CrowdSearch: Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones

Tingxin Yan, Vikas Kumar, Deepak Ganesan
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
{yan, vikas, dganesan}@cs.umass.edu

ABSTRACT

Mobile phones are becoming increasingly sophisticated with a rich set of on-board sensors and ubiquitous wireless connectivity. However, the ability to fully exploit the sensing capabilities on mobile phones is stymied by limitations in multimedia processing techniques. For example, search using cellphone images often encounters high error rate due to low image quality.

In this paper, we present CrowdSearch, an accurate image search system for mobile phones. CrowdSearch combines automated image search with real-time human validation of search results. Automated image search is performed using a combination of local processing on mobile phones and backend processing on remote servers. Human validation is performed using Amazon Mechanical Turk, where tens of thousands of people are actively working on simple tasks for monetary rewards. Image search with human validation presents a complex set of tradeoffs involving energy, delay, accuracy, and monetary cost. CrowdSearch addresses these challenges using a novel predictive algorithm that determines which results need to be validated, and when and how to validate them. CrowdSearch is implemented on Apple iPhones and Linux servers. We show that CrowdSearch achieves over 95% precision across multiple image categories, provides responses within minutes, and costs only a few cents.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; H.4.0 [**Information System Applications**]: General

General Terms

Algorithms, Human Factors, Performance

Keywords

Crowdsourcing, Human validation, Image search, Real time.

1. INTRODUCTION

Mobile phones are becoming increasingly sophisticated with a rich set of on-board sensors and ubiquitous wireless connectivity. One of the primary advantages of ubiquitous internet access is the ability to search anytime and anywhere. It is estimated that more than 70% of smart phone users perform internet search [18]. The growth rate in mobile internet search suggests that the number of search queries from phones will soon outstrip all other computing devices.

Search from mobile phones presents several challenges due to their small form-factor and resource limitations. First, typing on a phone is cumbersome, and has led to efforts to incorporate on-board sensors to enable multimedia search. While the use of GPS and voice for search is becoming more commonplace, image search has lagged behind. Image search presents significant challenges due to variations in lighting, texture, type of features, image quality, and other factors. As a result, even state-of-art image search systems, such as Google Goggle [21], acknowledge that they only work with certain categories of images, such as buildings. Second, scrolling through multiple pages of search results is inconvenient on a small screen. This makes it important for search to be *precise* and generate few erroneous results. Third, multimedia searches, particularly those using images and video clips, require significant memory, storage, and computing resources. While remote servers, for example cloud computing infrastructures [6], can be used to perform search, transferring large images through wireless networks incurs significant energy cost.

While automated image search has limitations in terms of accuracy, humans are naturally good at distinguishing images. As a consequence, many systems routinely use humans to tag or annotate images (e.g. Google Image Labeler [16]). However, these systems use humans for tagging a large corpus of image data over many months, whereas search from phones needs fast responses within minutes, if not seconds.

In this paper, we present CrowdSearch, an accurate image search system for mobile phones. CrowdSearch combines automated image search with *real-time* human validation of search results. Automated image search uses a combination of local processing on mobile phones and remote processing on powerful servers. For a query image, this process generates a set of candidate search results that are packaged into tasks for validation by humans. Real-time validation uses the Amazon Mechanical Turk (AMT), where tens of thousands of people are available to work on simple tasks for monetary rewards. Search results that have been validated are returned to the user.

The combination of automated search and human validation presents a complex set of tradeoffs involving delay, accuracy, monetary cost, and energy. From a crowdsourcing perspective, the key tradeoffs involve delay, accuracy and cost. A single validation response is often insufficient since

MobiSys'10, June 15–18, 2010, San Francisco, California, USA.

1) humans have error and bias, and 2) delay may be high if the individual person who has selected the task happens to be slow. While using multiple people for each validation task and aggregating the results can reduce error and delay, it incurs more monetary cost. From an automated image search perspective, the tradeoffs involve energy, delay, and accuracy. Exploiting local image search on mobile phones is efficient energy-wise but the resource constraints on the device limits accuracy and increases delay. In contrast, remote processing on powerful servers is fast and more accurate, but transmitting large raw images from a mobile phone consumes time and energy.

CrowdSearch addresses these challenges using three novel ideas. First, we develop accurate models of the delay-accuracy-cost behavior of crowdsourcing users. This study is a first-of-its-kind, and we show that simple and elegant models can accurately capture the behavior of these complex systems. Second, we use the model to develop a predictive algorithm that determines which image search results need to be validated, when to validate the tasks, and how to price them. The algorithm dynamically makes these decisions based on the deadline requirements of image search queries as well as the behavior observed from recent validation results. Third, we describe methods for partitioning of automated image search between mobile phones and remote servers. This technique takes into account connectivity states of mobile phones (3G vs WiFi) as well as search accuracy requirements.

The CrowdSearch system is implemented on Apple iPhones and Linux servers. Our results show that:

- A combination of automated search and human validation achieves over 95% precision across a range of image types including faces, flowers, books and buildings. These categories cover a spectrum from good to bad performance for automated search.
- We derive arrival and inter-arrival models that accurately predict the delay of responses from human validators. To our knowledge, this is the first attempt at modeling delay behavior of a crowdsourcing system.
- CrowdSearch balances accuracy and monetary cost while achieving user-specified deadlines for responses to search queries. We show that CrowdSearch 1) saves monetary cost by up to 50% in comparison with non-adaptive schemes, 2) incurs only 5-6% larger delay in comparison to a delay-optimal scheme, and 3) provides over 98% search precision across different deadlines.
- We show that dynamic partitioning of search functionality across mobile phones and remote server saves overall energy consumption up to 70%.

2. SYSTEM ARCHITECTURE

CrowdSearch comprises three components: (i) the mobile phone, which initiates queries, displays responses, and performs local image processing, (ii) a powerful remote server, or cloud computing backend, which performs automated image search, and triggers image validation tasks, and (iii) a crowdsourcing system that validates image search results.

System Operation: CrowdSearch requires three pieces of information prior to initiating search: (a) an image query, (b) a query deadline, and (c) a payment mechanism for human validators. There are several options for each of these components. An image query may be either a single image

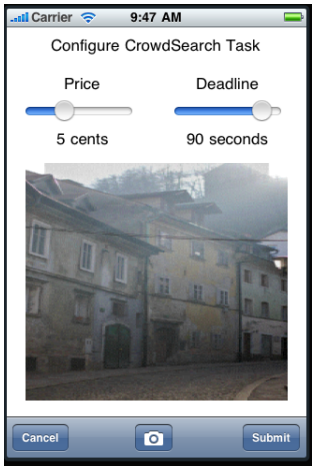


Figure 1: An iPhone interface for CrowdSearch system. A building image is captured by user and sent to CrowdSearch as a search query. Users can specify the price and deadline for this query.

or accompanied by other modalities, such as GPS location or text tags. The deadline can be either provided explicitly by the user using a micro-payment account such as Paypal, or a search provider, such as Google Goggle, who pays for human validation to improve performance and attract more customers. We defer a more detailed discussion of some of these options to §8. In the rest of this paper, we assume that a search query comprises solely of an image, and is associated with a deadline as well as payment mechanism. Given such a query, CrowdSearch attempts to return at least one correct response prior to the deadline while minimizing monetary cost.

