Improving Search Result Summaries by Using Searcher Behavior Data

Mikhail Ageev*
Moscow State University
mageev@yandex.ru

Dmitry Lagun
Emory University
dlagun@emory.edu

Eugene Agichtein
Emory University
eugene@mathcs.emory.edu

ABSTRACT

Query-biased search result summaries, or "snippets", help users decide whether a result is relevant for their information need, and have become increasingly important for helping searchers with difficult or ambiguous search tasks. Previously published snippet generation algorithms have been primarily based on selecting document fragments most similar to the query, which does not take into account which parts of the document the searchers actually found useful. We present a new approach to improving result summaries by incorporating post-click searcher behavior data, such as mouse cursor movements and scrolling over the result documents. To achieve this aim, we develop a method for collecting behavioral data with precise association between searcher intent, document examination behavior, and the corresponding document fragments. In turn, this allows us to incorporate page examination behavior signals into a novel Behavior-Biased Snippet generation system (BeBS). By mining searcher examination data, BeBS infers document fragments of most interest to users, and combines this evidence with text-based features to select the most promising fragments for inclusion in the result summary. Our extensive experiments and analysis demonstrate that our method improves the quality of result summaries compared to existing state-of-the-art methods. We believe that this work opens a new direction for improving search result presentation, and we make available the code and the search behavior data used in this study to encourage further research in this area.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Result summary generation, searcher behavior, mouse cursor movement

1. INTRODUCTION

While web search engines have been rapidly evolving, one constant in the search result pages has been the presence of some form of a document summary, or "snippet", provided to help a user to select the best result documents. For some queries, the generated snippets already contain the desired information, either by design (e.g., Google's "Instant Answers"[1] or Wolfram Alpha[2]) or by serendipity. Search engines have been increasingly successful at this, as evidenced by work on "good abandonment" [25] and others. Nevertheless, for a large class of queries where the desired information need is either ambiguous or cannot be answered succinctly, result snippets play a crucial role in guiding the users to the needed documents.

Our aim is to improve snippet generation for *informational* queries [7]. Specifically, our goal is twofold: If possible, the snippet text should include the desired information directly. Otherwise, the snippet should provide sufficient information for the user to distinguish the useful from the non-useful results. Fortunately, millions of users search daily, clicking on results and examining the documents. Our key insight is that by capturing such examination data, we could relate the *areas of interest* in the document to the *search intent*, subsequently generating more useful snippets for future searchers.

As a concrete example, consider an information need, specified as "How many pixels must be dead on a iPad 3 before Apple will replace it.". After issuing the search query "how many dead pixels ipad 3 replace", the user examined the document to find the answer, as shown in Figure 1 (the eye gaze positions were collected by using infra-red eye tracking). Note that while the user examined both useful and non-useful document fragments, she eventually focused on the best fragment containing the answer to the question. Hence, for this search intent and this document, an ideal snippet would include the text fragment where the user focused her attention. In contrast, if a document was *not* relevant, a good snippet should also include the area of most interest, as it could help users with the same intent to avoid visiting this result in the future.

This example intuitively motivates our goal to generate *Behavior-Biased* Snippets, or BeBS, which, as we show experimentally, can significantly improve snippet quality, which has been shown in eye-tracking studies to significantly affect search behavior (e.g. [10]). While putting an eye-tracker on every desk is not yet feasible, we can still *infer* searcher interest in particular page regions from their mouse cursor and scrolling behavior (e.g., following the ideas of [18] and [24]), and then incorporate this evidence to make the snippets more useful. To the best of our knowledge, our work is the first to successfully incorporate such behavior data into result snippet generation.

*Work done at Emory University.

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

[1] http://googleblog.blogspot.com/2010/09/google-instant-behind-scenes.html
[2] http://www.wolframalpha.com/

Figure 1: Attention heatmap of result document examination for the search intent "How many pixels must be dead on a iPad 3 before Apple will replace it." The user's attention is focused on a text fragment (highlighted in the callout box), containing the answer.

Specifically, our contributions include:

- A novel *behavior-biased* approach to snippet generation, that naturally integrates textual and behavioral evidence to improve snippets for informational queries (Section 3).
- A robust infrastructure for precisely associating fine-grained searcher behavior with document content (Section 3.3).
- Thorough experiments over hundreds of search sessions and thousands of page views, demonstrating significant improvements to snippet quality by harnessing searcher behavior data (Section 6).

Next, we describe related work to place our contributions in context.

2. RELATED WORK

Our work touches on three areas of research: search result summary (or snippet) generation, document summarization, and modeling searcher behavior and examination data.

Document Summarization and Search Snippet Generation Automatic summarization has been explored in many research areas including natural language processing, artificial intelligence and information retrieval. In particular, summaries desired in information retrieval application are mostly short, succinct text fragments informing user about document's relevance to the user's information need. Among various approaches for generating such summaries, the most popular is extracting sentences or sentence fragments from a document. Early work of Kupiec et al. [29] addressed sentence selection problem, using a Naive Bayes classifier to predict whether a sentence should be included in a static, query-independent document summary. Subsequently, Tombros and Sanderson [37] demonstrated that query-biased summaries are more advantageous than static or query independent summaries in allowing users to identify relevant documents with higher accuracy. One approach is to use linguistic and relevance features (e.g., Goldstein et al. [16]). Another related area is automatic question answering, where a system could perform deep semantic analysis to identify the question or query intent, and attempt to return an answer directly (e.g., [6, 21, 12]. Our approach is complimentary, as it could be used to improve selecting candidate answers by any of the text-based snippet generation systems. Most closely related to our goal is the research on query-focused summarization (e.g., [4, 13, 14]), which uses a variety of document- and query text features to generate a document summary focused on a given query. These ideas were further extended in machine learning-based approaches to web search snippet generation (e.g., [33, 26, 27]). Our work builds on these ideas by adding a radically different type of evidence –search behavior data– to make the result summaries more useful to the searcher.

