SMS-Based Web Search for Low-end Mobile Devices

Jay Chen
New York University
jchen@cs.nyu.edu

Lakshmi Subramanian
New York University
lakshmi@cs.nyu.edu

Eric Brewer
University of California, Berkeley
brewer@cs.berkeley.edu

ABSTRACT

Short Messaging Service (SMS) based mobile information services have become increasingly common around the world, especially in emerging regions among users with low-end mobile devices. This paper presents the design and implementation of SMSFind, an SMS-based search system that enables users to obtain extremely concise (one SMS message of 140 bytes) and appropriate search responses for queries across arbitrary topics in one round of interaction. SMSFind is designed to complement existing SMS-based search services that are either limited in the topics they recognize or involve a human in the loop.

Given an unstructured search query, SMSFind, uses a conventional search engine as a back-end to elicit several search responses and uses a combination of information retrieval techniques to extract the most appropriate 140-byte snippet as the final SMS search response. We show that SMSFind returns appropriate responses for 57.3% of ChaCha search queries in our test set; this accuracy rate is high given that ChaCha employs a human to answer the same questions. We have also deployed a pilot version of SMSFind for use with a small focus group in Kenya to explore the interaction issues of such a system and share our early experiences.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Question-answering (fact-retrieval) systems*

General Terms

Algorithms, Design, Experimentation

Keywords

Cell Phones, SMS, Search, Question/answering

MobiCom'10, September 20–24, 2010, Chicago, Illinois, USA.

1. INTRODUCTION

The exceptional growth of the mobile phone market has motivated the design of new forms of mobile information services. With the growth of Twitter [1], SMS GupShup [2] and other social messaging networks, the past few years have witnessed a growing prevalence of Short-Messaging Service (SMS) based applications and services. SMS-based services are also increasingly common in developing regions. Despite the increasing power of mobile devices with the advent of "smart phones", a significant fraction of mobile devices in developing regions are still simple low-cost devices with limited processing and communication capabilities. Due to a combination of social and economic factors, voice and SMS will likely continue to remain the primary communication channels available for a non-trivial fraction of the population in developing regions.

For any SMS-based web service, efficient *SMS-based search* is an essential building block. SMS-based search is a rapidly growing global market with over 12 million subscribers as of July 2008 [3]. An SMS message is constrained to 140 bytes which drastically limits the amount of information in a search response. SMS-based search is also non-interactive due to the search response time; anecdotally [4], existing SMS-based search engines take on the order of tens of seconds [5, 6] to several minutes per response [7]. Even without the 140-byte SMS size constraint, tailoring traditional web search to mobile devices is a challenging problem due to the small form factor and low bandwidth. Unlike desktop search, users on mobile devices rarely have the luxury of iteratively refining search queries or sifting through pages of results for the information they want [8]. In this paper, we address the problem of *SMS-based search*: *how does a mobile user efficiently search the Web using one round of interaction where the search response is restricted to one SMS message?*

Though we do not know the internals of existing SMS search algorithms, we can observe from the user interface and documentation that existing automated services for SMS web search such as Google SMS [5] and Yahoo! oneSearch [6]) encourage users to enter queries for a number of pre-defined topics, or *verticals*. These pre-defined topics are either identified through the use of special keywords within the search query such as "define" or "movies" (e.g. Google SMS: "define boils") or have specialized parsers to determine which of the topics is intended (e.g. querying "AAPL" to Yahoo! oneSearch is a query for information about "stock quote"). ChaCha [7], a recent SMS-based search engine, hires humans to search the web and answer questions in an SMS

response. TradeNet, now called eSoko [9], is a mobile marketplace platform in Ghana which would require an SMS based search service as well. Instant access to small bits of information is the motivation behind all of the existing SMS based search systems and of this work.

None of the existing automated SMS search services is a complete solution for search queries across arbitrary topics. Similar to traditional web search queries, SMS search queries suffer from the *long tail phenomenon*: there exists a long tail of search queries whose topics are not popular (e.g. "what are graduation gift ideas?" or "what chemicals are in a fire extinguisher"). We confirm that this indeed is the case in our sample of ChaCha questions where only 21% of the queries in our data set are verticals and 79% are long tailed.

In this paper, we describe the design and implementation of SMSFind, an SMS-based search engine specifically designed for the long tail of search queries that are spread across a wide range of topics. These topics represent the queries in a mobile search workload that are not answered by existing domain specific verticals. SMSFind is designed to integrate into an existing SMS search service to answer queries for unsupported long tail topics. Given a query, SMSFind uses a traditional search engine as a back-end to elicit several search results and extract the appropriate 140 bytes as the SMS search response. Section 6.2 further explains how vertical and long tail queries are defined in this work.

SMSFind uses a combination of well-known information retrieval techniques to address the appropriate information extraction problem. SMSFind is designed for *unstructured queries* supporting a similar format as a standard search engine query. The key idea of SMSFind is that meaningful SMS queries typically contain a term or a collection of consecutive terms in a query that provides a *hint* as to what the user is looking for. The hint for a query can either be explicitly provided by the user or automatically derived from the query. SMSFind uses this hint to address the information extraction problem as follows: Given the top N search responses to a query from a search engine, SMSFind extracts *snippets* of text from within the neighborhood of the hint in each response page. SMSFind scores snippets and ranks them across a variety of metrics.

We have evaluated SMSFind on a corpus of questions screen gathered from the website of the ChaCha SMS question/answering service. The ChaCha search queries are in question-style format which we converted to unstructured query format using a set of transformation rules. Based on extensive evaluation and human verification, we show that SMSFind can answer 57.3% of ChaCha queries. This is a significant result because these questions are currently being answered manually by humans. There are still questions that are difficult to answer using statistical methods such as, "do they kill horses to make glue" (as we discuss in Section 7.2), but to the best of our knowledge this is the first effort that addresses the problem of SMS-based search for long tail mobile queries.[1]

[1]Google Squared [10] appears to be working toward this as well, but the technical approach and how it relates to SMS search is unclear from the publicly available information at this time.

2. RELATED WORK

While there has been relatively little research on SMS-based search, in this section, we take a slightly larger view of the problem space and contrast SMSFind with mobile search services, question/answering (Q/A) systems from the Text REtrieval Conference (TREC) community [11], and text summarization techniques.