The major operations of CrowdSearch comprises of two steps. The first step is processing the image query using an automatic image search engine. The search engine searches through a database of labelled images and returns a ranked list of candidate results. Despite the fact that search is initiated by images from mobile phones, the database of labelled images can include images obtained from diverse sources including FlickR or Google. The downside of automated image search is that many results may be incorrect, particularly for non-planar objects such as faces.

The second step in our system is the CrowdSearch algorithm that uses a crowdsourcing system to validate the candidate results. In particular, we use the Amazon Mechanical Turk (AMT), since it has APIs for automatic task posting, and has a large user base of tens of thousands of people. Human validation is simple — for each <query image, candidate image> pair, a human validator just clicks on YES if they match and NO if they don't match. Only thumbnails of the images are sent for validation to minimize communication required from the phone. In return for the answer, the validator is paid a small sum of money as reward (one or few cents). The CrowdSearch algorithm considers several tradeoffs while determining how candidate images are validated. Since human validation consumes monetary cost, it attempts to minimize the number of human validators that need to be paid to get a good response for a user query. However, CrowdSearch also needs to take into account the deadline requirements as well as the need to remove hu-

man error and bias to maximize accuracy. To balance these tradeoffs, CrowdSearch uses an adaptive algorithm that uses delay and result prediction models of human responses to judiciously use human validation. Once a candidate image is validated, it is returned to the user as a valid search result.

3. CROWDSOURCING FOR SEARCH

In this section, we first provide a background of the Amazon Mechanical Turk (AMT). We then discuss several design choices that we make while using crowdsourcing for image validation including: 1) how to construct tasks such that they are likely to be answered quickly, 2) how to minimize human error and bias, and 3) how to price a validation task to minimize delay.

Background: We now provide a short primer on the AMT, the crowdsourcing system that we use in this work. AMT is a large-scale crowdsourcing system that has tens of thousands of validators at any time. The key benefit of AMT is that it provides public APIs for automatic posting of tasks and retrieval of results. The AMT APIs enable us to post tasks and specify two parameters: (a) the number of duplicates, *i.e.* the number of independent validators who we want to work on the particular task, and (b) the reward that a validator obtains for providing responses. A validator works in two phases: (a) they first accept a task once they identify that they would like to work on it, which in turn decrements the number of available duplicates, and (b) once accepted, they need to provide a response within a period specified by the task.

One constraint of the AMT that pertains to CrowdSearch is that the number of duplicates and reward for a task that has been posted cannot be changed at a later point. We take this practical limitation in mind in designing our system.

Constructing Validation Tasks: How can we construct validation tasks such that they are answered quickly? Our experience with AMT revealed several insights. First, we observed that asking people to tag query images and candidate images directly is not useful since: 1) text tags from crowdsourcing systems are often ambiguous and meaningless (similar conclusions have been reported by other crowdsourcing studies [8]), and 2) tasks involving tagging are unpopular, hence they incur large delay. Second, we found that having a large validation task that presents a number of <query image, candidate image> pairs enlarges human error and bias since a single individual can bias a large fraction of the validation results.

We settled on an a simple format for validation tasks. Each <query image, candidate image> pair is packaged into a task, and a validator is required to provide a simple YES or NO answer: YES if the two images are correctly matched, and NO otherwise. We find that these tasks are often the most popular among validators on AMT.

Minimizing Human Bias and Error: Human error and bias is inevitable in validation results, therefore a central challenge is eliminating human error to achieve high accuracy. We use a simple strategy to deal with this problem: we request several duplicate responses for a validation task from multiple validators, and aggregate the responses using a majority rule. Since AMT does not allow us to dynamically change the number of duplicates for a task, we fix this number for all tasks. In §7.2, we evaluate several aggregation approaches, and show that a majority of five duplicates is the best strategy and consistently achieves us more than 95% search accuracy.

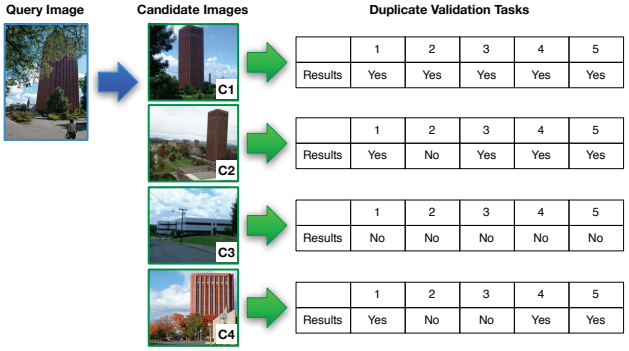


Figure 2: Shown are an image search query, candidate images, and duplicate validation results. Each validation task is a Yes/No question about whether the query image and candidate image contains the same object.

Pricing Validation Tasks: Crowdsourcing systems allow us to set a monetary reward for each task. Intuitively, a higher price provides more incentive for human validators, and therefore can lead to lower delay. This raises the following question: *is it better to spend X cents on a single validation task or to spread it across X validation tasks of price one cent each?* We find that it is typically better to have more tasks at a low price than fewer tasks at a high price. There are three reasons for this behavior: 1) since a large fraction of tasks on the AMT offer a reward of only one cent, the expectation of users is that most tasks are quick and low-cost, 2) crowdsourcing systems like the AMT have tens of thousands of human validators, hence posting more tasks reduces the impact of a slow human validator on overall delay, and 3) more responses allows better aggregation to avoid human error and bias. Our experiments with AMT show that the first response in five one cent tasks is 50 - 60% faster than a single five cent task, confirming the intuition that delay is lower when more low-priced tasks are posted.

4. CROWDSEARCH ALGORITHM

Given a query image and a ranked list of candidate images, the goal of human validation is to identify the correct candidate images from the ranked list. Human validation improves search accuracy, but incurs monetary cost and human processing delay. We first discuss these tradeoffs and then describe how CrowdSearch optimizes overall cost while returning at least one valid candidate image within a user-specified deadline.

4.1 Delay-Cost Tradeoffs

Before presenting the CrowdSearch algorithm, we illustrate the tradeoff between delay and cost by discussing posting schemes that optimize one or the other but not both.

Parallel posting to optimize delay: A scheme that optimizes delay would post all candidate images to the crowdsourcing system at the same time. (We refer to this as *parallel posting*.) While parallel posting reduces delay, it is expensive in terms of monetary cost. Figure 2 shows an instance where the image search engine returns four candi-

date images, and each candidate image is validated by five independent validators. If each of these validators is paid a penny, the overall cost for validating all responses would be 20 cents, a steep price for a single search. Parallel posting is also wasteful since it ignores the fact that images with higher rank are more likely to be better matches than those lower-ranked ones. As a result, if the first candidate image is accurate, the rest of the candidates need not to be posted.

Serial posting to optimize cost: In contrast to parallel posting, a scheme that optimizes solely the monetary cost would post tasks *serially*. A serial posting scheme first posts the top-ranked candidate for validation, and waits to see if the majority of validators agree that it is a positive match. If so, the process ends and returns a positive match, otherwise the next candidate is posted, and so on. This scheme uses the least number of tasks to find the first correct match, and thus costs considerably less than the parallel posting scheme. Taking Figure 2 as an example, the serial posting scheme would just cost five cents, since the first candidate is a valid match. However, in cases where top-ranked image is incorrect, serial posting incurs much higher delay than parallel posting.

Clearly, parallel and serial posting are two extreme schemes that sacrifice either cost or delay. Instead, we propose an algorithm that achieves a tradeoff between these two metrics.