Eye Tracking for Web Search Result and Document Examination: There is a rich history of using eye tracking technology to identify areas of interest and attention, and to study reading behavior. In the context of web search document examination, Buscher et al. [9] extracted sub-documents by tracking eye movements as implicit feedback and expanded search queries to improve the search result ranking. Buscher et al. also studied the prediction of salient Web page regions using eye-tracking [8]. This work, and others, have shown that user attention can help identify regions of documents of particular relevance or usefulness for the query. While eye tracking equipment limits the applicability of these findings, these studies served as inspiration to our work on biasing snippet generation to privilege the *inferred* areas of interest.

Mouse Cursor Tracking in Web Search: Recently, mouse cursor tracking has been proposed as a natural proxy for user's attention, to replace the requirement for eye tracking equipment. One of the earliest work of correlating cursor and gaze position in web search was by Rodden et al. [35], where the authors discovered the coordination between a user's eye movements and mouse movements when scanning a web search results page. Guo and Agichtein [18] extended this work to predict eye-mouse coordination (i.e., whether the mouse cursor is in close proximity to eye gaze at any given point in time) by modeling mouse movements. This work was further extended by Huang et al. to directly predict the gaze position from mouse cursor movement [23], with over 70% accuracy. Thus, there is mounting evidence that searcher attention in web search can be approximated by using mouse cursor, scrolling, and other interaction data.

Modeling Cursor Movement on Landing Pages: In addition to studying cursor movement on Search Engine Result Pages (or SERPs), mouse cursor analysis work has been recently applied to examination of the clicked documents, or landing pages. Hijikata et al. [22] proposed a method to extract text fragments of Web pages based on the user's mouse activity and found that the extracted fragments based on

mouse activity such as *text tracing, link pointing, link clicking and text selection* enable more accurate extraction of key words of interest than using the whole text of the page. More recently, White and Buscher [39] proposed a method that uses text selections as implicit feedback, while Guo and Agichtein [19] proposed a *Post Click Behavior (PCB)* model to estimate the "intrinsic" document relevance by modeling post-click behavior such as mouse cursor movements and scrolling. Our method extends this approach to identify specific regions, or fragments of the page, of particular interest to the users for including in the result summary.

In summary, our work extends previous state-of-the-art in snippet generation by considering a radically new source of evidence, namely the user examination data, in order to bias the resulting summaries towards the most "interesting" or "useful" parts of the document, as we describe next.

3. PROBLEM STATEMENT AND APPROACH

First, we formalize the problem of generating "useful" snippets. Then, we describe the key parts of our approach (Section 3.2), and the infrastructure we developed to accomplish the required data collection (Section 3.3).

3.1 Problem Statement

Following the literature on snippet quality [31], snippets must satisfy the aspects of *Representativeness*, *Readability*, and *Judgeability*:

1. *Representativeness*: measures how well the snippet summarizes parts of the web page relevant to the search query. A representative snippet would clearly show why the page was found by a search engine in response to the query.

2. *Readability* measures the ease with which the text of the snippet can be read and understood. Note, that readability does not depend on the search query [27].

3. *Judgeability* measures how well the snippet helps a user to understand whether the page is helpful for the specific search intent, and to decide whether to click on a link or not. An ideal snippet would either contain an answer to a user's information need (or a clear indication that an answer is present in the document), or else clearly show that the page is not relevant .

Our primary goal is to optimize the Representativeness and the Judgeability criteria by *biasing* the selected snippets towards the regions of most interest to the user, as inferred from the page examination data. That is, our goal is not to replace the existing text-based snippet generation approaches, but rather to add additional evidence (when available) about the parts of the document to privilege.

3.2 Approach

Our approach operationalizes the snippet quality criteria above by incorporating both textual and behavioral evidence using a robust machine learning-based approach. Specifically, we combine together the traditional text-based snippet generation features, and the inferred user interest in specific parts of a document.

First, following [27], a fragment scoring system is trained based on text-based features, using human judges, resulting in a strong text-only baseline that generates *candidate fragments* to be included into the snippet (Section 4.1). Separately, examination behavior data is collected over the landing pages, using our logging infrastructure described in the next section. Then, a behavior model is trained to infer the document fragments of interest to the user, based on user examination data (Section 4.3). Finally, the behavior-based prediction of interest in each candidate fragment is combined with the original (text-based) fragment score, in order to generate the final *behavior-biased* snippet candidate ranking (Section 4.4). Note that by decoupling the behavior modeling from the candidate generation method, our approach can be used with any other snippet generation approach that provides scores for the candidate fragments, which could be combined with the behavior scores for the final ranking step.

While general and flexible, our approach makes three key assumptions. First, our method is primarily targeted (and evaluated for) informational queries – that is, queries for which the user expects to find an answer in the text of the page, and optimizes the snippets accordingly. Second, we assume that document visits can be grouped by query intent, so that behavior features on the landing pages can be aggregated together for all the searchers with the same information need. While a number of methods have been proposed to cluster queries (and results clicks) by intent (e.g., [34]), we acknowledge that these techniques are not perfect, and may introduce noise in practice. Finally, we assume that user interactions on landing pages can be collected by a search engine or a third party. While this naturally introduces potential privacy concerns, this assumption is not far-fetched: already, browser plug-ins and toolbars collect user interactions on web pages; major organizations can (and often do) use proxies for external web access; and common page widgets like banner ads and visit counters inject JavaScript code to monitor basic user interactions and can be easily extended to collect more detailed data. While the privacy and security considerations of these methods are beyond the scope of this paper, we merely point out that these behavior gathering tools already exist and are widely deployed. We will discuss these issues and potential solutions in more depth in Section 7.

3.3 Page Examination Behavior Logging

A key component of our system is a mechanism for collecting searcher interactions on web pages, and tying them precisely to the page content at the word level. As a starting point, we adapt the publicly available EMU toolbar for the Firefox browser [17], that is able to collect mouse cursor movements over any visited webpage. Unfortunately, out-of-the-box EMU functionality is not sufficient, as the user interactions are not connected to the underlying page content: the available JavaScript API does not provide the text position under the cursor, which could depend on screen resolution, size of browser window, browser version, and personal browser settings.

