Don't Just Swipe Left, Tell Me Why: Enhancing Gesture-based Feedback with Reason Bins

Juan Felipe Beltran[†] **Ziqi Huang**[o] **Azza Abouzied**[†] **Arnab Nandi**[o]
[†]New York University Abu Dhabi [o]The Ohio State University
{juanfelipe,azza}@cs.umass.edu {huangzi,arnab}@cse.osu.edu

ABSTRACT

Despite several advances in information retrieval systems and user interfaces, the specification of queries over text-based document collections remains a challenging problem. Query specification with keywords is a popular solution. However, given the widespread adoption of gesture-driven interfaces such as multitouch technologies in smartphones and tablets, the lack of a physical keyboard makes query specification with keywords inconvenient. We present BINGO, a novel gestural approach to querying text databases that allows users to refine their queries using a swipe gesture to either "like" or "dislike" candidate documents as well as express the reasons they like or dislike a document by swiping through automatically generated "reason bins". Such reasons refine a user's query with additional keywords. We present an online and efficient bin generation algorithm that presents reason bins at gesture articulation. We motivate and describe BINGO's unique interface design choices. Based on our analysis and user studies, we demonstrate that query specification by swiping through reason bins is easy and expressive.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords

Interactive Search Refinement; Gesture-based Feedback; Reason Bins

INTRODUCTION

The widespread adoption[1] of portable, keyboard-less computational devices such as multitouch-based tablets and smartphones and the constantly evolving mobile-app ecosystem[2] are increasing consumer's expectations on the accessibility of sophisticated and powerful applications, such as text exploration and querying apps, from the palms of their hands.

[1]In the US, tablet and smartphone ownership is on a sharp rise, while desktop and laptop ownership is slightly declining [5].

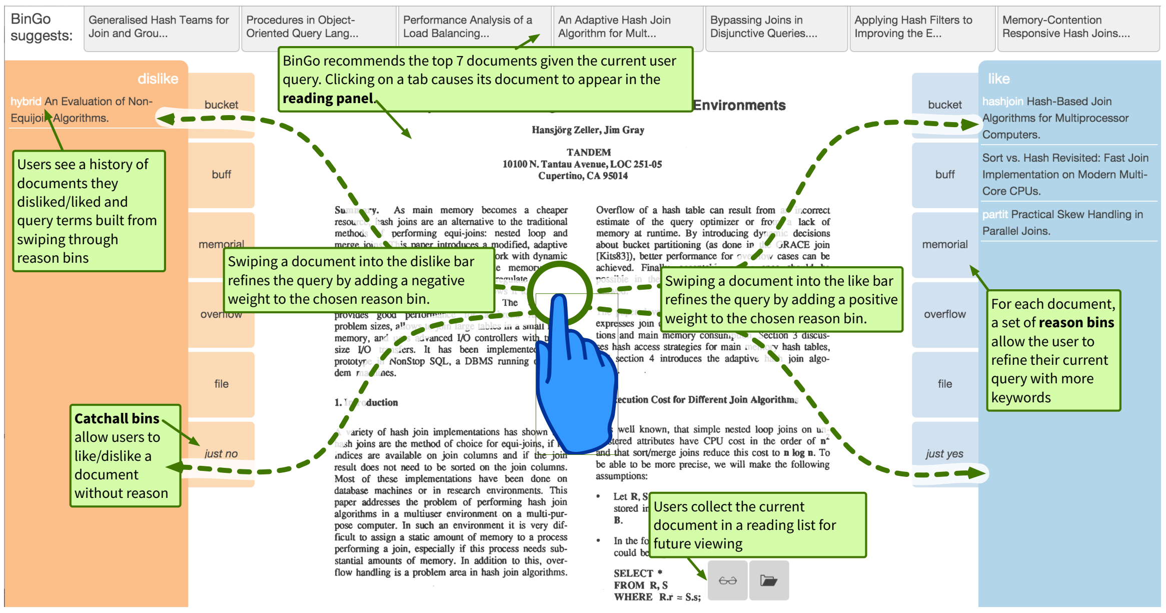
[2]Moreover, more Americans are accessing online services through mobile apps than through desktop web browsers [39].

IUI 2017, March 13-16, 2017, Limassol, Cyprus
© 2017 ACM. ISBN 978-1-4503-4348-0/17/03...$15.00
DOI: http://dx.doi.org/10.1145/3025171.3025212

The prevalent form of query specification over document collections, regardless of the underlying document retrieval model (vector space models with cosine similarity measures, query-likelihood models, Pagerank, etc.) remains keyword specification. While users have adjusted to keyword search on keyboard-devices, the absence of physical keyboards on tablets and smartphones makes keyword search a suboptimal user experience. Character or word-level *text entry* [38] mechanisms such as Dasher [61] are popular in accessibility contexts. The entry of text, however, using either virtual keyboards or these methods can be cumbersome. Typing with onscreen keyboards is tedious and users prefer simpler *gestural* interfaces: the success of mobile apps – such as Clear Todo List [48] – a gesture-driven todo list app that uses a rich-but-intuitive set of gestures to perform a variety of commands, and Tinder [1] – a dating app which lets users swipe photos of potential matches left if they dislike the photo or right if they like it – highlight the user demand to re-design interfaces that are more gesture-driven and less keyboard-driven.

In the context of multitouch and gesture-driven interfaces, gestural input is typically faster and more convenient than other modes of interaction. Since gestures can be performed on any part of the screen, there is no strict requirement to initiate the gesture on a specific location, unlike keyboards, which require the user to carefully locate and press a specific button. Gestural input also reduces the interaction to a smooth fluid motion, as opposed to a multitude of tapping actions with a virtual keyboard. Moreover, the options for user input can be dependent on the application state: different buttons can be shown on the screen depending on the currently valid options for user input. As shown by prior work in gestural query specification [43, 29], it is possible to have a rich, easy-to-use language of intuitive gestures for querying relational data.

Consider a "document review" use case, where the user is constructing a bibliography by sifting through a collection of research papers. In Figure 1, we depict an example prototype user interface, inspired by modern "card" UIs in mobile applications such as Google Now [62] and Tinder [1], where the document collection presented is the result of a recommendation algorithm. The swipe gesture allows users to provide input and feedback on the displayed document. For example, the user may notice that the paper in Figure 1 is a good addition to her bibliography, and hence "like" that paper by swiping right, with the intent of surfacing similar papers.


BinGo suggests:
Generalised Hash Teams for Join and Grou...
Procedures in Object-Oriented Query Lang...
Performance Analysis of a Load Balancing...
An Adaptive Hash Join Algorithm for Mult...
Bypassing Joins in Disjunctive Queries....
Applying Hash Filters to Improving the E...
Memory-Contention Responsive Hash Joins....
dislike
hybrid An Evaluation of Non-Equijoin Algorithms.
bucket
buff
memorial
overflow
file
just no
like
hashjoin Hash-Based Join Algorithms for Multiprocessor Computers.
Sort vs. Hash Revisited: Fast Join Implementation on Modern Multi-Core CPUs.
partit Practical Skew Handling in Parallel Joins.
just yes
Environments
Hansjörg Zeller, Jim Gray
TANDEM
10100 N. Tantau Avenue, LOC 251-05
Cupertino, CA 95014

BinGo recommends the top 7 documents given the current user query. Clicking on a tab causes its document to appear in the **reading panel.**

Users see a history of documents they disliked/liked and query terms built from swiping through reason bins

Swiping a document into the dislike bar refines the query by adding a negative weight to the chosen reason bin.