4.2 Optimizing Delay and Cost

Key Insight: CrowdSearch tries to provide a balance between serial and parallel schemes. More precisely, the goal is to minimize monetary cost while ensuring that *at least one valid candidate*, if present in the ranked list returned from the search engine, is provided to the user within a user specified deadline. If all candidates images are incorrect, no response is returned to the user. For example, in Figure 2, candidates C_1, C_2, and C_4 are all valid responses, hence it is sufficient to validate any one of them and return it as a response to the mobile user. Intuitively, such an objective also fits well with image search on mobile phones, where few good results are more appropriate for display.

To illustrate the main idea in CrowdSearch, consider the case where we have one query image and five candidate images returned by the image search engine. Assume that we posted the top-ranked candidate, C_1, as a validation task at time $t = 0$ and the user-specified deadline is time $t = 2\ mins$. At time $t = 1\ min$, say that two responses have been received for this task, a Yes and a No (i.e. sequence 'YN'). Within the next 1 *min*, there are multiple possible incoming responses, such as 'YY', 'YNY', and others, that can lead to a positive validation result under majority(5) rule. The CrowdSearch algorithm estimates the probability that any one of these valid sequences occurs during the next minute. This estimation is done by using: 1) models of inter-arrival times of responses from human validators, and 2) probability estimates for each sequence that can lead to a positive validation result. If the probability of a valid result within the deadline is less than a pre-defined threshold P_{TH}, for instance 0.6, CrowdSearch posts a validation task for the next candidate image to improve the chance of getting at least one correct match. The above procedure is performed at frequent time-steps until either the deadline is reached or all candidates have been posted for validation.

The Algorithm Having explained the key insight, we now present the complete CrowdSearch prediction algorithm as shown in Algorithm 1. This algorithm handles all the ongoing tasks, each task has received partial results from validators. For each task T_x, let S_i denote the partial sequence received. For instance, in the example above, two results of 'YN' have been received. The algorithm computes the probability that at least one of the ongoing tasks is going to have a positive answer under the majority(N) rule, where N is the number of duplicates for each validation task. For each task T_x, the CrowdSearch algorithm traverses all possible sequence of results S_j that begin with the received sequence S_i. If the sequence S_j leads to a positive answer, CrowdSearch predicts the probability that the remaining results can be received before the deadline, as shown in line 4-6 in algorithm 1. Here CrowdSearch calls two functions, *DelayPredict* and *ResultPredict*. The former function estimates the probability that sequence S_j will be received before the deadline, and the latter estimates the probability that the sequence S_j will occur given that sequence S_i has been received so far. We assume that these two probabilities are independent to each other since the delay and content of a result sequence have no clear causal relation. Thus, the product of the two, P_j, is the probability that the sequence S_j is received prior to the deadline. We compute P_+, which is the accumulation of P_j for all cases where the remaining sequence leads to a positive answer. This gives us the predicted probability that current task can be validated as positive given S_i results are received. Having predicted the probability for a single validation task, now we consider the case where multiple validation tasks are concurrently ongoing for a search query. Since our goal is to return at least one correct candidate image, we first compute P_{fail}, which is the probability that none of the ongoing tasks are correct. Hence $P_{suc} = 1 - P_{fail}$ is the probability we want to compute. P_{fail} is the product of the probability that each of the ongoing task fails, since all tasks are independent to each other. We compute P_{fail} as shown in line 13 in the algorithm 1. If P_{suc} is less than a pre-defined threshold P_{TH}, we post the next task, else we wait for current tasks. In the rest of this section, we discuss *DelayPredict* and *ResultPredict* that are central to our algorithm.

Algorithm 1 CrowdSearch()

```
1: P_fail ← 1
2: for each ongoing task T_x do
3:    S_i ← sequence of results received
4:    P_+ ← 0
5:    for each S_j that starts with S_i do
6:       P_delay ← DelayPredict(|S_i|, |S_j|)
7:       P_results ← ResultPredict(S_i, S_j)
8:       P_j ← P_results × P_delay
9:       if Majority(S_j) = Yes then
10:         P_+ ← P_+ + P_j
11:      end if
12:   end for
13:   P_fail ← P_fail × (1 − P_+)
14: end for
15: P_suc ← 1 − P_fail
16: if P_suc ≥ P_TH then
17:    Return True
18: else
19:    Return False
20: end if
```

4.3 Delay Prediction Model

We now consider the problem of predicting the time that sequence S_j is received given a partial sequence S_i has been received. This problem can be split into the sum of inter-arrivals from S_i to S_{i+1}, S_{i+1} to S_{i+2}, and so on till S_{j-1} to S_j. We start with a model of the inter-arrival delay between any two responses obtained from validators, and then use the inter-arrival models to determine the overall delay.

4.3.1 Delay modeling

We consider two cases, the first is when no responses have been received so far *i.e.* when $i = 0$, and the second when one or more responses have been received *i.e.* when $i > 0$. The two cases need to be handled differently since in the former case we model the delay for the arrival of the first response, whereas in the latter case we model the inter-arrival times between responses.

Case 1 - Delay for the first response: The delay of an AMT validator can be separated into two parts: *acceptance delay* and *submission delay*. Acceptance delay means the time from the validation task being posted to being accepted by a validators. Submission delay means the time from the task being accepted to being submitted by a validator. The total delay is the summation of acceptance delay and submission delay.

Figure 3(a) shows the complementary cumulative distribution function(CCDF) of our delay measurements over more than 1000 tasks. Note that the y-axis is in log scale. It is clear that both acceptance delay and submission delay follow exponential distributions with a small delay offset. The exponential behavior follows intuition due to the law of independent arrivals, and the fact that there are tens of thousands of individuals on AMT. The offset for acceptance delay results from delay in refreshing the task on the web browser, internet connection delays, and time taken by the validator to search through several thousands of tasks to locate ours. The offset for working time results from the time to read the task, input the result, and submit the answer.

Let λ_a and λ_s to represent the rate of acceptance and submissions, and c_a and c_s represent the offsets of acceptances and submissions. Their probability density function, denoted as $f_a(t)$ and $f_s(t)$, are

$$f_a(t) = \lambda_a e^{-\lambda_a(t-c_a)}$$

and

$$f_s(t) = \lambda_s e^{-\lambda_s(t-c_s)}$$

How can we obtain the overall delay distribution, which is sum of acceptance and submission delay? We first determine whether the two delays are independent of each other. Intuitively, the two should be independent since the acceptance delay depends on when a validator chanced upon our validation tasks, whereas submission delay depends on how fast the validators works on tasks. To confirm independence, we compute the Pearson's product-moment coefficient between the two delays. Our results indicate that the coefficient score is less than 0.05, which means there is no linear relation between acceptance and submission delay.