To associate the tracked mouse cursor positions with corresponding text fragments we employed the following technique. After the HTML page is rendered in the browser window, our JavaScript code modifies the document DOM tree, so that each word is wrapped by a separate DOM element tags. Then for each DOM Element, the window coordinates of that element are evaluated and saved in the Element's attributes. Then, the processed HTML page with the coordinates of each DOM Element is saved to the server by an

asynchronous request. The saved coordinates are updated if the page layout is changed due to a *resize* window event or an AJAX action.

Thus, for each page visit we know the searcher's intent (question), the search engine query that the user issued, the URL, the contents of the document, the bounding boxes of each word in the HTML text, and the log of behavior actions: mouse cursor coordinates, mouse clicks, scrolling, and an answer to the question that the user found in a page and submitted in the game interface. Next, we show how to use this information to infer patterns of browsing behavior that capture portions of document that are of most interest to the user.

4. BEHAVIOR-BIASED SNIPPET GENERATION

We now present the details of our Behavior-Biased snippet generation system (BeBS). First, we describe the text-only snippet generation system (Sections 4.1 and 4.2). Then, we introduce the method for inferring the most interesting or useful parts of the document from user behavior (Section 4.3), to incorporate into the combined snippet generation process (Section 4.4).

4.1 Text-Based Snippet Generation

In order to generate snippets, we extend the approach presented in Metzler and Kanungo [33]. The downloaded HTML pages are pre-processed and indexed with Natural Language Tool Kit (NLTK [3]). Extracted text is divided into sentences using *Punk* unsupervised sentence splitter [28]. We index the text of the web page excluding the <script> and <style> tags.

For a given query we first select all the sentences that have at least one match of query terms for further snippet fragment generation. Once the set of sentences is selected, our system generates all possible snippet fragment candidates by applying a sliding window moving along each sentence. We vary fragment length from 3 words to a maximum character length provided as an input parameter. We discard all fragments that do not contain any query term matches. Along with fragment generation our system scores each fragment using $TextScore$ function described in Section 4.2. This score is used to generate the final snippet.

The problem of summary generation has been studied extensively in natural language processing and summarization research communities. It has been shown in [36] that this problem is equivalent to a weighted set cover problem which is, in turn, known to be NP-Hard. There are several possible approaches, including greedy weighted set cover and relaxations, primarily based on integer linear programming. We resort to a greedy algorithm, due to relative simplicity of implementation. Our system can easily be extended with more advanced set cover solver if needed. As the set cover algorithm requires score computation for the set of selected fragments, we recompute the scoring function for the union of selected candidate fragments to find a set that greedily maximizes the score for the entire snippet.

4.2 Fragment Scoring

The fragment scoring required for snippet generation relies on a machine learning approach based on set of text features representing various quality aspects of fragment candidate. We extend the method of [33] by adding additional features capturing relevance of the fragment (relevance group), properties of query match (query match group) and readability of the fragment (readability group). These features are summarized in Table 1. For $TextScore$ score computation we used the Gradient Boosting Regression Tree model [15] (GBRT). GBRT is a powerful family of models that has been successfully used in many applications including sentence selection for search result summarization [33] and search result snippet readability assessment [27]. We train a GBRT model on a subset of training query-URL pairs to predict snippet fragment scores.

Gradient Boosting Regression Tree performs a numerical optimization in function space instead of parameter space. We provide a brief overview of the algorithm and refer to the original paper for detailed information [15]. A regression tree model $f(x), x \in R^n$, partitions the space of covariates into disjoint intervals $R_k, k = 1, 2, ..., K$ associated with leaf nodes of the tree. Each interval is assigned a value ϕ_k, such that $f(x) = \phi_k$ if $x \in R_k$. Thus, the tree model can be written in

$$T(x; \Theta) = \sum_{j=k}^{K} \phi_k I(x \in R_k)$$

where $\Theta = \{R_k, \phi_k\}_{k=1}^{K}$, and I is an indicator function. For a given loss function $L(y_i, \phi_i)$ the parameters Θ are result of the following optimization problem:

$$\hat{\Theta} = argmin_{\Theta} \sum_{k=1}^{K} \sum_{x_i \in R_k} L(y_i, \phi_k)$$

In our experiments we employ the squared loss function to train the regression trees. A gradient boosted regression tree is an ensemble model [15] that incorporates a series of regression trees, and can be written as:

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

where at each stage m, Θ_m is estimated to fit the residuals from the $m-1$th stage:

$$\hat{\Theta}_m = argmin_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \phi_{k_m})$$

and M is the number of stages (regression trees) in the model. In practice, one adds $T(x; \Theta_m)$ multiplied by ρ – the learning rate input parameter specified for the algorithm, resulting in the final predictor:

$$f_M(x) = \sum_{m=1}^{M} \rho T(x; \Theta_m)$$

In our implementation we used $M = 200$ regression trees, and $\rho = 0.01$ to train the GBRT model for fragment scoring.

4.3 Inferring Relevant Text Fragments from Search Behavior

To infer the text fragment importance from the user's browsing behavior, we again apply supervised machine learning, namely the Gradient Boosting Regression Tree (GBRT) algorithm [15], trained to identify "interesting" text fragments. Specifically, for each page visit of a user, the document is represented as a set of short text *fragments*. We consider a particular fragment to be "interesting" (attractive), if the user submitted an answer in the current session,

Feature	Description	Feature Group
ExactMatch	1 if fragment contains query as a substring, otherwise 0	Metzler - Kanungo
TermOverlap	Overlap of query terms and the fragment	
SynOverlap	Overlap of query terms expanded with synonyms and the fragment	
LanguageModelScore	Fragment score under the language model as in [33]	
Length	Total number of terms	
Location	Relative location of the fragment in the document	
BM25ScoreFragment	BM25 score of the fragment	Relevance
BM25ScoreSentence	BM25 score of the sentence from which fragment was extracted	
BM25ScorePerWord	BM25 of the fragment divided on number of words in the fragment	
NumMatches	Absolute count of query terms matched in the fragment	Query Match
SentenceBegDistance	Number of words between beginning of the sentence and first word in the fragment	
SentenceEndDistance	Number of words between end of the sentence and last word in the fragment	
QueryTermDistanceAvg	Average distance of query terms in the fragment measured in words	
QueryTermDistanceMin	Minimum distance of query terms in the fragment measured in words	
QueryTermDistanceMax	Maximum distance of query terms in the fragment measured in words	
NumDistinctTerms	Number of distinct terms in the fragment	Readability
NumPunctChar	Number of punctuation characters	
PercentPunctChar	Percent of punctuation characters	
NumLetterChar	Number of letter ([a-zA-Z]) characters	
NumWordsCap	Number of words with first letter capitalized	
PercentWordsCap	Percent of words with first letter capitalized	
PunctPerWord	Number of punctuation characters per word in the fragment	

Table 1: Text-based features for text fragments

and the answer shares words with the fragment (after stemming and stopword removal). Other fragments are labeled to be not interesting.