2.1 Mobile Search Characteristics

Mobile search is a fundamentally different search paradigm than conventional desktop search. Yet, we continue to view mobile web search as either an extension of the desktop search model for high-end phones (such as PDA/iPhone) or a slightly restricted version via XHTML/WAP on low-end devices. Mobile search in both of these settings differs from traditional desktop search in several ways as shown in recent studies by Kamvar et al. [12, 13]. The first study [12] found that the mobile search click-through rate and the search page views per query were both significantly lower in comparison to desktop search. Meaning, most mobile search users tend to use the search service for short time-periods and are either satisfied with the search engine snippet responses or do not find what they were looking for. The study also found that the persistence of mobile users was very low indicating that the *vast majority of mobile searchers approach queries with a specific topic in mind and their search often does not lead to exploration.* The second study [13] showed that the diversity of search topics for low-end phone users was much less than that of desktop or iPhone-based search. This result suggests that the information needs are broad, but are not satisfied by the information services available on low-end phones. We find corroborating evidence in our analysis of ChaCha queries that mobile question diversity across topics is high given a more expressive input modality such as voice. As a whole, these studies indicate a pressing need for rethinking the current mobile search model for low-end mobile devices.

2.2 SMS-based Search Services

SMS-based search is very different from conventional mobile search via XHTML/WAP. An attractive aspect of SMS-based search is the lower barrier to entry of SMS (in comparison to other data services) due to the use of low-end phones and widespread availability of SMS. In developing countries, SMS is the most ubiquitous protocol for information exchange next to voice. In addition, there is speculation that "the economic downturn will most likely dampen growth for the more luxury-based mobile services, but SMS is expected to continue its growth as it is popular, cheap, reliable and private." [14].[2] Many of the top search engine players like Google [5], Yahoo! [6], and Microsoft [15] have entered the SMS search market and developed their own versions of SMS-based search services. All these services have been tailored for very specific topics (e.g. directory, weather, stock quotes) and specialized in nature.

These automated services are not the only ones available. ChaCha [7] and Just Dial [16] (in India) are SMS- or voice-based question/answering systems using a human to respond to queries. The queries to ChaCha and JustDial are in natural language question form, interpreted by a human who

[2]Most notably by Nielsen [3], but also by other private market research firms [14].

looks for the solution online, and responds with an answer via an SMS. A recent private study by mSearchGroove [4] comparing the accuracy of responses of these services has shown that the automated systems suffered from low accuracy when satisfying an arbitrary set of queries: Google SMS 22.2%, Yahoo! oneSearch 27.8%, as compared to 88.9% for ChaCha. However, the study also observed that the median response time for the automated services was on the order of 10-14.5 seconds, whereas the median response time for ChaCha was 227.5 seconds. Involving a human in the loop drastically increases response time.

The problem we seek to answer is how do we build a system that achieves the best of both worlds: an SMS search system that is both a fast (automatic) and provides accurate query responses? One reason this problem is hard is that search queries are inherently ambiguous, yet returning a disambiguated result is especially vital to SMS search queries for various reasons [8].

2.3 Question/Answering Systems

The problem that SMSFind seeks to address is similar to, but distinct from, traditional question/answering systems developed by the Text REtreival Conference (TREC) [17] community. TREC has evolved into many separate tracks devoted to specific areas such as: simple ad-hoc tasks, question/answering tasks, million query tasks, etc. Nevertheless, our problem is different from each of the TREC tracks for at least one of three dimensions: (a) the nature of the input query; (b) the document collection set; (c) the nature of the search results in the query response. These three dimensions are derived by Carmel et. al. [18] in a model for assessing query difficulty.

First, from the input query standpoint, SMSFind is primarily suited for unstructured search style queries. SMSFind can be extended for simple natural language style queries using existing query transformation techniques as we discuss later. The distribution of the queries is also dependent on the mobile context (i.e. SMS or voice queries as opposed to Desktop search queries or iPhone queries). Second, SMSFind is a *snippet extraction* and *snippet ranking* algorithm which outputs condensed text snippets in existing pages as search responses. Third, the most notable differences of our problem in comparison to many TREC tracks is that in SMSFind the collection of documents being searched over is a set of heterogeneous, "noisy", and hyper-linked documents (web pages) indexed by Google. In contrast, the TREC ad-hoc track and TREC question/answering tracks have until recently used a collection of newswire documents which are very clean, and blog documents which have some noise [19]. Newswire and blog collections are also not hyper-linked and are several orders of magnitude smaller than the web. Prior work suggests that the size of the collection affects how well link analysis techniques may be useful to information retrieval [20]. More recently the TREC web, terabyte [21], and million query [22] tracks have used hundreds of millions of web pages as the document collection to allow for link analysis techniques to be applied toward the tasks, and there have been many systems that leverage the web for either the main or auxiliary corpus. The fact that the collection is at least two orders of magnitude greater than any TREC evaluation, significantly noisier in terms of information diversity, formatting, and being hyper-linked means that it is a different (and more challenging) problem, which leads us to adopt a different solution. The earliest and most closely related system to SMSFind in terms of architecture and approach is AskMSR [23]. AskMSR also leveraged the data redundancy of the web as opposed to sophisticated linguistic techniques to extract n-gram answers.

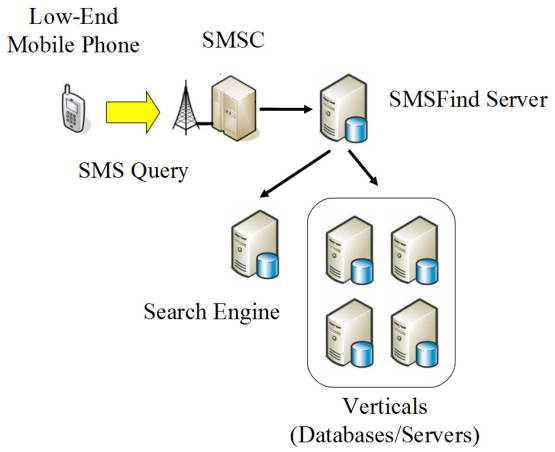

Figure 1: System architecture

SMSFind shares techniques with prior systems [24, 23, 25], but is designed specifically for the unique requirements of SMS search in terms of its input and output. SMSFind expects mobile search queries along with a hint and returns snippets whereas existing systems may expect natural language questions and return n-grams or entire paragraphs and possibly the source document.