Given independence, the PDF of overall delay can be computed as the convolution of the PDFs of acceptance delay and submission delay. Let $f_o(t)$ denote the PDF of overall delay we have:

$$\begin{aligned} f_o(t) &= f_a(t) * f_s(t) \\ &= \int_{c_s}^{t-c_a} [\lambda_a e^{-\lambda_a((t-u)-c_a)}][\lambda_s e^{-\lambda_s(u-c_s)}]du \\ &= \frac{\lambda_a\lambda_s}{\lambda_a-\lambda_s}(e^{-\lambda_s(t-(c_a+c_s))} - e^{-\lambda_a(t-(c_a+c_s))}) \end{aligned}$$

Let $c = c_a + c_s$. The above expression can be written as:

$$f_o(t) = \frac{\lambda_a\lambda_s}{\lambda_a-\lambda_s}(e^{-\lambda_s(t-c)} - e^{-\lambda_a(t-c)}) \quad (1)$$

Case 2 - Inter-arrival delay between responses: Figure 3(b) shows the CCDF of inter-arrival delay between validation tasks. These delays are clearly a set of exponential distributions with different rates. The constant offsets are counteracted since we are looking at the difference between arrivals. Thus, the PDF of inter-arrival time between response i and $i+1$, denoted as $f_i(t)$, can be modelled as an exponential function:

$$f_i(t) = \lambda_i e^{-\lambda_i t} \quad (2)$$

4.3.2 Delay Prediction

Given the arrival time for the first response and the inter-arrival times between responses, we can predict delay as follows. Consider the example shown in Figure 4. Suppose we have received two response at 60 seconds, and the current time to the origin of the query is 75 seconds. At this time, we want to predict the probability that five responses are received within the remaining 45 seconds to the deadline. Let X_j denote the time of receiving the j_{th} response, and $\hat{X_i}$ denote the tuple $X_1, X_2, \cdots, X_i$, and $\hat{t_i}$ denote the tuple $t_1, t_2, \cdots, t_i$, where t_i is the time of receiving the i^{th} response. Let D denote the deadline, t denote the current time, All times are relative to the beginning of the query. The probability we want to estimate is:

$$P_{ij} = P(X_j \leq D \mid X_{i+1} \geq t, \hat{X_i} = \hat{t_i}) \quad (3)$$

We consider this problem from the perspective of inter-arrival times between responses. Let $Y_{i,j}$ denote the inter-arrival time between the response i and j. This inter-arrival time is the sum of a set of inter-arrival times between adjacent responses:

$$Y_{i,j} = \sum_{k=i}^{j-1} Y_{k,k+1}$$

Therefore, P_{ij} can be presented as:

$$P_{ij} = P(Y_{i,j} \leq D - t_i \mid Y_{i,i+1} \geq t - t_i, \hat{X_i} = \hat{t_i})$$

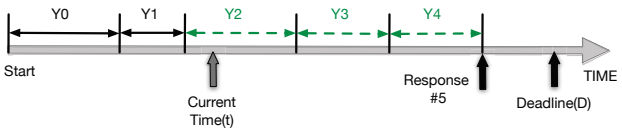
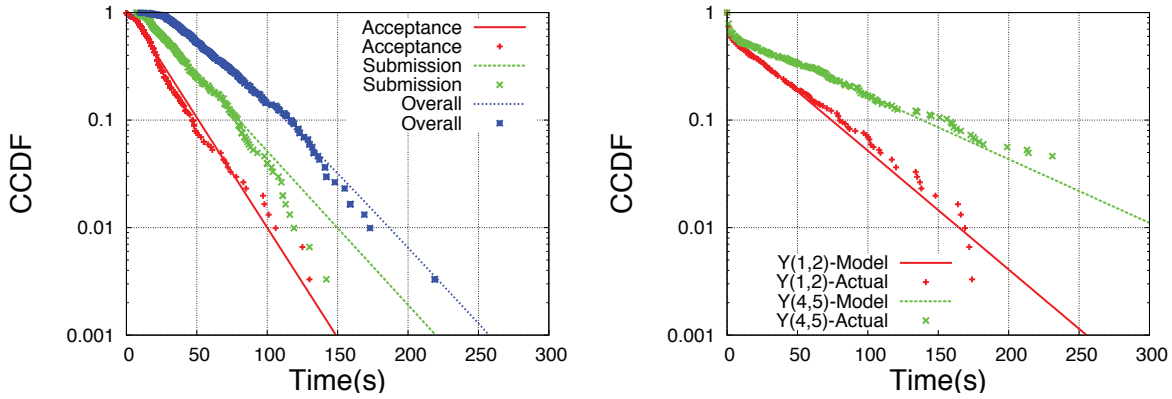


Figure 4: An example of predicting the delay of 5^{th} response given the time of 2^{nd} response. The delay is the summation of the inter-submission delay of Y_2, Y_3 and Y_4.



(a) Overall delay model

(b) Inter-arrival delay model

Figure 3: Delay models for overall delay and inter-arrival delay. The overall delay is decoupled with acceptance and submission delay.

From our inter-arrival delay model, we know that all inter-arrival times are independent. Thus, we can present the probability density function of $Y_{i,j}$ as the convolution of the inter-arrival times of response pairs from i to j.

Before applying convolution, we first need to consider the condition $Y_{i,i+1} \geq t - t_i$. This condition can be removed by applying the law of total probability. We sum up all the possible values for $Y_{i,i+1}$, and note that the lower bound is $t - t_i$. For each $Y_{i,i+1} = t_x$, the rest part of $Y_{i,j}$, or $Y_{i+1,j}$, should be in the range of $D - t_i - t_x$. Thus the condition of $Y_{i,i+1}$ can be removed and we have:

$$P_{ij} = \sum_{t_x=t-t_i}^{D-t_i} P(Y_{i,i+1} = t_x) P(Y_{i+1,j} \leq D - t_i - t_x) \quad (4)$$

Now we can apply the convolution directly to $Y_{i+1,j}$. Let $f_{i,j}(t)$ denote the PDF of inter-arrival between response i and j. The PDF of $Y_{i+1,j}$ can be expressed as:

$$f_{i+1,j}(t) = (f_{i+1,i+2} * \cdots * f_{j-1,j})(t)$$

Combining this with Equation 4, we have:

$$P_{ij} = \sum_{t_x=t-t_i}^{D-t_i} (f_{i,i+1}(t_x) \sum_{t_y=0}^{D-t_i-t_x} f_{i+1,j}(t_y)) \quad (5)$$

Now the probability we want to predict has been expressed in the form of PDF of inter-arrival times. Our delay models capture the distribution of all inter-arrival times that we need for computing the above probability: we use the delay model for the first response when $i = 0$, and use the inter-arrival of adjacent response when $i > 0$. Therefore, we can predict the delay of receiving any remaining responses given the time that partial responses are received.

4.4 Predicting Validation Results

Having presented the delay model, we discuss how to predict the actual content of the incoming responses, *i.e.* whether each response is a Yes or No. Specifically, given that we have received a sequence S_i, we want to compute the probability of occurrence of each possible sequence S_j that starts with S_i, such that the validation result is positive, *i.e.*, $majority(S_j) = Yes$.

This prediction can be easily done using a sufficiently large training dataset to study the distribution of all possible result sequences. For the case where the number of duplicate is set to be 5, there are $2^5 = 32$ different sequence combinations. We can compute the probability that each sequence occurs in the training set by counting the number of their occurrences. We use this probability distribution as our model for predicting validation results. We use the probabilities to construct a probability tree called *SeqTree*.

Figure 5 shows an example of a SeqTree tree. It is a binary tree where leaf nodes are the sequences with length of 5. For two leaf nodes where only the last bit is different, they have a common parent node whose sequence is the common substring of the two leaf nodes. For example, nodes 'YNYNN' and 'YNYNY' have a parent node of 'YNYN'. The probability of a parent node is the summation of the probability from its children. Following this rule, the SeqTree is built, where each node S_i is associated with a probability p_i that its sequence can happen.

Given the tree, it is easy to predict the probability that S_j occurs given partial sequence S_i using the SeqTree. Simply find the nodes that correspond to S_i and S_j respectively, and the probability we want is p_j/p_i.