Swiping a document into the like bar refines the query by adding a positive weight to the chosen reason bin.

For each document, a set of **reason bins** allow the user to refine their current query with more keywords

Catchall bins allow users to like/dislike a document without reason

Users collect the current document in a reading list for future viewing

Figure 1. The BINGO system. The left / right swipe gestures on a document indicate "dislike" / "like" actions. The system also allow users to specify their reason for "dislike"/"like" the document by swiping through one of the reason bins that the system provide.

However, this gestural input is only a binary input, and can be considered too vague in terms of feedback: *Was the paper liked because it addressed a certain topic? If so, which?* Thus, a simple gestural querying interface can be too primitive and may restrict the user's ability to articulate precise queries over the underlying text database.

We solve this problem in BINGO by enabling users to perform richer gestures as shown in Figure 1. BINGO generates and presents *reason bins* to the user *during* the swipe action. By enriching the swipe gesture, the user can now fluidly articulate *why* she likes or dislikes a document in the *same* gesture as the binary decision. As before, the user simply swipes left or right, but now through a reason bin. Thus, BINGO provides users with more control over their query with almost no extra gestural movements or distractions.

BINGO's unique gestural interface not only allows users to easily specify their queries without keyboards, it also provides an avenue for inexperienced users to perform effective searches in an unfamiliar domain. As the user decides on the relevance of a document to her search query, BINGO surfaces the salient terms of the document with respect to the entire collection with the help of the reason bins. As the user swipes through reason bins, BINGO refines a running weighted-keyword search query. Disliked bins add negatively-weighted keywords to the query and liked bins add positively-weighted keywords to the query. This contrasts with manual keyword or text query specification, which require the user to be familiar with the underlying document collection's vocabulary [12] or at least learn new keywords from displayed documents. BINGO's *bin-generation* algorithm is fine-tuned to perform well given the document collection and the retrieval model. The algorithm also ensures that reason bins only contain keywords that lead to query refinement avoiding wasted feedback cycles. For a user with little awareness of the data and the model, swiping through reason bins can be more effective than manual query construction with keywords.

In this paper, we briefly describe the rich body of work in information retrieval and search interfaces that BINGO builds on. We explain our interface design choices for BINGO and its inner workings. We then evaluate BINGO efficacy for three search scenarios: targeted search, surveys and exploration. We begin by describing these motivating search scenarios and their design implications.

MOTIVATING USE CASES

The following examples of use explain the nuances of searching text-databases on touch interfaces. Note that our work specifically addresses domain-specific or niche text-databases and not general web search.

Illustrative Example 1 – The first-year grad. Consider a first-year graduate student, with little or no familiarity with database research, conducting a literature review of spatial indexing techniques. A seasoned database researcher can quickly construct a keyword query with the following terms: 'spatial', 'index', 'rtree', 'Guttman'. However, the first-year graduate student formulates a less refined query: 'quickly access geographic data'. Depending on the underlying document retrieval model, the student may receive documents that are irrelevant. A quick search on Google scholar limited to only VLDB documents returns *'Semantic access: semantic interface for querying databases'* by Rishe et al. and *'PNUTS: Yahoo!'s hosted data serving platform'* by Cooper et al. as its top hits. While the remaining documents seem relevant in that they deal with spatial data, the student has to read several abstracts before arriving at more refined keywords such as 'spatial', 'index', 'rtree'. Even domain experts struggle when trying to construct a keyword query with the goal of reaching a specific document. We have all run into *tip-of-the-tongue* situations where we partially recall certain aspects of a paper but not enough to effectively retrieve it and spend minutes to hours searching through documents that are relevant to our search keywords but not exactly what we are looking for.

The above use case illustrates that for *targeted search*, where users search for a specific document, and *surveys*, where users collate a collection of documents relevant to a specific search topic, the query specification interface must:

1. *support an iterative query specification and revision process where users can easily refine their search queries as they observe more results.* BINGO allows for efficient revisions as it automatically refines the query as users like or dislike documents with or without reasons. Relying on swipes as the main mode of user query specification minimizes the time of each revision.

2. *surface the underlying vocabulary of the collection and keywords that both distinguish the current document from the collection and connect it to other documents.* BINGO suggests reason bins with terms most salient to the document allowing for keyword discovery by novices. It also suggests bins with terms that have a high *exploration potential* — terms that retrieve several documents. By modeling search as a graph traversal problem in our evaluation section, we demonstrate that reason bins connect documents in collection to form a well-connected graph with a small diameter. Thus by swiping through a few bins, one can easily reach any document within the collection even from a random starting document.

Illustrative Example 2 – The game explorer. Consider an avid gamer, stuck on a ten-hour train ride. He wishes to explore new games to play on his iPad. A typical search interface such as the one provided by giantbomb.com limits the user's interaction and hinders exploration. After providing a search keyword, the user scrolls through lists of relevant results that each provide little information. Each result needs to be clicked through and examined for content. Moreover, the user cannot eliminate irrelevant material without re-issuing a new keyword query. The process of entering new keywords on a screen is tedious and even after refining his query, the gamer still has to scroll through previously seen irrelevant material.

The above use case illustrates that for *exploration* tasks, where users do not have a precise notion of which documents are relevant or irrelevant, the query specification interface needs to:

3. *support changes in a vaguely-defined query over time.* Like most relevance feedback system, BINGO can construct a search query without explicit keywords as users swipe left or right without providing reasons for their choices. As users swipe through reason bins, new keywords are added to the query with little need for coherence among the keywords of a query so far.

4. *surface interesting search directions.* BINGO prevents the current query from overwhelming the search direction by eliminating its keywords from consideration for bins. This allows less salient terms within a document that introduce new exploration directions to be surfaced as reason bins.

RELATED WORK

BINGO builds upon prior work in gestural query interfaces by allowing the user to express *why* they like or dislike a candidate document within the same swipe gesture. It also builds upon work in document recommendation systems, specifically for the generation of reason bins.

Query Reformulation and Autocompletion

Information retrieval is typically an iterative and interactive process [35, 13]. Soo et al. [56] posit that users often reformulate queries to better express their information need for a variety of reasons, focused on the content, format, or resource representations of the query. Efforts toward assisting users towards this reformulation step include *"Did you mean?"* features in popular search engines such as Google, or *Refine Your Search* in Altavista. In the absence of keyboards, reason bins allow the user to provide granular content feedback to the system, allowing it to better understand their information need.

In the context of text-based queries, having the user pick from a selection of suggested terms [46] has been observed to be preferred over automatic query expansion [56]. Search suggestions for library portals have been developed for several decades [34], and open-domain query recommendation systems for Web search have been developed over the past decade [6, 26]. Most of these systems rely primarily on prior query logs and click traffic [19, 24, 8]. Autocompletion efforts in database systems [33, 42] can additionally leverage the structure of query and the schema and data in the database. Thus, for such settings – where the user cannot express the query, or is unaware of the data or schema of the dataset [15], prompting the user with suggestions is extremely valuable.

While query suggestion mechanisms can improve retrieval performance, users might also be distracted by the query suggestions [3]. Thus, *how* to show the suggestion terms should also be factored in to the design of the user interface. From a touch-focused interface perspective, providing the user with a means to reformulate or augment their query while still maintaining the like / dislike functionality was a critical consideration. In contrast to the "hard search" problem, BINGO blurs the line between query recommendations [59] and result exploration [40], while providing a convenient and usable interaction method for gestural interfaces. Our goal is to enable users to go beyond binary decisions in swipe interfaces: reason bins ameliorate the tediousness of earlier relevance feedback / query reformulation systems that required users to classify multiple documents or select from lists of keywords. We believe that our results do reveal a sweet spot between keyword only and pure relevance feedback. Even with exploratory search, there were users who preferred BinGo as it led them to new and unexpected games. As more of our searches are conducted on smartphones and handhelds, BinGo reintroduces relevance feedback with an effective gestural user interface that minimizes keyword entry yet still constructs useful search query terms.