For each fragment we create a set of behavior features that could capture fragment interestingness. One key feature is the duration of time when the mouse cursor was placed over the text fragment, or very close to the fragment. We also adapt the features to measure scrollbar and event activity from references [11] and [19], in order to detect "reading" vs. "skimming" behavior. The complete list of the fragment behavior features is presented in Table 2. Note that these features are exclusively focused on capturing user's behavior associated with focused attention, and, by design, do not contain any document or query information, in order for the learned behavior model to be applicable to unseen documents and queries.

The feature generation algorithm joins a sequence of behavior events and a set of bounding boxes for each word and DOM Element of a page. To speed up this join, our implementation uses a spatial R-Tree index of element bounding boxes, which allows for each logged event to be efficiently matched to the matching DOM Elements in the specified coordinate range.

We then train the GBRT model on the training set of the labeled document fragments, each represented using the behavior features described above. The data is stratified by the original document URLs, so that the URLs of the fragments in the training and test sets are disjoint. The training set is created from only those page visits where the document text has a non-empty intersection with the user's answer, and the answer is correct. The trained prediction model is then applied to all page visits in the test set. Note that when the predictor is applied on the test set, it has no information about the user's intent, answer, or the current query, and uses only the behavioral features of the current page visit. The predicted behavior-based fragment interestingness score, is then used as a key evidence for the final snippet generation algorithm, described next.

Feature	Description
MouseOverTime	Time duration when the mouse cursor was over the text fragment
MouseNearTime	Time duration when the mouse cursor was close to the text fragment in the window ($x \pm 100px$, $y \pm 70px$)
MouseOverEvents	The number of mouse events during *MouseOverTime*
MouseNearEvents	The number of mouse events during *MouseNearTime*
DisplayTime	Time duration when the text fragment has been visible in the browser window (depends on scrollbar position)
DispMiddleTime	Time duration when the text fragment was visible in the the middle part of the browser window

Table 2: Behavior features for text fragments

4.4 Combining Text and Examination Evidence

The final step in our approach is to *combine* the text-based score $TextScore(f)$ for a candidate fragment (Section 4.1) with the behavior-based interestingness score $BScore(f)$ (Section 4.3), inferred from the examination data. In our current implementation we combine these scores by linear combination:

$$FScore(f) = \lambda \cdot BScore(f) + (1 - \lambda) \cdot TextScore(f)$$

Note that $TextScore(f)$ is not normalized, and could have values in the range $[1, 5]$, while $BScore(f)$ is normalized to be between 0 and 1.

The parameter λ affects two characteristics of the algorithm: snippet *coverage* and *quality*. Snippet *coverage* is defined as the ratio of the snippets produced by the behavior-biased algorithm that differ from the snippets produced by the text-only baseline. Snippet *quality* is measured by judge-

ability, readability, and representativeness metrics, via manual assessments. As λ approaches zero, coverage would also approach zero (as text-based features would dominate candidate selection), and the algorithm effectively backs off to the baseline. In contrast, when λ is large, snippet *quality* might decrease by weighing the behavior-based score too highly compared to the text-based score. We performed manual assessments for five different parameter values of $\lambda \in [0, 1]$ to select the best value. Other more sophisticated ways to combine text and behavior evidence are possible, such as jointly learning over both text and behavior features, as could be explored in the future. However, we chose to follow the simpler linear approach for better interpretability of the results (e.g., by analyzing the results of varying the λ parameter).

5. DATA COLLECTION AND EXPERIMENTAL SETUP

This section presents the methodology used for acquiring search behavior data for training our system (Section 5.1), describes the resulting behavioral data (Section 5.2), the explicit snippet judgments dataset used for training and validating the text-based snippet generation baseline (Section 5.3). Note that the code and the data used for experiments are available from `http://ir.mathcs.emory.edu/intent/`.

5.1 Acquiring Search Behavior Data

To collect the search behavior data, we used the infrastructure created and published by [1], and modified it for our task. The participants played a search contest "game" consisting of 12 search tasks (questions) to solve. The stated goal of the game was to submit the highest possible number of correct answers within the allotted time. After the searcher decided that they found the answer, they were instructed to type the answer together with the supporting URL, into the corresponding fields in the game interface. Each search session (for one question) was completed by either submitting an answer, or by clicking the "skip question" button to pass to the next question.

Participants were recruited through the Amazon Mechanical Turk (MTurk) website. As a first step, the workers had to solve a ReCaptcha puzzle to verify that they are human and not an automated "bot". A browser verification check was performed to verify that the browser is compatible with our JavaScript tracking code. During the data postprocessing stage, we filtered out the users who did not answer even the easy, trivial questions, as it indicated either poor understanding of the game rules, or an attempt to make a quick buck without effort.

To capture all of the participants' search actions, they were instructed to use only our search interface. Our search interface performs web search using the public API of a popular web search engine (we used Bing for our experiments), and displays the result pages using the original page design, layout and stylesheets, so the user's search experience is not affected. The Apache Web server proxy functionality was used by configuring the modules *mod_proxy*, *mod_proxy_html*, and *mod_sed* so that the users could search and browse the Web as usual, while the URLs in the HTML links were automatically replaced to request the document through our proxy. As the requested documents were returned through our proxy, JavaScript logging code was injected into the document, as described in Section 3.3.