5. IMAGE SEARCH ENGINE

In this section, we briefly introduce the automated image search engine. Our search engine is designed using image search methods that have been described in the prior work including our own [30]. The fundamental idea of the image search engine is to use a set of compact image representations called visterms (visual terms) for efficient communication and search. The compactness of visterms makes them attractive for mobile image search, since they can be communicated from the phone to a remote search engine server at extremely low energy cost. However, extracting visterms from images consumes significant computation overhead and delay at the phone. In this section, we provide an overview of the image search engine, and focus on explaining the tradeoffs that are specific to using it on resource-constrained mobile phones.

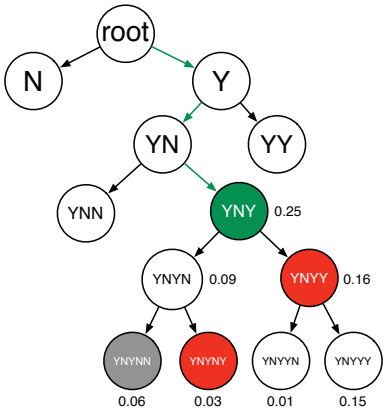


Figure 5: A SeqTree to Predict Validation Results. The received sequence is 'YNY', the two sequences that lead to positive results are 'YNYNY' and 'YNYY'. The probability that 'YNYY' occurs given receiving 'YNY' is 0.16/0.25 = 64%

5.1 Image Search Overview

The image search process contains two major steps: 1) extracting features from a query image, and 2) search through database images with features of query image.

Extracting features from query image: There are many good features to represent images, such as the Scale-Invariant Feature Transform (SIFT) [9]. While these features capture essential characteristics of images, they are not directly appropriate for search because of their large size. For instance, SIFT features are 128 dimensional vectors and there are several hundred such SIFT vectors for a VGA image. The large size makes it 1) unwieldy and inefficient for search since the data structures are large, and 2) inefficient for communication since no compression gains are achieved by locally computing SIFT features on the phone.

A canonical approach to reduce the size of features is to reduce the dimensionality by clustering. This is enabled by a lookup structure called "vocabulary tree" that is constructed in an a priori manner by hierarchical k-means clustering of SIFT features of a training dataset. For example, a vocabulary tree for buildings can be constructed by collecting thousands of training images, extracting their SIFT feature and using k-means clustering to build the tree. A vocabulary tree is typically constructed for each category of images, such as faces, buildings, or book covers.

Searching through database images: Once visterms are extracted from an image, they can be used in a manner similar to keywords in text retrieval [2]. The search process uses a data structure called the inverted index that is constructed from the corpus of images in the database. The inverted index is basically a mapping from each visterm to the images in the database containing that visterm. Each visterm is also associated with a inverted document frequency (IDF) score that describes its discriminating power. Given a set of visterms for a query image, the search process is simple: for each visterm, the image search engine looks up the inverted index and compute an IDF score for each of the candidate images. The list of candidates are returned ranked in order of their IDF score.

5.2 Implementation Tradeoffs

There are two key questions that arise in determining how to split image search functionality between the mobile phone and remote server. The first question is whether visterm extraction should be performed on the mobile phone or remote server. Since visterms are very compact, transmitting visterms from a phone as opposed to the raw image can save time and energy, particularly if more expensive 3G communication is used. However, visterm computation can incur significant delay on the phone due to its resource constraints. In-order to reduce this delay, one would need to tradeoff the resolution of the visterms, thereby impacting search accuracy. Thus, using local computation to extract visterms from a query image saves energy but sacrifices accuracy. Our system chooses the best option for visterm extraction depending on the availability of WiFi connectivity. If only 3G connectivity is available, visterm extraction is performed locally, whereas if WiFi connectivity is available, the raw image is transferred quickly over the WiFi link and performed at the remote server.

The second question is whether inverted index lookup should be performed on the phone or the remote server. There are three reasons to choose the latter option: 1) since visterms are already extremely compact, the benefit in performing inverted index lookup on the phone is limited, 2) having a large inverted index and associated database images on the phone is often not feasible, and 3) having the inverted index on the phone makes it harder to update the database to add new images. For all these reasons, we choose to use a remote server for inverted index lookup.

6. SYSTEM IMPLEMENTATION

The CrowdSearch system is implemented on Apple iPhones and a backend server at UMass. The components diagram of CrowdSearch system is shown in Figure 6.

iPhone Client: We designed a simple user interface for mobile users to capture query images and issue a search query. The screenshot of the user interface is shown in Figure 1. A client can provide an Amazon payments account to facilitate the use of AMT and pay for validation. There is also a free mode where validation is not performed and only the image search engine results are provided to the user.

To support local image processing on the iPhone, we ported an open-source implementation of the SIFT feature extraction algorithm [26] to the iPhone. We also implemented a vocabulary tree lookup algorithm to convert from SIFT features to visterms. While vocabulary tree lookup is fast and takes less than five seconds, SIFT takes several minutes to process a VGA image due to the lack of floating point support on the iPhone. To reduce the SIFT running time, we tune SIFT parameters to produce fewer SIFT features from an image. This modification comes at the cost of reduced accuracy for image search but reduces SIFT running time on the phone to less than 30 seconds. Thus, the overall computation time on the iPhone client is roughly 35 seconds.

When the client is connected to the server, it also receives updates, such as an updated vocabulary tree or new deadline recommendations.

CrowdSearch Server Implementation: The CrowdSearch Server is comprised of two major components: automated image search engine and validation proxy. The image search engine generates a ranked list of candidate images for each

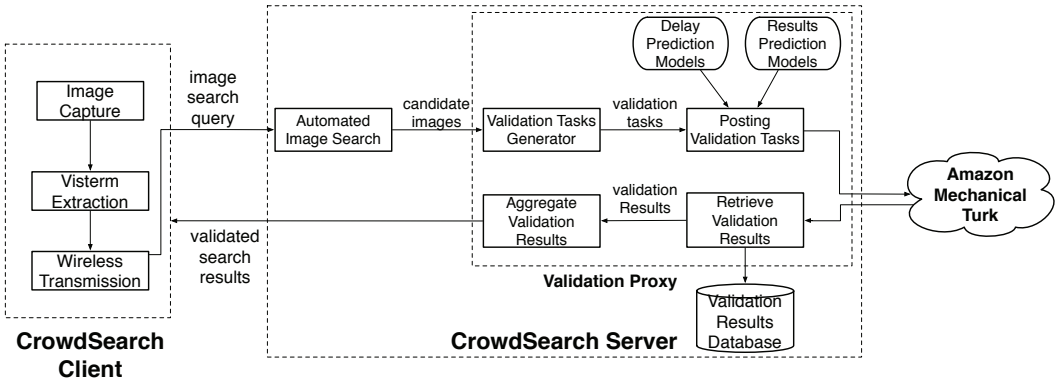


Figure 6: CrowdSearch Implementation Components Diagram

search query from iPhone clients. Our current implementation of the image search engine is about 5000 lines of C++ code, and we use hierarchical k-means clustering algorithms [13] to build this image search engine. Over 1000 images collected from several vision datasets [13, 11] are used as the training images, which cover four categories: human faces, flowers, buildings, and book covers.

The validation proxy is implemented with approximately 10,000 lines of Java and Python code. There are four major components in validation proxy as shown in Figure 6. The Validation Task Generator module converts each <query image, candidate image> pair to a human validation task. The Task Posting module posts validation tasks to AMT using the Amazon Web Service(AWS) API. The results are retrieved by the Retriever module, which also performs some bookkeeping such as approval of tasks, and payment to the AMT solvers. These operations are implemented with AWS API as well. The Result aggregation module generates an appropriate responses from validators' answers by using a majority rule, and returns the final validated search results back to the mobile user.