Faceted Search and Exploration

In the context of exploration of documents, the use of faceted search [60] blends search and browsing [28, 37], wherein a user navigates through a document collection based on filters and attributes, is a popular one. Systems such as *Querium* [20] have investigated the use of faceted search in a collaborative setting, combining it with relevance feedback, query fusion, and a session-based model for search histories. The generation of facets in such systems is typically done by presenting the most popular drill-downs to the user [51, 32]. By modeling the query session as a multi-way search tree, it is also possible to minimize the number of decisions to be made [11], or to use a cost-based approach [57]. We view the recommendation of facets to explore as analogous to our bin generation problem, and draw inspiration from approaches in faceted exploration.

Recommendation Algorithms

Prior research in text mining, classifiers and recommender systems [49, 58, 9, 31, 17, 7] has exhaustively investigated the paper discovery algorithms themselves. We draw from concepts in feature selection and decision-trees [47, 53], but focus not on the problem of generating better paper suggestions, but address a deficiency of the querying process: the gestural input is *too* primitive, and thus the user is not able to express *why* they liked / disliked a document. In BINGO, we attempt to glean more information from the user about the query intent while aiming for the same amount of user effort as a binary classification task.

Visual Querying

There is a wide body of work for visual search [63], visual query [16] and visual information seeking [4] interfaces, both in the desktop [64, 10] and mobile contexts [25]. These systems focus on the use of rapid filtering to iteratively refine search parameters, and thus reduce the result set. The role of guidance [21] is critical in these contexts and has considered the use of prior query logs [30, 65, 27] and visual / cognitive aspects [41, 35, 22].

Casual and Touch-based Interfaces

In addition to the use of visual querying, the use of gestures and touch interfaces for search [36] is heavily motivated by the low effort required [55] from the users to provide input (which for our system is a single swipe). Selecting reason bins are an example of a crossing-based [2] interaction, which have been shown to be at least as fast as target-pointing interaction. A driving principle for our system design is the minimization of user effort, similar to other casual interfaces: Li et al. [36] showcase searching using phone screen for contacts, maps or apps by tracing the letter over the screen, while Schulze et al. [55] focus on context-aware search for nearby locations and how to better visualize and interact with map results. TouchViz [23] explores the use of tablets for analytics, and introduces a *FLUID* interaction paradigm, enabling touch actions on the data visualization itself. ScopeG [18] provides a mobile-friendly interface towards exploring and comparing data; in contrast, BINGO focuses on one document at a time. Kinetica [52] further introduces multitouch gestures to explore multivariate data – such gestures can be incorporated into BINGO in the future to provide more nuanced feedback, such as strong or weak likes / dislikes.

OVERVIEW OF THE BINGO SYSTEM

Operationally, BINGO is a simple system: a user initiates a search for relevant documents by providing a *seed* keyword[3]. Using one of many standard document retrieval models, BINGO returns the most relevant document given the seed. BINGO differs from standard search interfaces in what follows.

If the user finds the document relevant, she swipes the document to the right (see Figure 1). As the user articulates her swipe-right gesture, a set of (five) reason bins, along with a *catchall* bin, appear at the right end of the screen. Each bin contains a highly salient term in the current document. If the user swipes the document through one of the bins, she indicates to the system that she not only finds the document relevant but also that she wants to refine or extend her search query to contain the swiped bin's term. If the user finds none of the bins as useful refinements to her query, she swipes through the catchall bin instead. BINGO will never present these bins to the user again. Similarly, the user can indicate that a document is irrelevant by swiping to the left: BINGO generates reason bins on the left end of the screen and refines the query according to the user's chosen bin. If the user does not swipe through a bin, BINGO assumes that the user swiped through the default catch-all bin. At each iteration, BINGO constructs a new query, ranks all documents according to relevance and suggests the top (seven) documents that have not been evaluated by the user. The user can save documents into a reading list for later viewing or save the search keywords used for further future searches.

Thus, BINGO consists of three main components: (i) a back-end text database that supports document retrieval and ranking given a query of words, (ii) a gesture-driven query interface that allows users to construct queries by swiping documents through *reason bins* and (iii) an efficient *bin generation* component that constructs reason bins given past user bin choices and the document the user is currently viewing. In this section, we describe each component starting with BINGO's interface.

The gesture-driven interface

BINGO is implemented as a web interface. It represents each document in the central pane of the screen (See Figure 1). This allows users to quickly grasp the content of the top-ranked documents without clicking each result. The top seven documents are presented as clickable tabs at the top of the screen. The center document can be swiped left to indicate irrelevance or right to indicate relevance. A history of past decisions is preserved on the side dislike/like bars. Swiping gestures trigger the appearance of five reason bins and a catchall bin: the user has to swipe the document through one of these bins. Swiping a document through a reason bin extends the user's running query with the bin's positively weighted term if the user swiped the document into the like sidebar and the

[3]If a user does not seed the search, BINGO returns a document chosen at random from the database

bin's negatively weighted term if the user swiped the document into the dislike sidebar.

BINGO keeps track of which bins have already been selected, thus displaying fresh bins to the user at each iteration. Additionally, a swipe through the catchall 'just yes'/'just no' bins blacklists the reason bins, which were ignored by the user: these reason bins were not interesting enough to warrant a swipe through.

Interface Design Considerations

We briefly motivate certain interface design choices particular to BINGO:

1. *Bin presentation on swipe-gesture articulation.* Displaying reason bins on the side of the document panel at all times creates a visually-cluttered interface and can distract users. A clean, clutter-free reading panel allows users to quickly scan and decide whether the document is relevant or not. Only presenting bins on gesture articulation, thus, encourages users to first decide on the relevance of the document prior to refining their query with more terms. Thus, as the user swipes in favor of or against a document, BINGO presents reason bins.

2. *Single bin-selection* Only allowing single bin selection is an intentional design choice: It allows users to make quick decisions, while still supporting multiple granularities of decision-making (Yes/No or Yes/No because). Even if some search accuracy is traded-off[4] for faster exploration and search, BINGO is an improvement over pure relevance feedback models that have no support for reason specification.

3. *Bin swipe-through vs. tap.* Many gesture combinations can be used to clarify query intent. A user, for example, can tap the screen to reveal the reason bins and then tap again on a reason bin to select it. While taps are relatively common and easy gestures, we believe they are not as suitable as swipes for driving search. First, swipes are not worse than taps: a recent study by Neglescu et al. reveals no significant difference in reaction time or cognitive load between taps and swipes on smartphones in the absence of distractions [44]. Second, taps are common gestures that are frequently triggered by accident. Third, many apps adopt swipe right gestures for approval and swipe left for disapproval[5]: the swipe interaction *affords* an extension where users can extend their left or right swipe to also swipe through a reason bin.

4. *Bin size.* Swiping through a target is constrained by Fitts Law: the size of and the distance to the target does impact interaction time [14]. The bins are large targets to enable a swifter flow through them. They protrude into the reading panel to minimize the distance the finger needs to travel to hit one of them. Finally, as the user's finger approaches a bin, it turns a deeper shade to visually confirm its selection.

[4]One can argue that some search accuracy is lost if users are incapable of selecting multiple reasons for liking or disliking a document.