2.4 Automatic Text Summarization

Snippet extraction is also tangentially related to summarization, but only to the extraction task (finding sections of the text and producing them verbatim), and not the abstraction task (producing material in a new way). However, the main difference is the goal of these two problems. The goal of extraction is still to summarize the contents of one or more documents. The goal of snippet extraction in SMSFind is to find the correct answer to a query. For this reason, our problem is more similar to information retrieval. That said, many of the of the techniques are used in both areas of research including: Naive-Bayes methods [26], Rich features [27], and other statistical methods.

3. SMSFIND PROBLEM

The architecture of our SMS search system (Figure 1) consists of a query server that handles the actual search query and results, and an SMS gateway that is responsible for communication between the phone clients and the query server. The client is a user with a mobile phone who sends an SMS message to the short code (a special telephone number, shorter than full telephone numbers used to address SMS and MMS messages) for our service, which arrives at the SMS gateway and is then dispatched to our server for processing. At our query server the query is then sent to a general search engine and result pages are downloaded. The query server extracts the results from the downloaded pages,

and returns them to the SMS gateway which sends it back to the client that issued the request.

3.1 Known Verticals vs Long Tail

Certain SMS based queries are best answered by querying known verticals (e.g., flights, directions, weather). Google SMS search may be viewed as a wrapper around its search service for a fixed set of known categories such as phone numbers, weather, flight information, address etc. Our complete SMS search service consists of appropriate parsers and vertical redirectors for a few known categories (phone numbers, weather, addresses). For instance, *weather.com* and *yellow.com* are examples of such verticals for weather and yellow pages. For these categories with existing verticals, generating an SMS search response requires a simple transformation of the query into an appropriate database query (or filling a form) at the corresponding vertical web portal.

The focus of SMSFind is to handle long tail queries that do not fit into verticals. Handling queries for verticals is a relatively straightforward if tedious process and suffers from rapidly diminishing returns. Our complete SMS search service supports a basic set of vertical topics which is a subset of the Google SMS topics. As a part of this system, the SMSFind algorithm is placed behind a categorizer that first determines whether a given query belongs to an implemented vertical based on the existence of defined keywords or if it should be sent to SMSFind.

3.2 SMSFind Search Problem

SMSFind is designed for unstructured search queries across arbitrary topics. Given any query, SMSFind first uses an existing search engine as a back-end to obtain the top few search result pages. Using these pages the remaining problem is to parse the textual content to obtain the appropriate search response that can be condensed to one SMS message. This is essentially a problem of determining appropriate condensed snippets of text across the result pages that form candidate SMS search responses. We define a *snippet* as any continuous stream of text that fits within an SMS message.

SMSFind uses a hint for every query as an important clue to mine every result page for appropriate text snippets. A *hint* refers to either a term or a collection of consecutive terms within the query that is roughly indicative of what type of information the user is searching for. Given that the hint will appear in the response page, SMSFind uses the neighborhood of the text around the hint to mine for appropriate textual snippets. To explain our problem and algorithm we assume that the hint is given by the user explicitly, but later we discuss how the hint may be automatically extracted from natural language questions.

The SMSFind search problem can be characterized as follows:

Given an unstructured SMS search query in the form of <query, hint> and the textual content of the top N search response pages as returned by a search engine, extract a condensed set of text snippets from the response pages that fit within an SMS message (140 bytes) that provide an appropriate search response to the query.

This problem definition explicitly assumes that the hint is specified for every query. Existing SMS-based search services like Google SMS have a similar explicit requirement where a keyword is specified as the last term in a query; the difference being that the existing systems only support a fixed set of keywords whereas SMSFind allows arbitrary hints.

4. SMSFIND SEARCH ALGORITHM

In this section, we describe our algorithm to extract snippets for any given unstructured query of the form <query, hint>.

4.1 Basic Idea

Search queries are inherently ambiguous and a common technique to disambiguate queries is to use additional contextual information from which the search is being conducted [28, 29]. Loosely, the term "context" is any additional information associated with a query that provides a useful hint in providing a targeted search response for a query [30, 31]. Similar to these works, SMSFind uses an explicit hint so the snippet extraction algorithm can identify the approximate location of the desired information in a search response page.

We motivate the use of a hint using a simple example of a long tail query. Consider the query "Barack Obama wife" where the term "wife" represents the hint. When we give this query to a search engine, most search result pages will contain "Michelle" or "Michelle Obama" or "Michelle Robinson" or "Michelle Lavaughn Robinson" within the neighborhood of the word "wife" in the text of the page. For this query, to determine any of these as appropriate search responses, SMSFind will search the neighborhood of the word "wife" in every result page and look for commonly occurring *n-grams* (where n represents one to five consecutive words). For example, "Michelle Obama" is a n-gram which is a $2-$gram.

A simple algorithm to determine the correct answer to this example query is to output all popular n-grams within the neighborhood of the hint and rank them based on different metrics (frequency, distance etc.). However, outputting commonly occurring n-grams as search responses is only appropriate when the actual search response for a query is a $1-5$ word answer. For several common SMS-search queries, the actual appropriate search response is embedded in a sentence or a collection of few sentences. In such cases, we need to extract entire snippets of text as a search response as opposed to just n-grams.

SMSFind makes a clear distinction between *n-grams* and *snippets*. Though both represent continuous sequences of words in a document, a n-gram is extremely short in length ($1-5$ words), whereas a text snippet is a sequence of words that can fit in a single SMS message. In our SMSFind algorithm, n-grams are used as an intermediate unit for our statistical methods whereas snippets are used for the final ranking since they are appropriately sized for SMS.

We next describe the basic SMSFind algorithm. Consider a search query (Q, H) where Q is the search query containing the hint term(s) H. Let $P_1, \ldots P_N$ represent the textual content of the top N search response pages to Q. Given (Q, H) and $P_1 \ldots P_N$, the SMSFind snippet extraction algorithm consists of three steps:

Neighborhood Extraction: For each result page P_i, SMSFind searches for each appearance of the hint term H and extracts a textual neighborhood around the hint which represents a candidate snippet of length covering one SMS message in either side of the hint. For each snippet, we extract all unique n-grams with up to 5 words. The choice of the limit 5 is motivated by the fact that the Linguistic Data

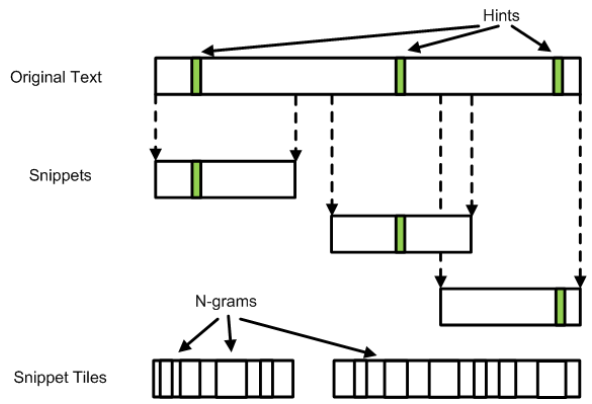

Figure 2: Snippet creation, aggregation into snippet tiles, and n-grams

Consortium (LDC) [32] publishes the web frequency of every n-gram with up to 5 words.