7. EXPERIMENTAL EVALUATION

We evaluate CrowdSearch over several thousand images, many tens of days of traces on the Amazon Mechanical Turk, and an end-to-end evaluation with a complete system running on mobile phones and a backend server. We evaluate four aspects of the performance of CrowdSearch : (a) improvement in image search precision, (b) accuracy of the delay models, (c) ability to tradeoff cost and delay, and (d) end-to-end system performance.

7.1 Datasets

Three datasets are used in our study.

Image Dataset: Four categories of images, including human faces, flowers, buildings, and book covers, are chosen as the dataset in our study. These four categories of images cover the precision spectrum of image search engines, from the extremely poor end to the extremely good end. Each dataset contains two parts: 1) training images that are used to build the image search engine, and 2) 500 query images captured by cellphone cameras to evaluate the performance of the search engine, and to generate human validation tasks to AMT. Note that query images are completely different from training images.

Training Dataset: From query images for each category, we randomly choose 300 images, and package each query image with its candidate results as validation tasks to AMT. We use only the top-5 candidate images for each query image. Each validation task is a <query image, candidate image> pair. In other words, we have $300 \times 5 = 1500$ tasks for each category of images. The queries are posted as a Poisson process with average interval of five minutes. The task posting lasts for several days to cover all time points in a day.

The validation results and the delay traces obtained from these queries comprise our training dataset. We use this training dataset to obtain parameters for our delay and result prediction models, and fix these parameters for all results in this section.

Testing Dataset: To validate our algorithms, we choose 200 query images from each of the four categories, and package each query images with its candidate images as validation tasks to AMT. For each query image, 5 candidate images are generated by image search engine as well. The posting process is exactly the same as the training dataset. The testing dataset is obtained two weeks after the training dataset to avoid overlap in the validators and to thoroughly evaluate our models.

The testing dataset provides us with a large set of validation tasks and the delay for each of the five responses for the tasks. This allows us to compare different task posting schemes by just changing the time at which a task is posted. The delays for responses to a particular task is assumed to be the same irrespective of when the task is posted. This assumption is reasonable since validators are independent of each other, and we find that most responses are from different validators.

7.2 Improving Search Precision

Our first series of experiments evaluate the precision of automated image search, and how much human validation can improve this precision. *Precision* is defined as the ratio of the number of correct results to the total number of results returned to the user. All four image categories are evaluated in this experiment.

We first look at the precision of automated search shown in Figure 7. On the x-axis is the length of the ranked list obtained from the search engine. The result shows that the top-ranked response has about 80% precision for categories such as buildings and books but has very poor precision for

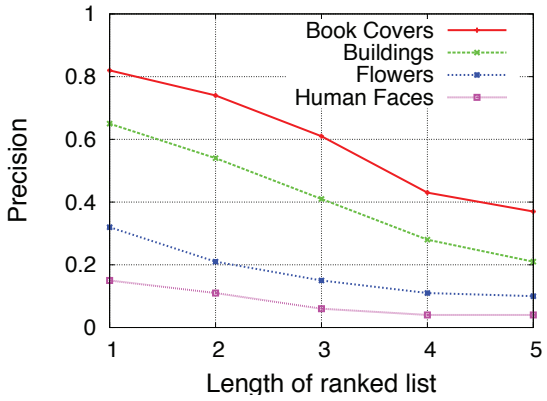


Figure 7: Precision of automated image search over four categories of images. These four categories cover the spectrum of the precision of automated search.

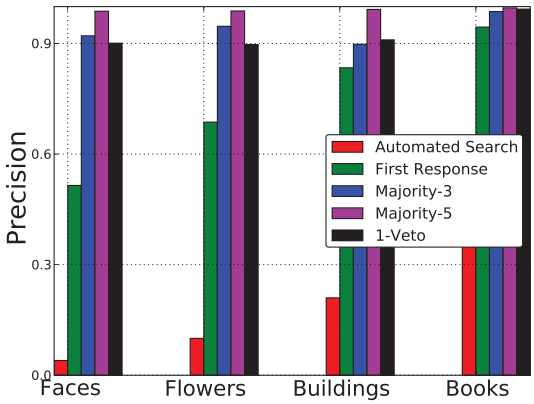


Figure 8: Precision of automated image search and human validation with four different validation criteria.

faces and flowers. The precision drops significantly as the length of the ranked list grows, indicating that even top-ranked images suffers from high error rate. Therefore, we cannot present the results directly to users.

We now evaluate how much human validation can improve image search precision. Figure 8 plots four different schemes for human validation: first-response, majority(3), majority(5), and one-veto (i.e. complete agreement among validators). In each of these cases, the human-validated search scheme returns only the candidate images on the ranked list that are deemed to be correct. Automated image search simply returns the top five images on the ranked list.

The results reveal two key observations: First, considerable improvement in precision is irrespective of which strategy is used. All four validation schemes are considerably better than automated search. For face images, even using a single human validation improves precision by 3 times whereas the use of a majority(5) scheme improves precision by 5 times. Even for book cover images, majority(5) still improves precision by 30%. In fact, the precision using human validators is also considerably better than the top-ranked response from the automatic search engine. Second, among the four schemes, human validation with majority(5) is easily the best performer and consistently provides accuracy greater than 95% for all image categories. Majority(3) is a close second, but its precision on face and building images is less than 95%. The one-veto scheme also cannot reach 95% precision for face, flower and building images. Using the first response gives the worst precision as it is affected most by human bias and error. Based on the above observation, we conclude that for mobile users who care about search precision, majority(5) is the best validation scheme.

7.3 Accuracy of Delay Models

The inter-arrival time models are central to the CrowdSearch algorithm. We obtain the parameters of the delay models using the training dataset, and validate the parameters against the testing dataset. Both datasets are described in §7.1. We validate the following five models: arrival of the first response, and inter-arrival times between two adjacent responses from 1^{st} to 2^{nd} response, to 4^{th} to 5^{th} response (§4.3.1). In this and the following experiments, we set the threshold to post next task be 0.6. In other words, if the probability that at least one of existing validation task is successful is less than 0.6, a new task is triggered.

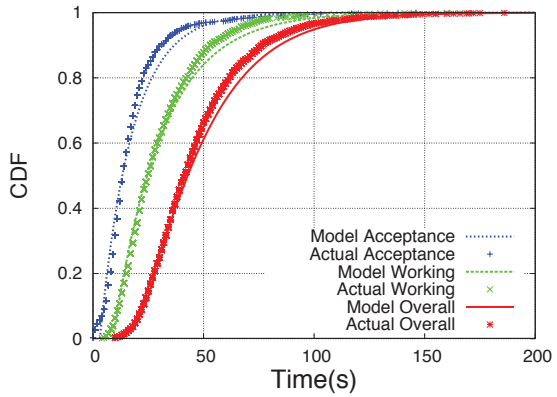
Figure 9(a) shows the cumulative distribution functions (CDF) for the first response. As described in §4.3.1, this model is derived by the convolution of the acceptance time and submission time distribution. We show that the model parameters for the acceptance, submission, as well as the total delay for the first response fit the testing data very well. Figure 9(b) shows the CDF of two of inter-arrival times between 1^{st} and 2^{nd} responses, and 3^{rd} and 4^{th} responses. (The other inter-arrival times are not shown to avoid clutter.) The scatter points are for testing dataset and the solid line curves are for our model. Again, the model fits the testing data very well.

