 queries each starting at the rank-positions (according to the estimated CTR) 1000, 3000 and 5000. Among these 300 queries, we found only 3 queries referring to a recent news event (2x in the block starting at 1000 and 1x in the block starting at 3000) and 5 further mentions of names from the area of entertainment for which we were unable to determine if they had been involved with a news event around the date the logs was taken.

Overall, it appears that our CTR prediction does indeed generalize to the distribution of frequent web queries we studied. Given the difficulty human annotators face in predicting click-through rates, this evaluation can naturally only serve as an indicator, but was the best we could do in absence of any observed click-through data. We performed the same experiment for two additional subsets of the query log, using the queries ranked (by their frequency) between 20K-25K and the queries ranked between 100K-105K. The results were similar to the ones described above.

5.4 Varying the Corpus Size

In this experiment we examined the need for using the entire news and blog corpus; here we ran experiments using features sets which were extracted from random, document-level samples of va-

Top Queries	Bottom Queries
hurricane in texas	msn
palin doll	learn from users just like
natalie dylan	learn
caylee	aol
casey anthony	verizon
caylee anthony	walmart
super lice	craigslist
ike	hotmail
tropical storm josephine	environmental
evan tanner	yellow pages
hurricane ike projected path	you tube
kanye west arrested	yahoo
greatest american dog	travelocity
eric mclean	corporation
palin gibson interview	white pages

Table 2: Queries with most/least estimated CTR in web query log.

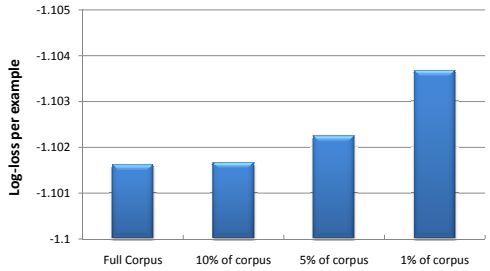

Figure 7: Accuracy for different corpus sizes

rying size of the underlying corpora. The results are shown in Figure 7. Retaining only 10% of the document corpus has limited impact on accuracy (even less than e.g., removing the Wikipedia-based features), but in turn means a reduction of approximately 10x in the main memory requirement and also a significant decrease in the feature computation costs. Any further reduction in size does, however, significantly impact accuracy.

6. CONCLUSION AND OUTLOOK

In this paper we describe a new approach for deciding when to surface news context as part of "regular" search results. We propose a supervised framework based on decision trees and boosting, which leverages features that characterize properties such as contextual coherence, temporal spikes and occurrence frequency of the query keywords relative to large text corpora. By continuously updating the underlying news and blog corpora via dedicated crawlers, this framework allows us to keep up with and adapt to emerging news stories. We found that this framework allows for highly accurate CTR prediction. Here, the use of both news as well as social media data (i.e., blogs) is important to the overall accuracy; both corpora have significant impact and complement each other.

As next steps, we intend to examine the importance of additional data sources as well as further differentiation of the existing corpora; for example, we currently view blogs as a single corpus, without differentiating between news, entertainment and IT blogs/posts. Also, it remains to be seen how much more complex features, which require more computational overhead (such as context-features leveraging occurrences of multi-word sub-phrases), can improve accuracy further and which index structures are needed to support them efficiently.

7. ACKNOWLEDGEMENTS

We are very grateful for the help and advice of Mikhail Bilenko, Nick Lester, Roger Menezes and Momo Jeng.

8. REFERENCES

[1] D. Agarwal, B. Chen, P. Elango, R. Ramakrishnan, N. Motgi, S. Roy, and J. Zachariah. Online Models for Content Optimization. In *NIPS*, 2008.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving Web Search Ranking by Incorporating User Behavior Information. In *ACM SIGIR*, pages 19–26, 2006.

[3] J. Allan, R. Papka, and V. Lavrenko. On-line New Event Detection and Tracking. In *ACM SIGIR*, 1998.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank using Gradient Descent. In *ICML*, 2005.

[5] J. Callan. Distributed Information Retrieval. In *Advances in Information Retrieval*, pages 127–150, 2000.

[6] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a Query Difficult? In *ACM SIGIR*, 2006.

[7] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual Advertising by Combining Relevance with Click Feedback. In *WWW Conference*, 2008.

[8] G. M. D. Corso, A. Gullí, and F. Romani. Ranking a Stream of News. In *Proceedings of the 14th international conference on World Wide Web*, pages 97–106, 2005.

[9] F. Diaz. Integration of News Content into Web Results. In *WSDM Conference*, 2009.

[10] J. Friedman. Greedy Function Approximation: a Gradient Boosting Machine. *Annals of Statistics*, 29(5), 2001.

[11] T. Jayram, S. Khot, R. Kumar, and Y. Rabani. Cell-Probe Lower Bounds for the Partial Match Problem. In *STOC*, 2003.

[12] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. *Information Processing and Management*, 36(6), 2000.

[13] A. C. König, K. Church, and M. Markov. A Data Struture for Sponsored Search. In *IEEE ICDE Conference*, 2009.

[14] P. Li and K. W. Church. Using Sketches to Estimate Two-way and Multi-way Associations. *Computational Linguistics*, 33, 2007.

[15] X. Li, Y.-Y. Wang, and A. Acero. Learning Query Intent from Regularized Click Graphs. In *In Proc. ACM of SIGIR*, 2008.

[16] J. Lin. Divergence Measures based on the Shannon Entropy. *IEEE Trans. on Information Theory*, 37(1), 1991.

[17] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[18] A. Maykov and M. Hurst. Social Streams Blog Crawler. In *M3SN Workshop at IEEE ICDE Conference*, 2009.

[19] G. Navarro and V. Mäkinen. Compressed full-text indexes. *ACM Comput. Surv.*, 39(1), 2007.

[20] F. Radlinski and T. Joachims. Active Exploration for Learning Rankings from Clickthrough Data. In *ACM SIKDD*, 2007.

[21] M. Regelson and D. Fain. Predicting Click-through Rate using Keyword Clusters. In *2nd Workshop on Sponsored Search Auctions*, 2006.

[22] M. Richardson, E. Dominowska, and R. Ragno. Predicting Clicks: Estimating the Click-Through Rate for New Ads. In *WWW Conference*, pages 521–529, 2007.

[23] Technorati. State of the Live Web. http://technorati.com/weblog/2007/04/328.html, April 2007.

[24] Technorati. State of the Blogosphere 2008. http://technorati.com/blogging/state-of-the-blogosphere/, 2008.

[25] M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer-Verlag, 1997.

[26] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Ranking, Boosting, and Model Adaptation. Technical report, Microsoft Research, 2008.