[5]e.g Tinder swipe left to reject, swipe right to like; Yahoo Mail: swipe left to delete mail, right to mark as read

5. *Bin placement.* Reason bins are placed on either side of the display to maintain the *external consistency* of the swipe interaction: it is standard convention to swipe screen objects left to indicate irrelevance and right to indicate relevance. The bins are placed as gates to the dislike/like bars to enable a smooth swipe through interaction.

6. *Scroll-context insensitive bins.* An earlier BINGO prototype dynamically generated bins depending on the scroll position of the document. Users, however, found the lack of bin permanence on scrolling confusing. Therefore, we trade bin diversity for usability and generate a small number of permanent bins ($k = 5$) for each document.

7. *Top-7 documents and top-5 bins.* Our choice of showing only top-7 documents and top-5 bins are motivated by studies on human memory capacity [66]. This is especially critical for reason bins: a full swipe gesture from the midpoint of a screen to either side takes a few milliseconds and in the course of the swipe, the user has to not only recognize the bins, but choose one to refine his/her query with. Moreover the relatively small screen sizes of keyboard-less devices allows little room for more than five to seven bins.

The backend text database

BINGO is built on a standard text database: a text document is represented as a bag of words. Documents are preprocessed to filter out stop words and to compute the term-frequency, inverse document frequency (tfidf) of each word within the document. Words that appear more often (high term-frequency) within a document are more representative of the document as long as they do not appear frequently within the document set (high inverse document frequency). Many retrieval models rely on tfidf to score and rank a document's relevance given a user query of search keywords. For each document, we sort and stores its words from high tfidf scores to low scores to enable efficient bin generation.

We also filter out words with poor *exploration potential*: words that appear in very few documents. Since the purpose of reason-bins is to extend (refine or relax) the current search query, terms are selected for bins only if they appear in θ or more documents[6]. Algorithm 1 describes the preprocessing of each document in the collection.

The bin generation algorithm

The bin generation algorithm is intentionally simple and easily integrates with existing document retrieval models. Given the document d under consideration, we select its $k = 5$ most salient terms as bins. In the vector space document retrieval model, these are terms with a high tfidf score — terms that appear often in the document but not as often in the collection of documents D and hence distinguish the document from others. The algorithm eliminates terms that have been previously selected as query terms or terms that have been previously presented as bins and ignored by the user, I, when determining the relevancy of another document.

[6]For our user tests, we set θ to be five documents. This also proved useful in eliminating garbled words that were unique to a document due to poor OCR. This parameter is easily configurable

Algorithm 1 Document Preprocessing

```
stop-words ← {is, are, and, ...}
connecting-words ← {words $w \in$ collection $D$ s.t. $w$ appears in at least $\theta = 5$ documents}

procedure PREPROCESS(document collection $D$)
    for each document $d \in D$ do
        word-score ← {}
        for each word $w \in d$.getwords() do
            if $w \in$ connecting-words - stop-words then
                $tf \leftarrow$ normalized frequency of $w$ in $d$
                $idf \leftarrow \log |D| - \log |d' \in D : w \in d'|$
                word-score.add($w$: $tf * idf$)
        $d$.candidate-bins ← sort(word-score, values,
                                  descending).keys()
```

Algorithm 2 Bin Generation Algorithm

```
$k \leftarrow 5$ (number of bins to generate)

procedure GENERATEBINS(document $d$, query $Q$, ignored bins $I$)
    bins ← $d$.candidate-bins - $Q$ - $I$
    return top(bins, $k$)
```

Bin generation needs to occur instantly in order to ensure an interactive query experience. In the interest of this property, the bulk of bin-generation — the elimination of stop words and words that do not appear in multiple documents, and the computation of tfidf scores and sorting of terms in a document by their tfidf scores — is done as a preprocessing step. Excluding preprocessing, online bin generation requires $O(|I| + |Q| + k)$ time. Thus, bin generation runs independently of the size of the current document d (number of words) , or collection D (number of documents).

Document ranking algorithm

For each document that the user reviews, BINGO automatically refines the query. By constantly and automatically refining the user query with each interaction, we not only can lead the user to a target document more efficiently, but we can also support collection exploration by presenting different documents as the query gradually changes. BINGO is merely a *query specification and refinement layer* over standard keyword-query document retrieval models.

In our current implementation, we support both the vector space model with cosine similarity where each word is weighted by its term-frequency, inverse document frequency [50] and the query likelihood model with Jelinek-Mercer smoothing when estimating the maximum likelihood of a word in a document[7]. We also support Rocchio relevance feedback to augment our search queries when users swipe documents through the catchall bins[8]. For a primer on document retrieval models, we refer the reader to [54].

[7]In our implementation, we found a smoothing factor of $\lambda = 0.9$ to perform well for our test datasets

EVALUATION

We evaluate BINGO, a novel search interface, for the following scenarios:

1. *targeted (tip-of-the-tongue)* search: users look for a target document in a database),
2. *survey* search: users retrieve all documents relevant to a specific search task, and
3. *exploration*: users have no explicit search task specification and are exploring the collection for documents that might be of personal interest.

To test the first scenario, we *simulate* user behavior over standard keyword search and BINGO and compare the search efficiency of both interfaces: a more efficient interface requires less interaction effort to reach the target document. We compute interaction effort as the number of *taps* required to add a keyword to a query or the number of *swipes*. We argue that our simulation and comparison framework provide a cost-effective means to rapidly test different search interfaces and different bin generation algorithms as well as simulate different user behaviors, while still providing representative test results. Note that we cannot force users into *tip-of-the-tongue* situations, which necessitates a simulated user-study.

We conduct user studies to compare user performance on both search interfaces for the remaining scenarios. We evaluate *precision* – proportion of documents a user added to his/her reading list to documents the user reviewed – and *revision time* – time between subsequent interactions such as swiping through a reason bin or typing a new keyword. These provide us with quantitative measures of BINGO's performance as a query specification tool in comparison to keyword specification (KWD) or pure relevance feedback with no reasons with simple swipes (SWP). We qualitatively evaluate BINGO's usability with Likert-test and open-ended questions[9].

In our evaluation, we focus on two domain-specific data sets[10]. Our choice of testing data sets is motivated by our need to find a pairing of data set and users, where users were somewhat well-versed in the data set's topic. The VLDB data set of database systems research papers and the GiantBomb games data set are easily accessible and together cover the wide range of applications that BINGO was designed for. Both data sets also had an accessible user base for testing: students of a graduate database class and users with some interest in video games.

[8]With Rocchio relevance feedback, we set $\alpha = 1$ for the primary query vector of user selected keywords/bins, $\beta = 0.75$ for words extracted from relevant documents and $\gamma = -0.25$ for words extracted from irrelevant documents

[9]One can also evaluate BINGO with cognitive load tests such as the Nasa-TLX test. In this paper we first-and-foremost focus on the efficacy of BINGO as a query specification interface and hence present precision and revision time results.

[10]We do not expect BINGO to be used as a tool for specifying general web search queries. Instead it is more appealing for limited data sets such as scholarly, legal articles, niche sites and even product databases provided they have descriptive text meta-data.

The underlying document retrieval model used for the following experiments is the vector space model with cosine similarity. Queries are represented as weighted keywords. We use the standard Rocchio model for pure relevance feedback without reason bins (SWP). Since BINGO is not a novel document retrieval model, but rather a gesture-driven query specification technique where generated queries are weighted keywords, we believe our results generalize to other document retrieval and feedback models that utilize keyword queries.