While the results were shown visually in Figure 9, we quantify the error between the actual and predicted distributions using the K-L Divergence metric in Table 1. The K-L Divergence or relative entropy measures the distance between two distributions [3] in bits. Table 1 shows the distance between our model to the actual data is less than 5 bits for all the models, which is very small. These values are all negative, which indicates that the predicted delay of our model is little bit larger than the actual delay. This observation indicates that our models are conservative in the sense that they would rather post more tasks than miss the deadline requirement.

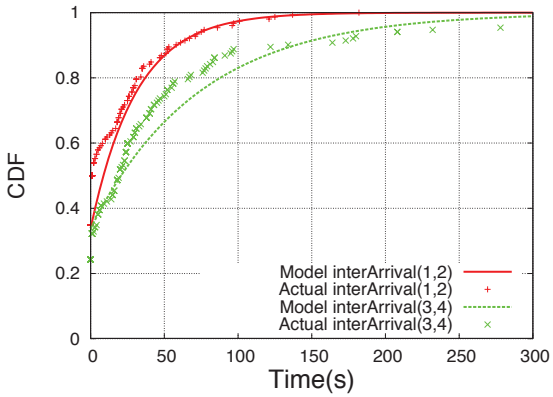
The results from Figure 9 show that the model parameters remain stable over time and can be used for prediction. In addition, it shows that our model provides an excellent approximation of the user behavior on a large-scale crowdsourcing system such as AMT.

7.4 CrowdSearch Performance

In this section, we evaluate the CrowdSearch algorithm on its ability to meet a user-specified deadline while maximizing accuracy and minimizing overall monetary cost. We compare the performance of CrowdSearch against two schemes: parallel posting and serial posting, described in §4.1. Parallel posting posts all five candidate results at the same time,



(a) Overall delay models vs. actual values



(b) Inter-arrival delay model vs. actual values

Figure 9: The inter-arrival delay and overall delay of predicting values and actual values. The rate for the models of overall, acceptance, working, inter-arrival(1,2) and inter-arrival(3,4) are: 0.008, 0.02, 0.015, 0.012 and 0.007.

Table 1: KL-divergence between delay models and distribution of testing dataset

Delay	KL-Divergence
Delay for the first response	-3.44
Inter-arrival Delay for response(1,2)	-4.60
Inter-arrival Delay for response(2,3)	-1.44
Inter-arrival Delay for response(3,4)	-3.12
Inter-arrival Delay for response(4,5)	-1.81

whereas Serial posting processes one candidate result at a time and returns the first successfully validated answer.

We evaluate three aspects: precision, recall, and cost. 1) *Precision* is defined as the ratio of the number of correct results to the total number of results returned to the user. 2) *Recall* is defined as the ratio of number of correctly retrieved results and the number of results that actually correct. 3) *Cost* is measured in dollars.

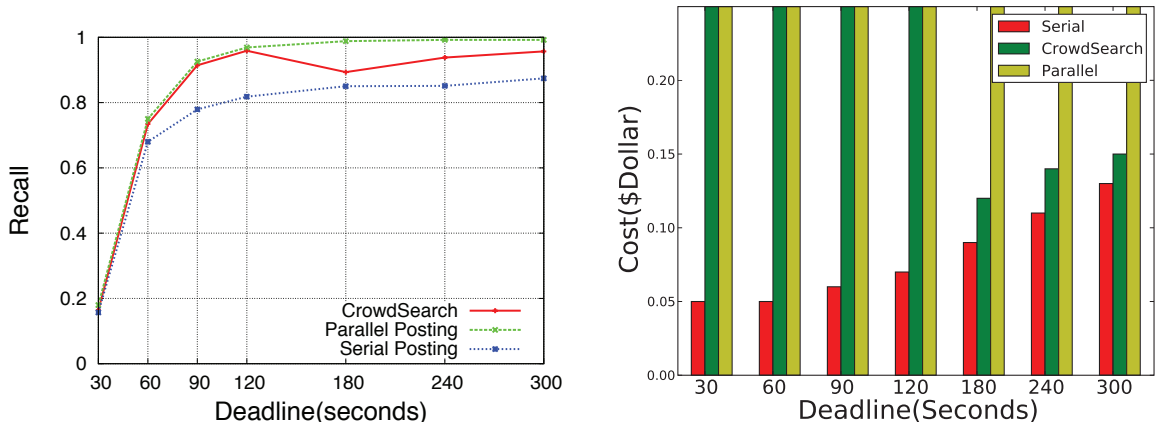
Varying user-specified deadline: To understand how the CrowdSearch algorithm works for different user-specified deadlines, we vary the deadline setting for the algorithm, and evaluate the precision, recall and cost performance of CrowdSearch. We study the testing trace for the buildings dataset. When the deadline is too low to receive a majority(5) response, all three algorithms (serial posting, parallel posting, and CrowdSearch) returns a response based on the majority of the received results. Thus, these schemes can provide an answer even if only one or three responses are received for a task.

Our experiments show that the precision for all three schemes are higher than 95% irrespective to the deadline setting. This is because human validation is intrinsically very good and has high accuracy. While precision is high, the impact of a short deadline can be observed in the recall parameter. Short deadlines may be insufficient to obtain a positive response, leading to the three posting schemes missing valid candidate images from the search engine.

Figure 10(a) shows the recall as a function of the user-specified delay. We first consider recall for the serial and parallel posting schemes. At extremely low deadline (30 sec), neither scheme obtains many responses from human validators, hence the recall is very poor and many valid images are missed. The parallel scheme is quicker to recover as the deadline increases, and recall quickly increases to acceptable regions of more than 90% at 90 seconds. The serial scheme does not have enough time to post multiple candidates, however, and is slower to improve. The performance of CrowdSearch is interesting. For stringent deadlines of 120 seconds or lower, CrowdSearch posts tasks aggressively since its prediction correctly estimates that it cannot meet the deadline without posting more tasks. Thus, the recall follows parallel posting. Beyond 120 seconds, CrowdSearch tends to wait longer and post fewer tasks since it has more slack. This can lead to some missed images which leads to the dip at 180 seconds. Again the recall increases after 180 seconds since the CrowdSearch has better prediction with more partial results.

Figure 10(b) shows the average price per validation task as a function of the user-specified delay. When the deadline is small, CrowdSearch behaves similar to parallel search, thus the cost of these two schemes are very close. When deadline is larger than 120 seconds, the cost of CrowdSearch is significantly smaller and only 6-7% more than serial search. In this case, the deadline recommendation provided by the CrowdSearch system to the user would be 180 or 240 seconds to obtain a balance between delay, cost, and accuracy in terms of precision and recall.

Breakdown for fixed deadline: To better understand how CrowdSearch improves over alternate approaches, we drill down into the case where the delay is fixed at 5 minutes (300 seconds). Figure 11(a) and 11(b) show the CDF of delay for building images and face images respectively. In both cases, we see that CrowdSearch tracks the serial posting scheme when the overall delay is low, since validation results are coming in quickly to post one task after the other. As the delay increases, CrowdSearch tracks the parallel posting scheme since it observes that the deadline is approaching and decides to post multiple tasks in-order to maximize chances of obtaining at least one valid response. Another interesting observation is about the case of faces. Crowd-



(a) Recall (b) Monetary Cost

Figure 10: Recall and Cost with different deadlines using building images. The recall of CrowdSearch is close to parallel, while the cost of CrowdSearch is close to serial.