N-gram Ranking: We rank the n-grams based on three metrics: distance to the hint, frequency of occurrence, and mean rank of the result page. We also use the relative rarity of a n-gram on the web to normalize the n-gram ranking.

Snippet Ranking: We define the rank of any snippet as a cumulative sum of the top-few ranked n-grams within the snippet. Among all snippets, we determine the top-ranked snippet(s) as the search response.

We now elaborate upon these three steps.

4.2 Neighborhood Extraction

Given a query (Q, H) and its result pages $P_1, \ldots P_N$, SMSFind first extracts snippets around the hint H in each of the pages. For each page P_i, we find every occurrence of the hint H in the page. Each snippet is up to 140 bytes long and the hint is as centered as much as the surrounding text allows. We found that delineating snippets by sentence boundaries lead to many corner cases due to noise that could skew the statistical results. The snippets are then merged if they overlap to avoid double counting into *snippet tiles* (Figure 2). These snippet tiles form the basis of all further measurements and calculations, and it is only within these snippet tiles that the final result is extracted.

From a practical standpoint of not needing to download several pages and having sufficient diversity to extract n-grams and snippets, we found that a value of $N = 10$ works well. We extract the text from these web pages by filtering out all scripts, hypertext tags, and non-ASCII symbols so we are left with plain text which is similar to what would be rendered by a standard browser.

4.3 N-gram Ranking

N-gram ranking is a critical step in the SMSFind snippet extraction algorithm. Given any snippet extracted around the hint, the first step in our n-gram ranking algorithm is to gather all possible n-grams in the snippet. Table 1 illustrates briefly how the n-grams are generated. The goal of the n-gram ranking algorithm is finding the n-grams that are most likely to be related to the correct response.

The basic rationale of our n-gram ranking algorithm is that any n-gram which satisfies the following three properties is potentially related to the appropriate response for a query with a specified hint:

1. the n-gram appears very frequently around the hint.
2. the n-gram appears very close to the hint.
3. the n-gram is not a commonly used popular term or phrase.

Table 1: Slicing example for the text "the brown cow jumped over the moon", hint = "over"

N-gram	Frequency	Min. Distance
"the"	2	1
"the brown"	1	3
"the brown cow"	1	2
"brown cow jumped"	1	1
...	-	-

As an example, the n-gram "Michelle Obama" is not a commonly used phrase and appears relatively frequently and in close proximity of the hint "wife" for the top search response pages to the query "Barack Obama wife". Therefore, this n-gram is highly relevant for the search response for the query.

For each unique n-gram in any snippet, we compute three independent measures:

Frequency - The number of times the n-gram occurs across all snippets.

Mean rank - The sum across every occurrence of a n-gram of the PageRank of the page in which it occurs, divided by the n-gram's raw frequency. This is to incorporate the ranking system of the underlying search engine in our overall ranking function. (Some n-grams have a raw mean rank of less than 1 because the page containing the search results is assigned a rank of 0.)

Minimum distance - The minimum distance between a n-gram and the hint across any occurrences of both. Intuitively, this metric indicates the proximity of the hint defined by the user is to the search query. It is used as a part of our overall ranking function to allow the user hint to disambiguate two otherwise similarly ranked n-grams.

An example of the metrics we have at this point is shown in Table 2. In this example, the query "the office dwight actor" should return the response "rainn wilson" as highlighted in the table. Note that this example is exactly in the range of queries that we are interested in, it is too rare for a custom extractor and common enough to be detectable by our system. From the list of n-grams we can observe that after slicing, most of the top results are highly relevant to the query according to our metrics.

Filtering n-grams: Before ranking n-grams, we filter the set of n-grams based on the three measures: frequency, mean rank and minimum distance. A n-gram should have a minimum frequency and should be within a certain minimum distance of the hint to be considered. For $N = 10$, we set a minimum frequency bound of 3 and a minimum distance threshold of 10; we choose these thresholds experimentally based on manual analysis of n-grams across sample queries. Similarly, we ignore all n-grams with a very low mean PageRank.

Ranking n-grams: Associating relative importance to any of the metrics or naively ranking based on a single met-

Table 2: List of top 10 n-grams results for the query "the office dwight actor" and their associated raw metrics prior to normalization

N-gram	Frequency	Minimum Distance	Mean Rank
wilson	16	1	1.5
rainn	16	1	1.25
rainn wilson	15	1	1.33
dwight schrute	9	2	0.78
schrute	9	2	0.77
actor rainn wilson	7	0	1.14
plays dwight	7	2	0.57
actor rainn	7	0	1.14
wilson who plays	5	2	0.8

ric is not appropriate. To combine the different metrics into a single metric we perform two simple steps. First, we normalize the raw frequency, mean rank, and minimum distance to a uniform distribution within the range between 0 and 1. We denote the three normalized scores of a n-gram s as $freq(s)$, $meanrank(s)$, $mindist(s)$. Second, the overall ranking score of a n-gram s is a linear combination of the three normalized ranks:

$$rank(s) = freq(s) + meanrank(s) + mindist(s)$$

We use the ranking score to rank all n-grams associated with a query. We need to consider one important detail in the normalization of the frequency score. If two n-grams s, t have the same frequency measure but if n-gram s has a much lower web frequency than n-gram t (s is rarer than t), then s needs to be higher ranked than t. We use the "Web 1T 5-gram Version 1" dataset from the LDC to obtain the frequency for any n-gram and compute its normalized frequency.

4.4 Snippet Ranking Algorithm

If the answer to a query is a very short response of a few words, then the best SMS based search response is to output all the top-ranked n-grams associated with a query. However, given a query, it is hard to determine whether the answer is embedded in a single n-gram or is actually a combination of multiple n-grams. Our best bet in such a scenario is to output the best possible snippet under the hope that the answer is embedded in the snippet and the user can interpret it.