Targeted Search Tasks

Before we can describe our experimental setup, we must first explain how bins induce links over a collection of documents. Consider a graph where each document d is a node with an out-degree of at most k edges: an edge (d, d') for each of its k (positive) reason bins. Each reason bin acts as a single-term search query and forms an edge between d and the top ranked document d' returned by that query.

Ideally, this bin-induced graph has no disconnected components, has a small diameter and a small average shortest path between any two documents. A connected graph entails that regardless of which document a user starts in, eventually he/she will arrive at the required target document. The graph's diameter provides a rough measure of the worst case number of swipes a user has to make to reach the target document. Finally, the average shortest path provides a rough estimate of the number of swipes between any two documents. Even though these measurements are not necessarily reflective of real performance, they allow us to quickly determine the viability of BINGO for targeted search queries.

We conducted a graph analysis of 2,035 VLDB PDF documents that produce little OCR errors when converted to plain text. The dataset had 300 documents with no in-degrees; the rest of the dataset was fully connected. The diameter of the connected components of the graph was 14 edges and the average shortest path between any two connected documents was 4.6 edges.

Based on the results of this preliminary graph analysis, it appears that the VLDB dataset is *fluid* enough, such that even if a user starts a search far away from the target document, with a few swipes he/she will eventually arrive at the target.

To compare the performance of BINGO against keyword search at arriving at a target document, we randomly chose 200 target documents. We chose the highest ranked tfidf word from the target document that appears in at least 100 documents to be the *seed* keyword that we start all searches with. The seed behaves as a topic term if it appears in many documents. In certain cases (8 documents), our seed keyword causes our target document to appear first in the search, we drop such cases from our results.

We simulate keyword specification for two types of users: (i) the *expert* user and (ii) the *inexperienced* user [45]. Expert users tend to use specific keywords in their search, whereas inexperienced users usually conduct their searches using *more generic* (or less specific) keywords. In our tests, the simulated expert user adds one keyword at a time to his/her search query: these keywords are added in descending order of tfidf score from the 50 highest ranking tfidf terms of the target document. The inexperienced user also adds one keyword at a time to his/her search query from the 50 highest ranking tfidf terms of the target document but the words are inserted in descending order of collection frequency: while the expert user can refine the query with specific terms (e.g. rtree, spatial index, etc.), the inexperienced user will tend to pick more generic terms (e.g. query, geographic data). Figure 2 is a histogram of the number of keywords specified to reach different target documents. While often the simulated expert can reach the target document using only one or two keywords, the inexperienced user often has to provide several keywords.

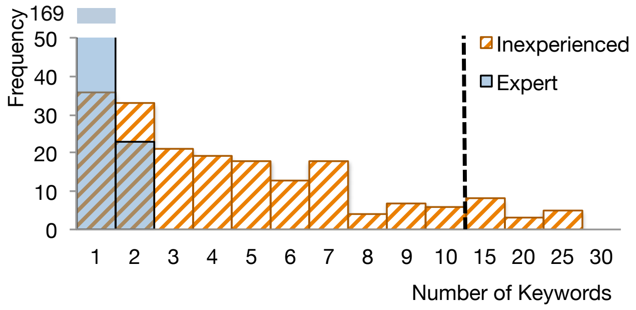


Figure 2. Number of keywords provided by a simulated expert or an inexperienced user to reach a target document chosen at random. The dashed vertical line indicates a change of histogram bin-width on the x-axis.

The words chosen by the simulation illustrate that it behaves close to our intuition. For example to reach the target document: 'M-tree: An Efficient Access Method for Similarity Search in Metric Spaces', the simulated expert user specified {triangle, mtree}, the inexperienced user specified {tree, search, object, index, node, cover, partition, metric}; to reach the target document: 'Code Generation for Efficient Query Processing in Managed Runtimes', the expert user specified {foreach, linq}, the inexperienced user specified {foreach, query, object, input, memory, source, express, program}.

We simulate user behavior over BINGO as follows: given a choice of k reason bins on the current document, the simulated user picks the bin with the highest tf-idf score in the top-50 tf-idf terms of target document or swipes the 'catch-all' bin. Each swipe through a reason bin adds the bin's term to the running search query.

Note, that since the choice of keywords is limited to the top fifty keywords in the target document, the keyword search simulations maintains an unfair advantage over BINGO's simulations where terms are generated from the current document and not the target document.

We successfully terminate the search when the target documents appears in the top seven ranked documents. If more than 100 keywords or swipes are required to reach the target, we terminate the search and deem it a failure.

Figure 3 compares the performance of BINGO against keywords in terms of interaction cost. While each search requires

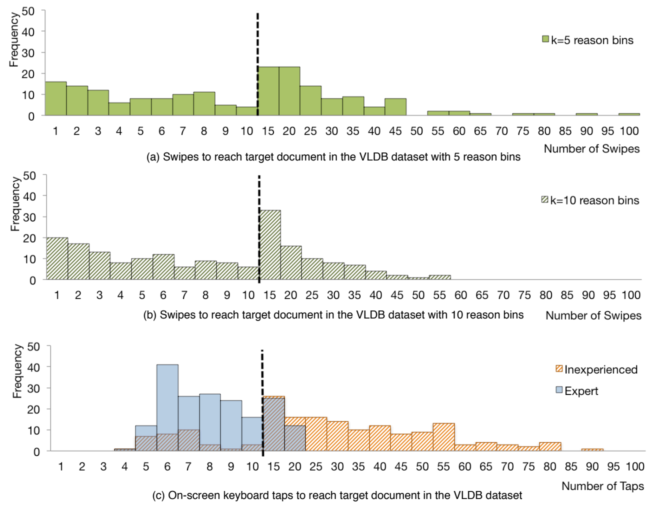


Figure 3. A histogram of the interaction cost of reaching a target document through (a) swiping with $k = 5$ reason bins, (b) with $k = 10$ reason bins, and (c) tapping keywords for expert and inexperienced users. The dashed vertical line indicates a change of histogram bin-width on the x-axis.

on average 1.2 keywords from the expert user to 4.9 keywords from the inexperienced user, the interaction cost in terms of characters tapped on an on-screen keyboard ranges from 8.6 taps to 34 taps respectively. The interaction cost of BINGO is much lower: 16.1 swipes are required on average with 5 reason bins and 11.8 swipes with 10 reason bins. Moreover, taps have a longer tail compared to swipes.

While increasing the number of reason bins reduces the average number of swipes, the reduction is not high enough to overweigh the costs of interface clutter: doubling the bins does not halve the number of swipes.

From these results we conclude that BINGO *provides a sweet spot in terms of the interaction effort required to conduct a targeted search when users are not familiar or experienced with a particular dataset.*

Survey Search Tasks

We conducted a comparative user study of BINGO's search interface against keyword-only search interfaces (KWD) and swiping without reason bins (SWP). These interfaces (KWD and SWP) represent two ends of the interface spectrum. While the former allows for full expressiveness by using keyboard input, the latter allows for fast and convenient interaction, but does not allow for fine-grained articulation of the user's input. To control for aesthetics, we designed KWD and SWP to have aesthetics similar to those of BINGO (See Figure 4). Note, SWP used Rocchio relevance feedback to build search queries from like/dislike decisions.

Our goal in this study was to observe how much time a user spends modifying his/her query at every search iteration until he/she is satisfied with the result given a survey search task. This *revision time* is a measure of how effective an interface is at enabling users to refine their search query. We also observe *precision* or how many documents are added to a reading list given the total number of documents a user reviewed.