5.2 Browsing Behavior Dataset

A total of 109 MTurk participants finished their tasks. After filtering out the users who did not follow the game rules, we obtained 1175 search sessions, performed by 98 users. Our data for these users consists of 3,294 queries, 1,598 unique queries, and 2,997 SERP clicks on 662 distinct URLs. Of these, the document behavioral data was collected for 2,289 page visits (76%) and 508 distinct URLs, comprising the final dataset for our experiments described below[3]. For each page view there were on average 400 atomic browsing events (mouse movements, scrolling, key pressing) on average. Note that a document might be visited from different queries, and for each query-URL pair, the snippet generation algorithm would produce a snippet independently. So the comparative experiments for snippet quality evaluation were performed on a set of 707 different query-URL pairs.

The set of the documents with collected behavior data was divided randomly into equal-sized training and test set, stratified by the document URLs to ensure no snippets derived from the same document to be included in both the training and test set. The training set was used to train the regression algorithm for predicting fragment interestingness, and the test set was used to evaluate the generated snippets.

5.3 Text Fragment Quality Judgments

In order to train the text-based baseline snippet generation algorithm (Section 4.1), we require a set of labeled text fragments. For this, we collected 949 text fragment quality judgments through the Amazon MTurk service. The assessors were asked to re-rank 10 text fragments randomly chosen by our fragment generator system to obtain a reliable training set. Each fragment was judged by 3 assessors. The fragments used in for this data collection were generated from query-URL instances taken from our training set, and do not overlap with our test set. We specifically asked assessors to re-rank fragments to avoid potential inaccuracies caused by using an absolute scale. In order to train the fragment scorer, we performed rank aggregation by computing the average rank of the fragment among the rankings produced by different judges.

6. RESULTS

We now present the main results of our study. First, we report the intermediate result of using the behavior data to infer the interesting (useful) fragments in the document. Then, we report the main results of the paper where the quality of the generated snippets, with and without using behavior data, compared using human judgments (Section 6.2).

6.1 Prediction of Fragment Interestingness

This experiment evaluates how well we can predict interesting fragments by observing the user's document examination behavior. We define the fragment to be interesting if it is related to the answer for the question. For each visited page we collect the user's answer (if submitted), and all the correct answers from all the users who answered this question. Then we automatically compare those answers to each text fragment in the document.

[3]For the remaining 24% of the page visits, the behavioral data was missing due to conflicts between our JavaScript tracking code and other JavaScript code already present on the page, and these documents were omitted from the dataset.

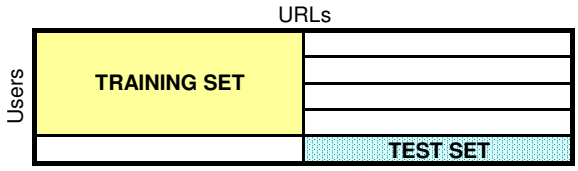

Figure 2: Illustration of the cross-validation setup: the training and test sets are disjoint by both users and URLs.

For this experiment, the document text is represented as a set of short text fragments, each consisting of five words. Those 5-word sequences are, on one hand, almost unique in a typical web document, and thus could be used as an identifier of a text position in a document, and on the other hand are short enough to match to local behavior patterns. For each fragment a set of behavior features is computed as described in section 4.3.

The cross-validation experiment set up as illustrated in Figure 2. The set of users and URLs are divided into a training set and a test set, so that the training set and the test set are disjoint for both sets of users and URLs. 10-fold cross-validation was performed, so that each user-URL pair appeared in a test set for some cross-validation split.

In the training set, a fragment's label is set to $label\,(fragment_i) = 1$ if the user submitted an answer in the current session, the answer is correct, and the answer has common words with the fragment. If the user submitted a correct answer, but the answer shares no words with a document fragment, then $label\,(fragment_i) = 0$. If the user did not submit an answer, or the answer is incorrect, we excluded the fragment from the training set. The answer and the fragment were compared after stemming and stopword removal. Similarly, in the test set we use for evaluation only those fragments that share words with the submitted search query.

The Gradient Boosting Regression Tree algorithm was trained on the training set of fragments, and applied to the test set. So each fragment in the test set receives a *fragment interestingness* $BScore(fragment_i) \in \mathbb{R}$.

We evaluate the interestingness of fragments by comparing the fragment's text with user's answer (if it exists), and to all the correct answers submitted by all users for the same question. We use the standard ROUGE-1 and ROUGE-2 metrics [32] for evaluation of the fragment intersection with answers, as these metrics are commonly used for evaluation of automatic summarization and annotation algorithms.

ROUGE-N metric for a fragment and a set of answers A, is computed as the recall of the answer set covered by the fragment word N-grams:

$$\text{ROUGE-N}\,(fragment, A) = \frac{\sum_{a \in A} \sum_{gram_n \in fragment} \text{Count}_{match}(gram_n)}{\sum_{a \in A} \sum_{gram_n \in fragment} \text{Count}(gram_n)}$$

Figure 3 shows the relationship between the interestingness of a fragment and the behavior score. The graph shows that when the score is high (≥ 0.5), the average intersection between the fragment and user's answer is much higher than those when the fragment score is low. All ROUGE-N metrics increase when the behavior score increases, but the ROUGE-2 values over all correct answers are always very small (changing from 0.003 to 0.007). We note that

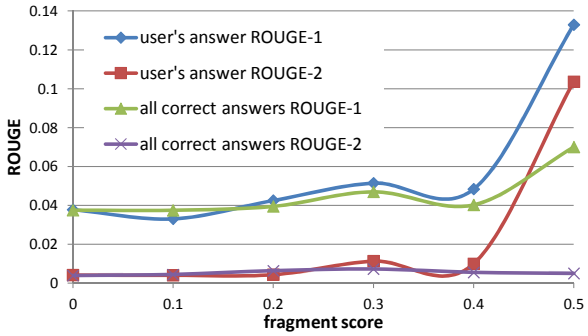

Figure 3: Fragment ROUGE-1 and ROUGE-2 values vs. behavior-based interestingness score $BScore$.