The n-gram ranking algorithm cannot be extended to ranking snippets since almost all of the snippets are unique, and their frequency measure will be 1. We extend the n-gram ranking algorithm to compute the rank of snippets based on the n-grams present within a snippet. In addition, different snippets may contain different number of n-grams which may introduce a bias in the ranking function. To avoid such a bias, we introduce a top-K n-grams based ranking function.

Snippet rank: Consider a snippet S with a set of n-grams $T = \{t_1, \ldots t_m\}$ with corresponding n-gram ranks $rank(t_1), \ldots rank(t_m)$. Let $t_{i_1}, \ldots t_{i_K}$ represent the top K ranked n-grams in T. Then the rank of snippet S is:

$$rank(S) = \sum_{j=1}^{j=K} rank(t_{i_j})$$

In other words, we define the rank of a snippet based on the cumulative rank of the top-K ranked n-grams within the snippet. In practice, we choose $K = 5$. In the *snippet ranking* phase, we determine the highest ranked snippet is the most relevant response to the original query. Recall that our snippets were designed to be under 140 bytes, but the snippet tiles may actually be longer depending on overlaps during the merging process. To find the best snippet, we first split each snippet tile into snippets using a 140 byte sliding window across each snippet tile that respects word boundaries. We then score each snippet based on the sum of the top K n-grams and return the top scoring snippet as the final result.

In the evaluation, we show that ranking n-grams first before ranking snippets is critical for better accuracy; in other words, directly scoring snippets from the web page results without using n-grams performs very poorly in practice.

4.5 Hint Extraction from the Query

We have thus far assumed that every query is associated with a hint, and for unstructured queries it is a natural tendency for the user to enter the hint either at the beginning or at the end. However, even if questions are entered in natural language format, extracting the hint automatically is not difficult. We describe a simple rule-based approach for deriving the hint for natural language question-style queries as a proof of concept. The approach we use is similar to surface pattern matching techniques [25].[3]

A cursory analysis of a sample of 100,000 queries from ChaCha SMS-based search queries reveals that a large fraction of queries use common syntactic structures where the hint is easily identified. Nearly 95% of ChaCha search queries are English questions with 14 terms or less (75% of queries contain less than 10 terms). Based on the structure of the question, we have categorized a common class of questions and their corresponding transformation rules to determine the hint.

We found that that nearly 45% of the queries began with the word "what", of which over 80% of the queries are in standard forms (e.g. "what is", "what was", "what are", "what do", "what does"). For each of these patterns, we can write a simple transformation rule to extract the hint from the corresponding sentence which is typically either immediately after the question or toward the end of the sentence. For example, for the query "what is a quote by ernest heming-

[3]More generally, determining what a natural language question is "about" has been studied, and there has been extensive work on question classification which is related to this problem [33].

way", the "what is X" pattern uses a simple matching rule to extract the hint term "quote" (if "a" is ignored as a stop word). The remaining terms minus stop words are then used as the query, and the final <query, hint> is: <"ernest hemingway", "quote">.

5. IMPLEMENTATION

The core SMSFind algorithm is implemented in only 600 lines of Python code and uses publicly available parsing libraries. We have not implemented any optimizations or caching, but SMSFind generally returns results within 5-10 seconds while running on a 1.8Ghz Duo Core Intel PC with 2 GB of RAM and a 2 Mbps broadband connection. This response time is dominated by the time required to fetch query results from Google and download web pages referred to by the results. To deploy SMSFind as a search service we implemented a front-end to send and receive SMS messages. We setup a SMS short code with a local telco in Kenya, and route all SMS requests and responses to and from our server machine running the service across a Samba 75 GSM modem and Kannel an open source SMS Gateway [34]. As a proof of concept, and to improve the overall usability of our system we have also implemented interfaces to several basic verticals as a part of the service including: weather, definitions, local business results, and news. Each of these interfaces to verticals is under 150 lines of Python code. We are still iterating with our focus group on the implementation details of our system.

6. EVALUATION

Completely evaluating such a system is non-trivial; a large set of realistic queries must be asked, and answers must be judged either by judges or the users themselves. We derive a realistic set of queries by modifying real ChaCha queries. We then evaluate our system performance in absolute terms to explore the limitations of our system and potential improvements.

In our evaluation, the answers returned by SMSFind are judged correct *if the all of the correct answer terms appear anywhere in the returned snippet.* The correct answers were first decided by three judges who came up with correct answers by manual inspection of each question, referring to the ChaCha answer for reference. Sometimes even the ChaCha answer was considered incorrect by the judges and a different answer was determined online and used instead. There were also queries that had several acceptable answers, we consider all of those "correct" in our evaluation. This is the simple rating system we use throughout the evaluation. We do not currently rank our scores based on other important features such as readability or clarity. For a full-fledged question/answering system, incorporating mechanisms to improve readability [35] would likely be necessary.

6.1 Data

Our set of queries consists of a corpus of 100,000 real SMS search queries scraped from ChaCha's [7] public website on June 1, 2009. These queries are in Natural Language question format. In contrast to the logs analyzed previous studies we found in our data an average of 9.20 words per query and 47.09 characters per query, as compared to 3.05 and 18.48 respectively reported in [36]. The TREC [37] question/answering track datasets are based on search engine queries, and TREC-9 datasets were actual users' questions. However, it is unclear whether they were gathered from mobile devices. From previous studies it is evident that mobile search logs have significant differences across many characteristics depending on the search medium and device [12, 13]. Queries from ChaCha are mobile (either voice or SMS) and the answers are constructed by a human and returned via SMS. Therefore, we elected to use the ChaCha dataset due to its high realism for query types and their distribution across topics in a mobile setting. We only use all 100,000 queries for our aggregate analysis of query length and topic distributions. Since our evaluation consists of both rewriting, and manual judgement of queries and responses, both of which are labor intensive, we were unable to perform detailed evaluation with the full set of 100,000 queries. Instead, a randomly selected 1,000 query subset is used for our more detailed manual evaluations. Out of these 1,000 queries, we found through manual analysis that 793 are long tailed.