Participants and Methods

We recruited 20 participants. Ten subjects were students familiar with database concepts. They were assigned VLDB search tasks, which involved conducting a literature review on 2,035 VLDB papers published between 1975 and 2013. The remaining subjects were students with sufficient digital literacy to conduct online searches. They were assigned game search tasks on the GiantBomb games database, which consisted of 44,150 game descriptions scraped from GiantBomb.com. None of the subjects have used BINGO before.

We first presented a video tutorial of BINGO. Each user then had ten minutes to freely use all three search interfaces, BINGO, KWD, SWP to get familiar with them. Each user was then asked to perform two search tasks. In each interface, users had 5 minutes to complete each task. The order of interfaces and tasks tested for each user was randomized to mitigate user fatigue and learning effects. Users also took sufficient breaks between each task. The tasks were:

Q1-GAMES. Find as many war games involving ships or aircraft carriers as you can.

Q2-GAMES. Find as many decision-making or long-term strategy multi-player games.

Q3-VLDB. Find as many papers as you can that discuss techniques to quickly rank and recommend k tuples in a database.

Q4-VLDB. Find as many papers as you can that analyze vehicle tracking data.

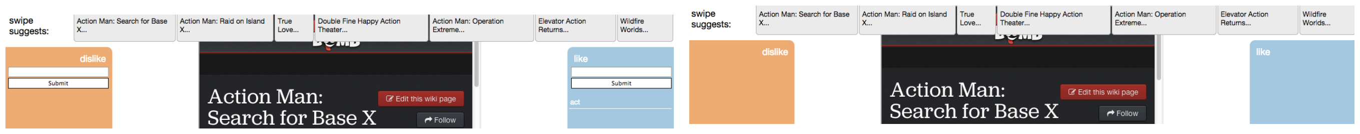
Revision Time

For each user, we observed the time between interactions (swiping through a bin with BINGO, adding a new keyword with KWD, simply swiping left/right with SWP, etc). Since users have different reading behaviors and reading speeds — some users tend to read each document more carefully — we normalized the time between interactions by the user's longest interaction interval. A user's revision time for a given query task on a particular tool is simply the mean normalized inter-interaction time.

Figure 5 illustrates the average of all users' revision time per interface and search task combination. Users review documents with SWP the fastest. A two-factor repeated-measures mixed analysis of variance of revision time reveals only a significant main effect of tool used: $F_{2,119} = 35.020$, p-value < 0.001. No other main effects (dataset, query task or subjects) or interaction effects were significant. We then conducted two-tailed t-tests on all pairwise comparisons. Users spend significantly more time in between interactions when using KWD compared with BINGO and SWP hinting that it requires time to physically type keywords and think of new ones (p-values < 0.001). The slight increase in revision time when using BINGO compared to SWP is also significant (p-value $= 0.02$).

Precision

We measure precision as proportion of documents a user added to his/her reading list given all documents recommended by the interface. A precision value of one indicates


swipe suggests:
Action Man: Search for Base X...
Action Man: Raid on Island X...
True Love...
Double Fine Happy Action Theater...
Action Man: Operation Extreme...
Elevator Action Returns...
Wildfire Worlds...
dislike
Submit
Action Man: Search for Base X
Edit this wiki page
Follow
like
Submit

(a) KWD Interface. Users can provide both positive and negative search keywords.

(b) SWP Interface. Users can only swipe a document left to dislike, right to like. No reason bins or keywords supported

Figure 4. KWD and SWP interfaces are designed to match the aesthetics of BINGO. The underlying document retrieval model is preserved across all interfaces (hence we support negative search keywords in KWD). SWP uses Rocchio relevance feedback to construct search query vectors that use the underlying document retrieval model.

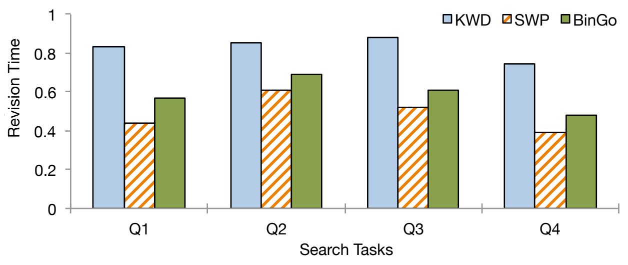


Figure 5. Average revision time per interface/task combination. Q_1 and Q_2 are search tasks on the games dataset, and Q_3 and Q_4 are search tasks on the VLDB dataset.

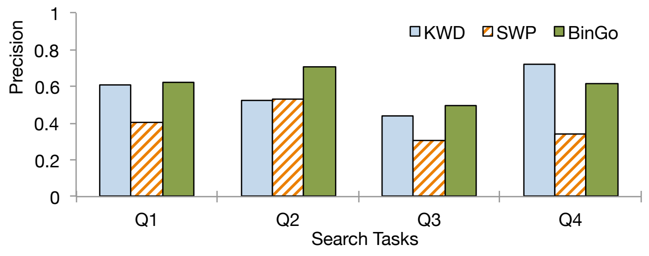


Figure 6. Average precision per interface/task combination. Q_1 and Q_2 are search tasks on the games dataset, and Q_3 and Q_4 are search tasks on the VLDB dataset.

that the user added all recommended documents to his/her reading list; while a precision value of zero indicates the user did not add any of the recommended documents to his/her reading list.

Figure 6 illustrates average precision among all users for a particular search task, interface combination. We observe that BINGO is generally competitive with KWD and that both BINGO and KWD appear to outperform SWP. A two-factor repeated-measures mixed analysis of variance of precision reveals a significant main effect of tool used, $F_{2,119} = 16.181$, p-value < 0.001, and a significant main effect of dataset, $F_{1,119} = 6.261$, p-value $= 0.02$ No other main effects (query task or subjects) or interaction effects were significant. We then conducted two-tailed t-tests on all pairwise comparisons. The slight precision differences between KWD and BINGO are significant (p-value = 0.05) and the differences between KWD and BINGO, and SWP are significant (p-values < 0.001)[11].

From these results, we conclude that BINGO *provides a good balance between the revision speed that* SWP *provides and the search precision that* KWD *provides.*

We also conducted a post-study questionnaire and asked users to rate the ease of use of the different interfaces on a 10-point Likert scale: users found SWP the easiest to use ($\mu_{\text{VLDB}} = 7.5$, $\mu_{\text{GAMES}} = 7.8$, likely due to the lack of an additional interaction target to cross-through), followed by BINGO ($\mu_{\text{VLDB}} = 6.8$, $\mu_{\text{GAMES}} = 7.2$) and then KWD ($\mu_{\text{VLDB}} = 5.7$, $\mu_{\text{GAMES}} = 5.5$). This result is particularly compelling given how conditioned users are on using basic keyword specification interfaces for search.

Users were also asked to rate the quality of recommendations made by the different search interfaces on a 10-point Likert scale: users found KWD to provide the highest recommendation quality ($\mu_{\text{VLDB}} = 7.6$, $\mu_{\text{GAMES}} = 7.4$), matched by BINGO ($\mu_{\text{VLDB}} = 7.8$, $\mu_{\text{GAMES}} = 7.2$) and followed by SWP ($\mu_{\text{VLDB}} = 6.1$, $\mu_{\text{GAMES}} = 5.8$). This qualitative finding corroborates our earlier quantitative precision findings.

Exploration

Finally, we evaluated the effectiveness of BINGO as an exploratory search interface through a comparative user study with KWD. Unlike the previous user-study, the search task we tested here was open-ended. We asked the participants to:

E1-GAMES. Find as many games that you would like to play or suggest to others in less than 15 minutes.