ROUGE-1 is much greater than ROUGE-2 for high scores, as the interesting fragment might contain useful information for the answer, but the user reformulates the obtained information and submits reformulated answer. The ROUGE-N values for a user's answer are much greater than those for all correct answers, as other users might obtain valuable information from other documents, and some questions have distinct correct answers.

This experiment shows that we can predict fragment interestingness by using behavior features only. In the next section, we apply the computed behavior scores for a practical task of improving search result summaries.

6.2 Snippet Quality Evaluation

Evaluation Setup: Our evaluation follows the snippet quality desiderata outlined in Section 3.1. Specifically, the snippet quality is evaluated by performing blind paired preference tests. For each query and URL, a pair of snippets produced by two different algorithms were evaluated by an assessor. A pair of snippets were presented on a page in random order, so the assessors did not know which algorithm produced which snippet.

Each assessor was asked to examine the web page, the search query, and the pair of snippets, and to answer three questions that correspond to our snippet quality criteria:

- Which of the snippets better summarize the parts the web page relevant to the search query? You need to read the web page before answering this question.
- Which of the snippets is written better – and is more readable?
- Imagine that you have the following search intent: "*question*". Which snippet helps you to identify relevant content better, and helps you decide whether to click on this result or not? You must consult the list of the correct answers for this question before answering.

For each question there were three possible answers: "Snippet 1 is better", "Both snippets are similar for this criterion", and "Snippet 2 is better".

We hired 14 pre-qualified Amazon MTurk workers, who have previously shown accurate results and high agreement with our "gold standard" subset of snippets labeled by the authors. As an additional test, we also included a small portion of exactly the same snippets to check the quality of MTurk workers, to verify that for the *same* snippets the worker submitted "Both snippets are similar" label for each criterion. As a result, we collected pairwise preference

Feature Group	NDCG@10
Single Feature Group	
Metzler-Kanungo	0.719
Relevance	0.741
Readability	**0.743**
QueryMatch	0.719
All Except One Feature Group	
All-Metzler-Kanungo	0.725
All-Relevance	0.736
All-Readability	**0.743**
All-QueryMatch	0.717
All	**0.764**

Table 3: Feature ablation results for fragment text scoring (10-fold cross validation)

labels for 2959 snippet pairs, 8525 atomic judgements (three judgements for each snippet pair, excluding "no answer" responses). The total amount paid to the MTurk workers was $217. One half of the obtained judgements were used for development and debugging purposes, and the other half were used for testing and reporting the results in the next section.

Evaluation Metrics: As a main evaluation metric we use the fraction of labels that give *preference* to the BeBS system, compared to the baseline. The preference ratio metric is evaluated for each criterion in (judgeability, readability, representativeness). The reason is that we found that the snippet quality criteria are difficult for assessors to judge absolutely, but can be easily compared as relative preferences.

6.3 Analysis of the Text-based Baseline

Before studying the benefits of behavior-based snippet generation, we optimized the text-based baseline method (Section 4.1) by varying the combination of features used. The results of the feature ablation experiments are reported in Table 3. As this table shows, using *all* of the text-based features achieves highest judge preference for the resulting snippets, thus, we use all of the features described in Table 1 for the baseline system performance in subsequent experiments.

6.4 Evaluation of the BeBS System

This experiment compares the text-based baseline with our BeBS system. Recall, that the λ parameter (the relative weight of the behavior score) affects two characteristics of the algorithm: coverage and snippet quality, as described in Section 4.4. To explore the resulting trade-offs, Figure 4 reports the judgeability, readability, representativeness, and coverage for five values of λ. The binomial distribution two-sided statistical significance test with confidence level 0.9 was computed, and the corresponding confidence intervals are presented on the graph. The graph shows that setting of $\lambda = 0.7$ provides significant improvement on all three snippet quality metrics. For this value, the coverage is 40%, which means that the behavior features provide improvement in representativeness for $0.4 * 0.68 = 27\%$ of all snippets, and produce worse snippets for $0.4 * (1 - 0.68) = 13\%$ of all snippets. When $\lambda = 0.5$, judgeability and readability also improve, but the improvement in representativeness is small and not statistically significant. When $\lambda = 0.9$, coverage grows up to 53%, but results in more noise and degrades snippet quality. When λ is set to a low value, the coverage drops, and we have too few modified snippets from the baseline to observe statistically significant differences in snippet quality. The graph shows that for $\lambda \in \{0.1, 0.3\}$, the confidence intervals cross $y = 0.5$ axis, and this means that the difference in snippet quality is not statistically significant. The Table 4 reports the detailed assessment data for the two best runs.

Baseline vs. behavior with $\lambda = 0.7$			
	Judg.	Read.	Repr.
baseline is better	44	35	34
similar	23	27	29
behavior is better	67	71	71
no answer	0	1	0
ratio of improved	0.60^*	0.67^*	0.68^*
p-value	0.018	0.0003	0.0002

Baseline vs. behavior with $\lambda = 0.5$			
	Judg.	Read.	Repr.
baseline is better	108	107	122
similar	160	149	207
behavior is better	162	142	140
no answer	41	73	2
ratio of improved	0.60^*	0.57^*	0.53
p-value	0.0006	0.0015	0.14

Table 4: Pairwise preference tests for snippets with behavior features added, number of judgements

Finally, we show two examples that demonstrate how incorporating behavior features can affect the resulting snippets. The first example (Figure 5, left) compares two snippets, the first produced by the baseline algorithm (top), and the second by using BeBS (bottom), for the query "sports invented in australia", corresponding to the question "What sports did Britain get from Australia?". The example shows that the snippet produced by BeBS is substantially more useful than the baseline, in clearly indicating to the searcher that the answer is present in the document. Note that this document is a difficult case for traditional snippet generation approaches: the landing page contains 8 matches for the query word "sport", and more than 50 matches for each of words "invent", and "Australia". Furthermore, the relevant fragment is located near the bottom of the document (three scrolling screens down), yet it is included into the snippet because several users scrolled near the end of the page and inspected the text near this fragment thoroughly, resulting in a high behavioral score.