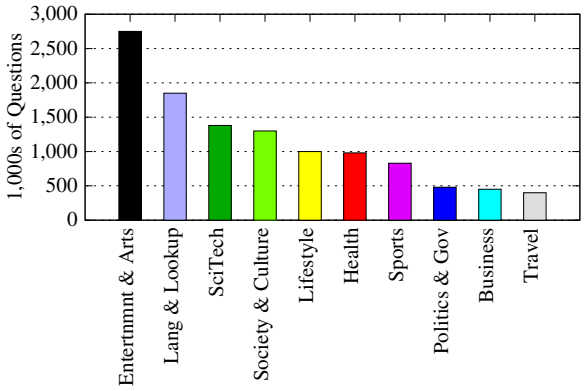

Figure 3: ChaCha number of questions per category as published on ChaCha's website [7]

6.2 Query Topics and Identifying the Long Tail

In most query log studies, the queries are categorized according to topic to give a representation of the types of queries being submitted. Although this type of categorization is useful for that purpose, for our evaluation these categories do not help in deciding whether queries are part of the long tail of queries we wish to evaluate. Figure 3 illustrates the published questions per category on ChaCha's website [38]. This system of categorization by topic is similar to previous studies [12, 13, 36]. From these categories it is unclear whether topics such as "Entertainment" and "Travel" could map to verticals or are too broad and should be directed to our system.

Further dividing topics into sub-topics (using ChaCha's published breakdowns) reveals that some of these finer granularity of sub-topics are directly mappable to verticals: e.g. "Yellow Pages" and "Definitions" may be mapped easily to data sources at "yellowpages.com" or "dictionary.com" with little effort. Other sub-topics are still too broad for a simple vertical (e.g. the sub-topics of "Politics & Gov." such as "Law" or "World Governments").

Each question from ChaCha has an associated set of topics it belongs to as assigned by ChaCha. Table 3 lists the top 10 most highly represented sub-topics that identified in

Table 3: ChaCha top sub-topics by percentage and existence of potential verticals

Topic	% of total Questions	Existing Vertical?
Celebrities	10.6%	no
Movies (not show times)	9.0%	no
Yellow Pages	8.9%	yes
Definitions	8.1%	yes
Music	8.5%	no
Relationships	7.1%	no
Food & Drink	5.4%	no
Conditions & Illness	4.8%	no
Animals & Plants	4.6%	no
Games	4.5%	no

Table 4: Example questions from ChaCha (including spelling errors)

Question
"who do think is gonna win the nba game tonight cavs vs magic"
"what does tipo mean in spanish"
"what is the difference between hot and cute in a girl"
"what is the purpose of a widows peak"
"does the subaru wrx still come in hatchbacks"
"can you list all beastie boys albums chronigically"
"when they give you the death penalty what do they in sect you with"
"what and when is the highest scoring basketball ever in the nba"
"why does the rainbow represent gayness"
"is there eternal life"

our sample (103 topics total), the percentage of questions belonging to these sub-topics, and whether there are corresponding verticals that are implemented in any existing SMS search systems. The sum of the percentages of the complete table do not add up to 100 because questions may belong to more than one topic. Certain topics such as "Celebrities" have online databases such as Internet Movie Database (IMDB) which could easily be used as a source of structured information to answer queries, and implementing verticals on top of these would not be difficult. Using the sub-topics, we removed queries from our 1,000 query corpus that do not have existing verticals in any existing automated SMS search system to identify the long tail queries (793 long tail queries).

These remaining queries are exactly long tail questions SMSFind is designed to answer. Several examples are listed in Table 4. The question distribution across sub-topics and the diversity of questions in general suggests that mobile user information needs even on low-end devices are more diverse than previously demonstrated [13]. This may be due to the more expressive input modality of voice or the user's perception of the system as being highly intelligent. We defer a more detailed comparison between these mobile search patterns and those in the literature to future studies.

Table 5: Summary of SMSFind results

Input, Output	% Correct
Mixed, Snippets	57.3%
Long Tail, Snippets	47.8%
Long Tail, N-Grams	22.9%
Long Tail (w/hint), TF-IDF Snippets	20.2%
Long Tail (w/out hint), TF-IDF Snippets	5.2%
Long Tail Unmodified Queries, Snippets	16.1%
Long Tail Unmodified Queries, N-Grams	6.7%

6.3 Baseline Evaluation

We conduct a few baseline evaluations of our system using our sample of 1,000 queries from ChaCha containing both vertical and long tail queries. We modify these queries assuming the user understands how to use the system and is willing to enter keywords along with a hint term rather than a natural language question. For each ChaCha query we rewrite the query solely by removing and reordering words given that we have knowledge of how the system works and know what would likely be a good choice for the hint term. As an example, "what are the symptoms of chicken pox" is rewritten as "chicken pox symptoms".

Using these queries we find that our system (including redirection to existing verticals) results in 57.3% correct answers. In comparison, Google SMS returns correct results for only 9.5% of these queries. We note that the low performance of the Google SMS could be due to a variety of reasons that we discuss in Section 7, and the result is provided here only for reference.

What is more interesting is if we remove the queries that are redirected to existing verticals. To focus on the core SMSFind algorithm, we consider only the performance of the SMSFind algorithm on the 793 long tail query subset. All further evaluation in this section is conducted with this set of 793 long tail queries. Table 5 summarizes the results of our evaluation. We find that for the long tail queries SMSFind returns 47.8% correct results. Furthermore, if we consider only the highest n-grams returned rather than the entire snippet, the performance drops to 22.9%. These results broadly indicate that the raw performance of SMSFind has significant room for improvement, and returning snippet answers generally results in better performance.

We observed earlier that there are issues with readability, but what do the snippet results actually look like? A representative set of examples is shown in Table 6 for both correct and incorrect results. Compared to the ChaCha human written responses we observe that the readability of our snippets is poor and could be improved; however, finding the optimal snippet construction is a separate problem where existing techniques could be applied [35].