Our goal in this study was to study how effective BINGO is at guiding exploration tasks by providing new search directions.

Participants and Methods

We recruited 21 new participants. The subjects were all students with sufficient digital literacy to conduct online searches and were familiar with video games. None of the subjects used BINGO before. The participants were split into two groups: a control group of 11 participants that used KWD only and a test group that used BINGO. As before, we presented a 5-minute training video tutorial to the users. We then asked users to play with their assigned tool for five minutes. The users directed this playtime.

Games Saved

Figure 7 illustrates the number of games saved (i.e. added to the reading list) for each user per user group. We hypoth-

[11] Pairwise precision comparisons on tools after a per data set breakdown reveal no significant difference between KWD and BINGO on the VLDB dataset and no significant difference between KWD and SWP on the games dataset)

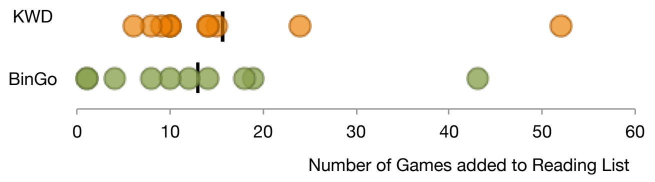


Figure 7. Number of recommendations added to the reading list while exploring the Games dataset

esized that BINGO will outperform KWD in terms of enabling users to explore more (interesting) games. However, the data fails to support this hypothesis: it shows no significant difference between the number of games saved among the different user groups. A post-study survey allowed us to better understand user exploration behavior. With KWD, we asked users to rate on 5-point Likert scale whether they chose keywords that reflected aspects of the game in view that they liked/disliked. Users expressed this to be the case ($\mu = 4.18$). Users also found it somewhat easy to come up with such descriptive words ($\mu = 3.73$). With BINGO, users somewhat found the bins to reflect aspects of the game that they liked/disliked ($\mu = 3.1$).

We asked users if they chose keywords based on what they wanted to see more/less of in future recommendations. Users found this also to be the case ($\mu = 4.64$) and it was easy to come up with such filters ($\mu = 4.27$). With BINGO, users somewhat found the bins to reflect aspects of the game that they would have liked to see more or less of ($\mu = 3.6$).

Finally, we asked users if they chose keywords that would open up their search space to new and interesting directions. Users did not find this usual ($\mu = 3.18$) and reported difficulty in coming up with such keywords ($\mu = 2.91$). With BINGO, users found the bins more helpful in providing new and interesting search directions ($\mu = 3.2$).

We asked users to expand on their ratings. With KWD, one user commented that game descriptions provided him with inspiration to add new keywords. Some users chose keywords specific to games they already played and used the interface to look for those games and explore from there. As one user puts it, "the keywords allowed me to take control in the direction that I want to go but I wouldn't necessarily say that it took me in new interesting directions." We believe game recognition with KWD led users to save more games for later.

With BINGO, some users were expecting bins to provide general concepts instead of document-specific terms. One user commented: "[bins] were related to the aspects I wanted to see more of in other games but were somewhat limited e.g. monkey was followed by banana but lacked the general description level - animals". One user found that BINGO quickly narrowed down the search results to a set of very similar games, which made exploration hard: "when I selected the [bin] London, only games about London showed up - very heavy filter. Perhaps UK ... should be there, too. Moreover it was hard to switch to a new area, ... London related [games] kept showing up for the next 10 results". Some users found reason bins to provide interesting suggestions to expand the search. One user commented: "interesting games that I wouldn't have searched for popped up." It appears that if a reason bin led to a relevant new area, users were pleased and if the reason bin led to a new area that the user lost interest in on the first recommendation, users were quite disgruntled. Note, the current BINGO prototype does not support several features such as removing selected bins from the search.

We also asked users if they wanted to see less reason bins (1) or more reason bins (5) and the average user rating was 2.5. Many users wanted to keep the number of bins as is.

We share with the reader a few remarkable users comments: "Using google to search has a certain art to it, or using keyword search in general. In order to convince people to forgo the skills they've learned in making keyword search engines return good answers, the search recommendations and bin words would have to be way better". This comment highlights the extent of user training bias against innovating in the search space. One user commented on BINGO "On Mobile - YES. Very intuitive, Tinder-like. ... us lazy humans. :)"

CONCLUSION

In this paper, we recognize that popular solutions to querying over text databases can be seen on a spectrum – traditional keyword search allows for free form input but tedious and slow to use. On the other end, binary "like / dislike" classification of documents, popular with recommender systems, allows users to rapidly sift through a large number of documents. This model is also very amenable to swipe-based gestures. BINGO considers a novel best-of-both-worlds solution, allowing for querying of text databases to be performed entirely using gestures. By integrating "reason bins" into a swipe-based gesture, users are able to articulate the reason for liking or disliking a paper in a fluid and efficient gesture.

Our evaluation considers multiple search cost models, keeping in mind user tradeoffs between performing keyword input and gestures. Second, we perform user studies over multiple variants of the interface, evaluating the usage of our system over a series of objective and subjective metrics. We show that BINGO is not only easy to use, but also allows for more efficient querying over text databases. We show that users find its result quality comparable to keyword input with minimal added effort compared with simple swiping gestures.

FUTURE WORK

Going forward, there are several interesting avenues of future work. From a text database querying standpoint, we can look towards alternative signals for bin generation such as prior searches. We can also modify the weights of keywords added to a query based on how hesitant a user was when selecting a reason bin (estimated by swiping duration). From a UI perspective, we wish to consider different visual layouts for the bins depending on the actual form factor of the input device, such as radial and fisheye menus. This would allow us to optimize the gestures for either thumb-based interaction on small smartphones, or multi-finger gestures on large touch screens.

Acknowledgements We acknowledge the support of the U.S. National Science Foundation under awards IIS-1420941, IIS-1422977, IIS-1527779, and CAREER IIS-1453582.