The second example shows a case where the BeBS produced an inferior snippet compared to the baseline. The searcher issued a query "metals less dense than water", and the baseline algorithm produced a good snippet that contains the correct answer "Potassium and Lithium", relevant to both search query and search intent. But many users did not found the answer on the landing page, and instead examined the attractive section of the page corresponding to "Popular Searches", with the list of suggested queries. That resulted in a high behavior scores for that fragment, and consequently BeBS produced a snippet with poorly readable and irrelevant text. Additional behavior data, and further tuning of the behavior score prediction may alleviate these errors, as we plan to explore in future work.

6.4.1 Behavior Feature Importance Analysis

To estimate relative importance of behavior features for snippet generation, we analyzed the Gini importance index [5] for each behavior feature from the table 2. The table 5 shows that the most important features are *DispMid-*

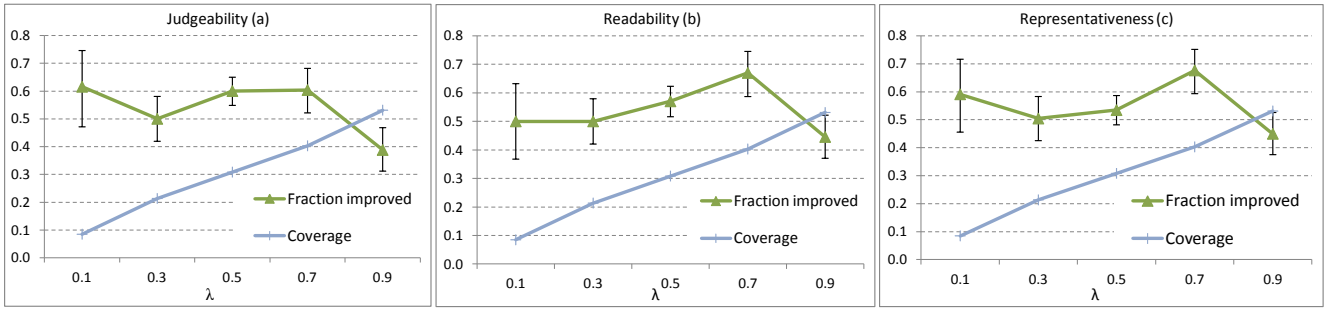

Figure 4: Snippet judgeability (a), readability (b), and representativeness (c) vs. coverage for varying values of λ.

Query: **sports invented in australia**
Question/Intent: **What sports did Britain get from Australia?**

List of **Australian** inventions - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/List_of_**Australian_inventions**
Australian **inventions** consisting of products and technology **invented** in **Australia** from pre-European-settlement in 1788 to the ... is used in **sports** broadcasts and provides viewers with spectacular views of events such as motor racing, which are impossible

List of **Australian** inventions - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/List_of_**Australian_inventions**
Australian **inventions** consisting of products and technology **invented** in **Australia** from pre-European-settlement in 1788 to the ... Vale near London, Mr. and Mrs. Edward Hirst of Sydney **invented** the combination polo and lacrosse **sport** which was first played

Query: **metals less dense than water**
Question/Intent: **Which metals float on water?**

Two **Metals** More **Dense Than** Mercury - Ask Jeeves
uk.ask.com/beauty/Two-**Metals**-More-**Dense**-**Than**-Mercury
What two **metals** are **less dense** than mercury Potassium and Lithium. ... are **metals** more **dense** than non- **metals Metals** have a tightly packed crystal lattice ... **Water Metals** Used to Make Pewter What Are the Ingredients in Solder Elements to Make

Two **Metals** More **Dense Than** Mercury - Ask Jeeves
uk.ask.com/beauty/Two-**Metals**-More-**Dense**-**Than**-Mercury
Densest Known Solid Nonmetallic Element Densest Known Solid Metallic Element **Density** of **Water Metals** Used to Make Pewter What Are the Ingredients in Solder Elements to Make Brass About Privacy Policy Partner Programme 2012 IAC Search & Media

Figure 5: Comparison of snippets produced by a baseline algorithm (top), and by the BeBS system (bottom) for two different queries. The relevant snippet parts are highlighted in yellow.

Feature	Gini coefficient
DispMiddleTime	0.51
MouseOverTime	0.34
DisplayTime	0.12
MouseNearTime	0.02
MouseOverEvents	0.01
MouseNearEvents	0.01

Table 5: Feature importance for behavioral features

dleTime - the time duration when the text fragment was visible in the middle of the browser window, and *MouseOverTime*, the time duration when the mouse cursor hovered over the text fragment. While the first feature has been previously shown [11] to be beneficial for re-ranking search results, we are encouraged to find it to be also helpful for snippet generation. The *MouseOverTime* feature has been shown to be correlated with user interest [18], thus confirming our hypothesis that searcher interest can be helpful for generating better snippets.

7. CONCLUSIONS AND FUTURE WORK

We presented a novel way to generate *behavior-biased* search snippets that privilege the document fragments examined by the users. To our knowledge, our work is the first to incorporate search examination behavior data into snippet generation result summary generation. To accomplish this, we developed a robust methodology to acquire precise document examination data at scale using mouse cursor movement and scrolling, and to associate these interactions with the fragments of the underlying document. Our results show significant quantitative improvements over a strong text-based snippet generation baseline. We complement the results with qualitative observations about the cases where examination behavior can be helpful or harmful to the generated snippets. Our code is based on the freely available EMU plug-in for the Firefox browser [17], and the search behavior data used for experiments is shared with the research community to encourage further work in this area.