6.4 Is Distillation Using N-grams Beneficial?

Since we are returning snippets rather than n-grams, it is natural to ask whether n-grams are necessary or if ranking snippets alone would perform just as well. To confirm that the intermediate distillation stage using n-grams is beneficial we compare against a simple per-term TF-IDF approach. In the first portion of this experiment we gather SMS sized

Table 6: Example questions with snippet answers returned by our system

Original Question	Example Snippet Answer	Judged Correct?
"how many packs of cigarettes are in a pack"	"to my contacts block user 10 packs and 200 cigarettes in a carton 3 years ago 0 rating good answer 0 rating bad answer report"	"yes"
"what does satine die from in moulin rouge"	"for a motion picture television or other visual media edit soundtrack main article moulin rouge music from baz luhrmann's film songs"	"no"
"what rush album does natural science come from"	"seventh album permanent waves and the last of their by the way that really does kick *ss cygnus-x1.net a tribute to rush album review"	"yes"
"what is the least expensive state to live in"	"the wholesale price which is adjusted quarterly cigarette tax several states are continuing to raise excise taxes on cigarettes and other"	"no"
"what is the world record for the most hot dogs eaten in ten minutes"	"contest results 2008 nathan's hot dog eating contest 10 minutes friday july 4 2008 no name hot dogs 1 tie joey jaws chestnut u.s"	"no"

snippets directly using a 140 byte sliding window across all documents. In this second portion of the experiment, we do the same thing except the snippets (d_j) are scored based on the sum of the standard TF-IDF scores as the metric for each of the i terms (t_i) of the query.

$$(tf\text{-}idf)_{i,j} = tf_{i,j} \times idf_i$$

where

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

With $n_{i,j}$ is the number of occurrences of the considered term t_i in snippet d_j, and the denominator is the sum of number of occurrences in all terms in snippet d_j. and

$$idf_i = log\frac{|D|}{|\{d : t_i \in d\}|}$$

With $|D|$: total number of snippets and
$|\{d : t_i \in d\}|$: number of snippets where the term t_i appears

We find that with only TF-IDF, the correct result is returned in only 5.2% of queries. The results of the per-snippet correctness are shown in Table 5 for comparison. With the hint term used in conjunction with TF-IDF 20.2% of queries are correctly returned. Overall, both naive TF-IDF and TF-IDF with a hint term return worse results than with the intermediate distillation using n-grams.

6.5 Returning Multiple Snippets

It is conceivable that for the queries that are not answered well, such as vague queries or explanations, slightly longer snippets or more results could improve our results. The application may allow multiple SMS messages to be requested in the form of a "more" link. To explore this possibility we experimented with returning multiple snippets. We compare two different snippet selection methods to improve result diversity. In both methods, we first order the snippets by rank. The first method is to return a snippet for each of the top ranked n-gram results. The second method returns only snippets for the top ranked n-gram result, but requires that the returned snippets are from different occurrences of the hint. We find that as more results are returned, the number of queries answered increases slightly (1 - 5%) for each additional response returned. Both methods show a similar rate of improvement, but the first method of maximizing n-gram diversity consistently performs better by approximately 10%.

6.6 How Important is the Hint Term?

One meta question we have considered briefly in our evaluation is: *can people actually come up with hints that are useful in the first place?* Requiring an additional hint term is "acceptable" since existing services by Google and Yahoo do exactly this, so our user interface is at least no worse than the norm. However, it is interesting to see how sensitive our system is to the existence of a well specified hint term.

We observe that if the hint term is not properly identified, and arbitrarily assigned as the last word in the query, the correctness of the snippets drops from 47.8% to 16.1% when compared to the modified queries. Table 5 shows the performance of our algorithm on our queries without any hint term identification or query modification for comparison. This suggests that the hint term is important to our system and a generally useful mechanism for focusing on the desired snippets of information.

7. DISCUSSION

The design of our system and the evaluation was motivated by the SMS search problem. Our goal was to understand how existing techniques would perform in this problem domain and not to supplant or reinvent the vast amounts of research in the Information Retrieval (IR) space.

7.1 Data Sets

We had initially experimented with Google Trends queries and other desktop search logs, but found that most queries were too ambiguous and did not match the type of queries found in mobile environments. We decided that the ChaCha dataset would provide more realism as to the actual queries people would make. To get a sense of the query difficulty and confirm that the ChaCha queries are actually "precision oriented" we performed a cursory evaluation based on work by Mothe and Tanguy [39] to evaluate the Syntactic Links Span and Average Polysemy Value of the queries in our corpus compared to those used in various TREC tracks. We

confirm that for these two features our mobile questions are significantly different from those in the question/answering track. The closest match for these two features was the set of "search query" format queries from the Million Query Track.

7.2 Difficult Types of Queries

In terms of the performance of our system, we observed that the queries that are not answered properly regardless of format are often ambiguous, explanations, enumerations, problems that require analysis, or time-sensitive queries. This is not surprising as many statistical techniques have similar limitations. Our algorithm in particular requires proximity to the hint term and we also expect the answer to be only a few words long. Queries that are ambiguous such as: "how many calories are in baked cod" are ambiguous in the sense that the human intelligence was required to assume that Long John Silver's baked cod is nearly equivalent to "baked cod". *Explanations* are likely to require more space to answer properly. We observe that queries such as "how do i make an antenna for my tv? i have a wire hanger" are unlikely to be answered within 140 bytes by even a human intelligence. *Enumerations* such as "words that rhyme with long" are also difficult for SMSFind since the answer is unlikely to all appear consecutively near any hint term in a significant number of places on web pages. *Analysis* questions (e.g. "what is the least expensive state to live in") are problematic since the answer is not immediately available in the text of web pages. *Time sensitive* information is difficult for a combination of lack of web page sources and proximity to a useful hint term. Given the ubiquity of these difficult query types, it is somewhat surprising that our simple algorithm is able to answer over half of the dataset.

7.3 Comparison to Existing Systems

The inclusion of Google SMS performance on our data set is only to illustrate that the information needs of people using only voice and SMS to ask questions may not be captured by that particular system of verticals. A direct comparison between the two numbers is unreasonable since we have no information as to the distribution of queries Google SMS receives and how that compares to our test set. It is entirely possible that the distribution of Google SMS queries is different, and their verticals successfully satisfy a large percentage of queries received by their service.

7.4 Pilot Deployment

We deployed our system with a small focus group of 40 users consisting of both urban slum residents and college students in Nairobi, Kenya. Our pilot lasted for two weeks during which users with low-end mobile phones and no data plan found the service to be the most useful. Verticals were not initially implemented in our pilot, and SMSFind was tasked with answering all queries. We found in our initial feedback session that people requested local business contact information and other simple verticals which were not being answered properly by SMSFind. To address this, we implemented several verticals to increase query coverage.

After the inclusion of verticals with specific keywords, we found that our users took time adjusting to the syntax of requiring the hint/topic at a specific position of the query. We are currently exploring possible improvements to both the user interface and performance, and we expect to extend our scale of deployment over the next few months.

7.5 NLP and Mobile Craigslist

Combining statistical and linguistic techniques in the context of answering questions using the web has been thoroughly surveyed by Lin et. al. [40]. As part of future work, we plan to incorporate more sophisticated statistical and NLP techniques. Specifically within the NLP summarization literature, there are probabilistic models which learn how important various factors of a given shingle are for text summarization [26]. One fundamental distinction in SMSFind in comparison to NLP techniques is that the corpus varies as a function of the search query; given a query, the corpus for SMSFind is the search pages returned by a search engine. Given a highly variable corpus, directly applying existing NLP-based probabilistic models may not be appropriate since the model should be a function of the query.