REFERENCES

1. Tinder. `https://www.gotinder.com`.
2. Accot, J., and Zhai, S. More than dotting the i's — foundations for crossing-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, ACM (New York, NY, USA, 2002), 73–80.
3. Adamczyk, P. D., and Bailey, B. P. If not now, when?: the effects of interruption at different moments within task execution. *SIGCHI* (2004).
4. Ahlberg, C., and Shneiderman, B. Visual information seeking: tight coupling of dynamic query filters with starfield displays. *SIGCHI* (1994).
5. Anderson, M. Technology device ownership: 2015. `http://www.pewinternet.org/2015/10/29/technology-device-ownership-2015/`, October 2015.
6. Baeza-Yates, R., Hurtado, C., and Mendoza, M. Query recommendation using query logs in search engines. *EDBT* (2005).
7. Baeza-Yates, R., Ribeiro-Neto, B., et al. *Modern information retrieval*. ACM press New York, 1999.
8. Balfe, E., and Smyth, B. Improving web search through collaborative query recommendation. *ECAI* (2004).
9. Basu, C., Cohen, W. W., Hirsh, H., and Nevill-Manning, C. Technical paper recommendation: A study in combining multiple information sources. *JIPM* (2011).
10. Bates, M. J. The design of browsing and berrypicking techniques for the online search interface. *OIR* (1989).
11. Bhamidipati, S., Kveton, B., and Muthukrishnan, S. Minimal interaction search: Multi-way search with item categories. *ITWP* (2013).
12. Bhavnani, S. K. Domain-specific search strategies for the effective retrieval of healthcare and shopping information. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, ACM (New York, NY, USA, 2002), 610–611.
13. Bruza, P., McArthur, R., and Dennis, S. Interactive internet search: keyword, directory and query reformulation mechanisms compared. *SIGIR* (2000).
14. Burno, R. A., Wu, B., Doherty, R., Colett, H., and Elnaggar, R. Applying fitts' law to gesture based computer interactions. *Procedia Manufacturing 3* (2015), 4342 – 4349.
15. Capra, R. G., and Marchionini, G. The relation browser tool for faceted exploratory search. *JCDL* (2008).
16. Catarci, T., Costabile, M. F., Levialdi, S., and Batini, C. Visual query systems for databases: A survey. *JVLC* (1997).
17. Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning* (1995).
18. Deloatch, R., Gou, L., Kau, C., Mahmud, J., and Zhou, M. Scopeg: A mobile application for exploration and comparison of personality traits. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces*, IUI '16 Companion, ACM (New York, NY, USA, 2016), 102–105.
19. Dignum, S., Kruschwitz, U., Fasli, M., Kim, Y., Song, D., Beresi, U. C., and De Roeck, A. Incorporating seasonality into search suggestions derived from intranet query logs. *WI-IAT* (2010).
20. Diriye, A., and Golovchinsky, G. Querium: a session-based collaborative search system. In *European Conference on Information Retrieval*, Springer (2012), 583–584.
21. Downey, D., Dumais, S., Liebling, D., and Horvitz, E. Understanding the relationship between searchers' queries and information goals. *JIKM* (2008).
22. Downey, D., Dumais, S. T., and Horvitz, E. Models of searching and browsing: Languages, studies, and application. *IJCAI* (2007).
23. Drucker, S. M., Fisher, D., Sadana, R., Herron, J., and schraefel, m. Touchviz: A case study comparing two interfaces for data analytics on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 2301–2310.
24. Dupret, G., and Mendoza, M. Automatic query recommendation using click-through data. *PPAI* (2006).
25. Girod, B., Chandrasekhar, V., Chen, D. M., Cheung, N.-M., Grzeszczuk, R., Reznik, Y., Takacs, G., Tsai, S. S., and Vedantham, R. Mobile visual search. *Sig. Proc. Magazine* (2011).
26. He, Q., Jiang, D., Liao, Z., Hoi, S. C., Chang, K., Lim, E.-P., and Li, H. Web query recommendation via sequential query prediction. *ICDE* (2009).
27. Hölscher, C., and Strube, G. Web search behavior of internet experts and newbies. *Computer Networks* (2000).
28. Hub, A., Blank, D., Henrich, A., and Muller, W. Picadomo: Faceted image browsing for mobile devices. *CBMI* (2009).
29. Idreos, S., and Liarou, E. dbtouch: Analytics at your fingertips. In *CIDR* (2013).
30. Jansen, B. J. Search log analysis: What it is, what's been done, how to do it. *LIS research* (2006).
31. Joachims, T. *Text categorization with support vector machines*. Springer, 1998.
32. Kashyap, A., Hristidis, V., and Petropoulos, M. Facetor: cost-driven exploration of faceted query results. *JIKM* (2010).
33. Khoussainova, N., Kwon, Y., Balazinska, M., and Suciu, D. Snipsuggest: context-aware auto-completion for sql. *VLDB* (2010).

34. Kriewel, S., and Fuhr, N. *Adaptive search suggestions for digital libraries*. Springer, 2007.

35. Lau, T., and Horvitz, E. Patterns of search: analyzing and modeling web query refinement. In *UM99 User Modeling*. Springer, 1999, 119–128.

36. Li, Y. Gesture search: a tool for fast mobile data access. *UIST* (2010).

37. Lu, S., Mei, T., Wang, J., Zhang, J., Wang, Z., Feng, D. D., Sun, J.-T., and Li, S. Browse-to-search. *ICME* (2012).

38. MacKenzie, I. S., and Soukoreff, R. W. Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction* (2002).

39. MarketingCharts. In the us, time spent with mobile apps now exceeds desktop web access. `http://www.marketingcharts.com/online/in-the-us-time-spent-with-mobile-apps-now-exceeds-the-desktop-web-41153/`, March 2015.

40. Morton, K., Balazinska, M., Grossman, D., and Mackinlay, J. Support the data enthusiast: Challenges for next-generation data-analysis systems. *VLDB* (2014).

41. Nakarada-Kordic, I., and Lobb, B. Effect of perceived attractiveness of web interface design on visual search of web sites. *SIGCHI* (2005).

42. Nandi, A., and Jagadish, H. Assisted Querying using Instant-response Interfaces. *SIGMOD* (2007).

43. Nandi, A., Jiang, L., and Mandel, M. Gestural Query Specification. *VLDB* (2014).

44. Negulescu, M., Ruiz, J., Li, Y., and Lank, E. Tap, swipe, or move: Attentional demands for distracted smartphone input. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, ACM (New York, NY, USA, 2012), 173–180.

45. Nelson, M., Shaw, M., and Strader, T. *AMCIS 2009*. Lecture notes in business information processing. Springer, 2009.

46. Ortega, R. E., Avery, J. W., and Frederick, R. Search query autocompletion. *US Patent 6564213* (2003).

47. Peng, H., Long, F., and Ding, C. Feature selection based on mutual information criteria. *TPAMI* (2005).

48. Realmac Software. Clear - tasks, reminders & to-do lists. `http://realmacsoftware.com/clear`.

49. Resnick, P., and Varian, H. R. Recommender systems. *CACM* (1997).

50. Robertson, S. E., and Walker, S. Okapi/keenbow at trec-8. *TREC* (1999).

51. Roy, S. B., Wang, H., Nambiar, U., Das, G., and Mohania, M. Dynacet: Building dynamic faceted search systems over databases. *ICDE* (2009).

52. Rzeszotarski, J. M., and Kittur, A. Kinetica: Naturalistic multi-touch data visualization. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 897–906.

53. Safavian, S. R., and Landgrebe, D. A survey of decision tree classifier methodology. *SMC* (1991).

54. Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *JIPM* (1988).

55. Schulze, F., and Woerndl, W. Using touch gestures to adjust context parameters in mobile recommender and search applications. *CTS* (2011).

56. Soo Y. Rieh, H. X. Analysis of multiple query reformulations on the web: The interactive information retrieval context. *IPM* (2006).

57. Stefanidis, K., Pitoura, E., and Vassiliadis, P. On relaxing contextual preference queries. *MDM* (2007).

58. Sugiyama, K., et al. Scholarly paper recommendation via user's recent research interests. *JCDL* (2010).

59. Telang, A., Chakravarthy, S., and Li, C. Querying for information integration: How to go from an imprecise intent to a precise query? *COMAD* (2008).

60. Tunkelang, D. Faceted search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* (2009).

61. Ward, D. J., Blackwell, A. F., and MacKay, D. J. Dasher – a Data Entry Interface Using Continuous Gestures and Language Models. *UIST* (2000).

62. Wikipedia. Google Now, 2016. `https://en.wikipedia.org/wiki/Google_Now`, [Online; accessed 6-Jan-2017].

63. Wolfe, J. M., and Gancarz, G. Guided search 3.0. *Basic and clinical applications of vision science* (1997).

64. Xu, S., Jin, T., and Lau, F. A new visual search interface for web browsing. *WSDM* (2009).

65. Zha, Z.-J., Yang, L., Mei, T., Wang, M., and Wang, Z. Visual query suggestion. *ICME* (2009).

66. Zhai, S., and Milgram, P. Human performance evaluation of manipulation schemes in virtual environments. *VR* (1993).