While our work makes significant advances to search snippet generation, it also opens many directions for future improvements and extensions. While our method was primarily targeted (and evaluated for) informational queries, an important consideration is how to generalize our approach to a wider class of queries. For other query classes (e.g., navigational), snippets might be optimized using a different criteria, but snippet generation could still take advantage of the search examination data. We have also assumed that document visits can be grouped by query intent. A number of methods have been proposed to cluster queries (and result clicks) by intent (e.g., [34, 30, 38]), and one future direction is to extend our method to work with automated query clustering techniques, which might introduce additional noise. Another key assumption is that user interactions on landing pages can be collected by a search engine. As we pointed out earlier, commercially deployed methods already exist to collect interaction data on landing pages, for applications such as advertising. While addressing the privacy issues inherent in search behavior logging is beyond the scope of this paper, research in privacy-preserving data mining [2]

could potentially alleviate these concerns. Furthermore, as efficient machine learning methods and computational capabilities of personal and mobile devices continue to advance, mining searcher interactions could be performed directly on the user's device, only sharing the less detailed predictions with the search engine. Finally, our approach is not necessarily limited to desktop-based computers with a mouse. Modeling interactions on mobile and touch-enabled devices also allows to infer searcher interest and attention [20], and incorporating this richer searcher behavior data into result summary generation is another promising future research direction.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation grant IIS-1018321, the DARPA grant D11AP00269, and by the Russian Foundation for Basic Research Grant 12-07-31225.

8. REFERENCES

[1] M. Ageev, Q. Guo, D. Lagun, and E. Agichtein. Find it if you can: A game for modeling different types of web search success using interaction data. In *Proc. of SIGIR*, 2011.

[2] C. C. Aggarwal and S. Y. Philip. *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

[3] S. Bird. Nltk: the natural language toolkit. In *Proc. of the COLING/ACL.*, pages 69–72, 2006.

[4] S. Blair-Goldensohn and K. McKeown. Integrating rhetorical-semantic relation models for query-focused summarization. In *Proc. of DUC*, 2006.

[5] L. Breiman and L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[6] E. Brill, S. Dumais, and M. Banko. An analysis of the askmsr question-answering system. In *Proc. of ACL*, EMNLP '02, pages 257–264, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[7] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, Sept. 2002.

[8] G. Buscher, E. Cutrell, and M. R. Morris. What do you see when you're surfing?: using eye tracking to predict salient regions of web pages. In *Proc. of CHI*, CHI '09, pages 21–30. ACM, 2009.

[9] G. Buscher, A. Dengel, and L. van Elst. Query expansion using gaze-based feedback on the subdocument level. In *Proc. of SIGIR*, 2008.

[10] G. Buscher, S. T. Dumais, and E. Cutrell. The good, the bad, and the random: an eye-tracking study of ad quality in web search. In *Proc. of ACM SIGIR*, SIGIR '10, pages 42–49, New York, NY, USA, 2010. ACM.

[11] G. Buscher, L. van Elst, and A. Dengel. Segment-level display time as implicit feedback: a comparison to eye tracking. In *Proc. of SIGIR*, 2009.

[12] H. Dang, D. Kelly, and J. Lin. Overview of the trec 2007 question answering track. In *Proc. of TREC-2007*, 2007.

[13] H. Daumé III and D. Marcu. Bayesian query-focused summarization. In *Proc. of ACL*, 2006.

[14] S. Fisher and B. Roark. Query-focused summarization by supervised sentence ranking and skewed word distributions. In *Proc. of DUC*, 2006.

[15] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):pp. 1189–1232, 2001.

[16] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing text documents: sentence selection and evaluation metrics. In *Proc. of SIGIR*, pages 121–128. ACM, 1999.

[17] Q. Guo and E. Agichtein. Exploring mouse movements for inferring query intent. In *Proc. of SIGIR*, 2008.

[18] Q. Guo and E. Agichtein. Towards predicting web searcher gaze position from mouse movements. In *Proc. of CHI*. ACM, 2010.

[19] Q. Guo and E. Agichtein. Beyond dwell time: estimating document relevance from cursor movements and other post-click searcher behavior. In *Proc. of WWW*, 2012.

[20] Q. Guo, H. Jin, D. Lagun, S. Yuan, and E. Agichtein. Mining touch interaction data on mobile devices to predict web search result relevance. In *Proc. of SIGIR*, 2013.

[21] S. Harabagiu and F. Lacatusu. Using topic themes for multi-document summarization. *ACM Trans. Inf. Syst.*, 28(3):13:1–13:47, July 2010.

[22] Y. Hijikata. Implicit user profiling for on demand relevance feedback. In *Proc. of IUI*, pages 198–205. ACM, 2004.

[23] J. Huang, R. White, and G. Buscher. User see, user point: gaze and cursor alignment in web search. In *Proc. of CHI*, pages 1341–1350, New York, NY, USA, 2012. ACM.

[24] J. Huang, R. W. White, G. Buscher, and K. Wang. Improving searcher models using mouse cursor activity. In *Proc. of SIGIR*, 2012.

[25] J. Huang, R. W. White, and S. Dumais. No clicks, no problem: using cursor movements to understand and improve search. In *Proc. of CHI*. ACM, 2011.

[26] T. Kanungo, N. Ghamrawi, K. Y. Kim, and L. Wai. Web search result summarization: title selection algorithms and user satisfaction. In *Proc. of CIKM*, 2009.

[27] T. Kanungo and D. Orr. Predicting the readability of short web summaries. In *Proc. of WSDM*, 2009.

[28] T. Kiss and J. Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[29] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proc. of SIGIR*, SIGIR '95, pages 68–73, New York, NY, USA, 1995. ACM.

[30] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *Proc. of SIGIR*, 2008.

[31] S. Liang, S. Devlin, and J. Tait. Evaluating web search result summaries. *Advances in Information Retrieval*, pages 96–106, 2006.

[32] C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[33] D. Metzler and T. Kanungo. Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*, 2008.

[34] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *Proc. of WWW*, 2010.

[35] K. Rodden, X. Fu, A. Aula, and I. Spiro. Eye-mouse coordination patterns on web search results pages. In *Proc. of CHI*, 2008.

[36] H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *Proc. of the 12th Conf. of the European Chapter of the ACL*, pages 781–789, 2009.

[37] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proc. of SIGIR*, pages 2–10. ACM, 1998.

[38] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proc. of WWW*, 2001.

[39] R. W. White and G. Buscher. Text selections as implicit relevance feedback. In *Proc. of SIGIR*, 2012.