To better understand this contrast, we briefly describe the problem of SMS-based Mobile Craigslist where the goal is to provide SMS-based search responses for Craigslist queries. This is a problem we are currently exploring where probabilistic models derived from NLP are directly applicable since there exists a well-defined corpus for any given search category (e.g. cars, hotels, apartments). Using this corpus, we can use standard NLP techniques to learn the various features for each category and derive a succinct summary of each Craigslist posting for a category based on the features. Hence, having a well-defined corpus substantially helps in improving the accuracy of summarization.

8. CONCLUSION

There has been little work on SMS-based search for arbitrary topics due to, until recently, the initial lack of a well-defined business case. The explosive growth in prevalence of affordable low-end mobile devices throughout the world has created a large market for mobile information services. Since mobile users in many parts of the world use low-end mobile devices with SMS as their primary data transport, SMS-based search becomes a critical problem to address on the path to enabling SMS-based services.

In this paper, we have presented SMSFind, an automated SMS-based search response system that is tailored to work across arbitrary topics. We find that a combination of simple Information Retrieval algorithms in conjunction with existing search engines can provide reasonably accurate search responses for SMS queries. Using queries across arbitrary topics from a real-world SMS question/answering service with human-in-the-loop responses, we show that SMSFind is able to answer 57.3% of the queries in our test set. Although more powerful IR and NLP techniques are bound to improve performance, this work represents a foray into an open and practical research domain.

Acknowledgements

We would like to thank our shepherd Moustafa Youssef and the anonymous reviewers for their detailed comments which has helped in significantly improving our paper. We would also like to thank Jonathan Ledlie and several other members of Nokia Research at Cambridge, Palo Alto and Nairobi for their help with this project. Brendan Linn was a part of this project during its early stages and contributed to the early design. This project was partially funded by National Science Foundation under the CNS-NECO-0831934 award and Nokia Research.

9. REFERENCES

[1] Twitter. http://www.twitter.com/.

[2] SMS GupShup. http://www.smsgupshup.com/.

[3] Critical Mass: The Worldwide State of the Mobile Web. *The Nielsen Company*, 2008.

[4] Pump Up The Volume: An Assessment of Voice-Enabled Web Search on the iPhone. http://www.mcubedigital.com/msearchgroove/.

[5] Google SMS. http://www.google.com/sms.

[6] Yahoo One Search. http://mobile.yahoo.com/onesearch.

[7] ChaCha. http://www.chacha.com.

[8] T. Sohn, K.A. Li, W.G. Griswold, and J.D. Hollan. A diary study of mobile information needs. In *Proceedings of the 26th annual SIGCHI conference on Human factors in computing systems*, 2008.

[9] Esoko. http://www.esoko.com/.

[10] Google Squared. http://www.google.com/squared/.

[11] Text REtrieval Conference(TREC). http://trec.nist.gov/.

[12] M. Kamvar and S. Baluja. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 701–709, 2006.

[13] M. Kamvar, M. Kellar, R. Patel, and Y. Xu. Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices. In *Proceedings of the 18th international conference on World wide web*, pages 801–810, 2009.

[14] Global - Key Telecoms, Mobile and Broadband Statistics. *www.budde.com.au*, 2009.

[15] Windows Live Mobile. http://home.mobile.live.com/.

[16] Just Dial. http://www.justdial.com.

[17] D. Harman. Overview of the first text retrieval conference (TREC-1). In *First Text Retrieval Conference (Trec-1): Proceedings*, page 1, 1993.

[18] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, page 397, 2006.

[19] H.T. Dang, D. Kelly, and J. Lin. Overview of the TREC 2007 question answering track. In *Proceedings of TREC*, 2007.

[20] A. Singhal and M. Kaszkiel. A case study in web search using TREC algorithms. In *Proceedings of the 10th international conference on World Wide Web*, page 716, 2001.

[21] C. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 terabyte track. In *Proceedings of the 13th Text REtrieval Conference, Gaithersburg, USA*, 2004.

[22] J. Allan, B. Carterette, J.A. Aslam, V. Pavlu, B. Dachev, and E. Kanoulas. Million query track 2007 overview, 2007.

[23] E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, page 264. Association for Computational Linguistics, 2002.

[24] C.L.A. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra, and P. Tilker. Statistical selection of exact answers (MultiText experiments for TREC 2002). In *Proceedings of TREC*, pages 823–831, 2002.

[25] M.M. Soubbotin and S.M. Soubbotin. Use of patterns for detection of answer strings: A systematic approach. In *Proceedings of TREC*, volume 11. Citeseer, 2002.

[26] C. Aone, M.E. Okurowski, and J. Gorlinsky. Trainable, scalable summarization using robust NLP and machine learning. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 62–66. Association for Computational Linguistics, 1998.

[27] C.Y. Lin. Training a selection function for extraction. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 55–62. ACM, 1999.

[28] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *ACM Transactions on Information Systems*, volume 20, pages 116–131, 2002.

[29] R. Kraft, C.C. Chang, F. Maghoul, and R. Kumar. Searching with context. In *Proceedings of the 15th international conference on World Wide Web*, pages 477–486, 2006.

[30] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[31] S. Lawrence. Context in web search. In *IEEE Data Engineering Bulletin*, volume 23, pages 25–32, 2000.

[32] Linguistic Data Consortium. http://www.ldc.upenn.edu.

[33] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[34] Kannel. http://www.kannel.org/.

[35] T. Kanungo and D. Orr. Predicting the readability of short web summaries. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 202–211, 2009.

[36] J. Yi, F. Maghoul, and J. Pedersen. Deciphering mobile search patterns: a study of yahoo! mobile search queries. In *Proceedings of the 17th international conference on World Wide Web*, 2008.

[37] TREC Question Answering Track. http://trec.nist.gov/data/qamain.html.

[38] ChaCha Categories. http://www.chacha.com/categories.

[39] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty-a case study on previous trec campaigns. In *SIGIR workshop on Predicting Query Difficulty-Methods and Applications*, pages 7–10, 2005.

[40] J. Lin and B. Katz. Question answering techniques for the World Wide Web. *EACL-2003 Tutorial*, 2003.