Search posts more tasks after about 45 seconds, whereas it waits until about 75 seconds to become more aggressive in the case of the building dataset. This difference is because automated image search is more accurate for buildings, hence CrowdSearch receives more positive validation responses and decides to wait longer.

How much cost does CrowdSearch incur? Figure 11(c) compares the monetary cost of Serial posting and CrowdSearch for building images and face images respectively. In these two cases, parallel posting always incurs the highest cost of 25 cents per query, since it posts all 5 candidate images at the beginning. For building images, the monetary cost of CrowdSearch is 50% smaller than the parallel algorithm, and only 10-15% more than serial algorithm, which has the minimum possible cost. For face images, the difference between CrowdSearch and parallel algorithm is 10%, and the difference with serial posting is only 2%.

The above experiments shows that CrowdSearch has delay performance close to parallel posting, and monetary cost close to serial posting. Since parallel posting has the lowest possible delay, and serial posting has lowest possible cost, CrowdSearch achieves an excellent balance between between delay and cost.

7.5 Overall Performance

Finally, we evaluate the overall performance of CrowdSearch system comprising iPhones, a remote server, and crowdsourcing systems. Our evaluation covers two aspects of the CrowdSearch system: energy efficiency and end-to-end delay. In prior work [30], we showed that tuning the parameters of SIFT processing can reduce image search accuracy by up to 30%. We do not re-evaluate this result here.

Energy Efficiency : We now provide benchmarks for the energy-efficiency of the image search engine. We consider two design choices: 1) remote processing where phones are used only as a query front-end while all search functionality is at the remote server, and 2) partitioned execution, where the visterm extraction is performed on the phone and the search by visterms is done at the remote server. In each design, we consider the case where the network connection is via AT&T 3G and WiFi. The query is a VGA resolution image.

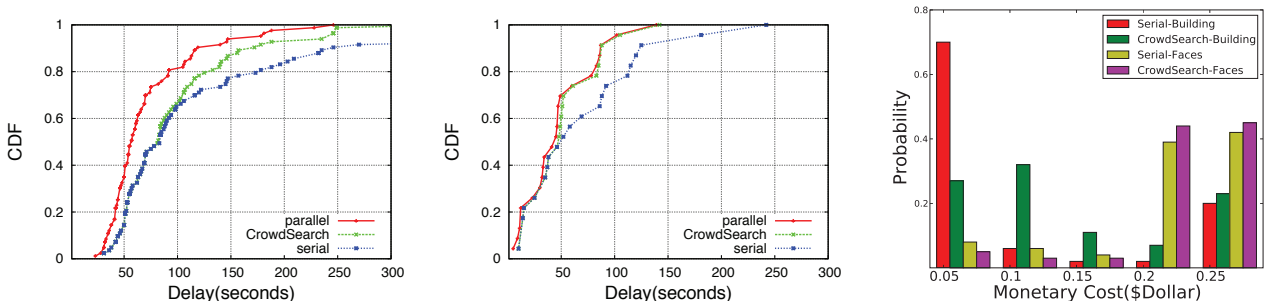
Figure 12(a) shows the average energy consumption for the four combinations of design and network. The numbers represent averages over 100 runs. The results show that the energy consumption of the partitioned scheme is the same irrespective of 3G or WiFi. This is because visterms are extremely compact and do not add much to the thumbnail image that is transmitted to the server for human validation. The energy-efficiency of the remote processing scheme greatly depends on the network used. With WiFi connectivity, remote processing is more efficient than local processing since transmitting a VGA image only takes a few seconds. In contrast, communicating the image via 3G is considerably more expensive, as 3G has greater power consumption and lower bandwidth than WiFi. In this case, partitioned processing can provide considerable energy gains.

Our results confirm our design choice of using remote processing when WiFi is available and local processing when only 3G is available.

End-to-end Delay: Dynamic partitioning of search functions also leads to different end-to-end delays. In this experiment, we evaluate the end-to-end delay for local and remote processing under 3G and WiFi. The crowdsourcing aspect of our system uses a 90 second deadline (although it returns results earlier if available). The results are averaged over 200 images. Figure 9 shows that local processing can be a significant fraction of the overall delay. While local processing consumes roughly a minute in our implementation, we believe that there are several optimizations that can be performed to reduce this delay. Human validation delay is about 4 times the delay incurred for transmitting the VGA image over a 3G network. We believe that as crowdsourcing systems increase in the number of validators that they support, the overall delay can reduce even further.

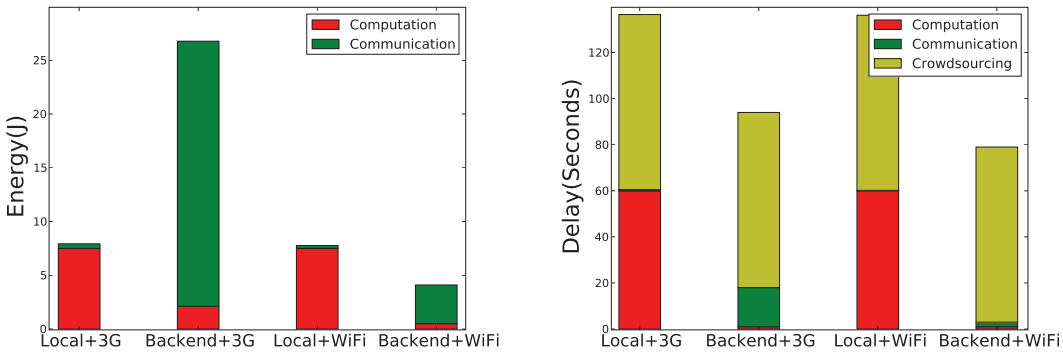
8. DISCUSSION

Although our results show that CrowdSearch can provide significant benefits for image search, there are several possi-



(a) Delay of Building Images (b) Delay of Face Images (c) Monetary cost compare

Figure 11: Comparison of delay and monetary cost for parallel posting, serial posting, and CrowdSearch. The comparison covers two set of images: buildings and faces. They represent the good and poor performance of automated search.



(a) Energy consumption (b) End-to-end delay

Figure 12: Overall energy consumption and end-to-end delay for CrowdSearch system. Overall energy consumption is comprised of computation and communication. End-to-end delay is comprised of delay caused by computation, communication and crowdsourcing of AMT.

ble improvements and potential opportunities that we have not explored in this paper.

Improving CrowdSearch Performance: CrowdSearch currently incurs a delay of the order of a few minutes, and incurs a cost of tens of cents. A natural question is whether further reductions are possible in delay and cost. We believe that improvements are likely along both axes.

The delay incurred for CrowdSearch will continue to reduce as crowdsourcing systems increase in popularity and scale, as evidenced by the increasing number of crowdsourcing startups that have emerged in recent months [17, 24, 20]. While there is room for improvement, it is not clear that delay guarantees of under a few tens of seconds are possible due to the intrinsic overhead for users to search, accept, visually inspect, and submit a task. Thus, a more realistic model for such systems may be one where the users post their queries to CrowdSearch and go offline (shutting their device, switching to other apps, etc). CrowdSearch processes the search query and sends the results to the user via notification, such as iPhone push notification or SMS.

The price of human validation can also be reduced through further optimization of the task posting scheme used in CrowdSearch. A simple optimization is to be more adaptive about how many duplicates are requested for each validation task. Our current system is limited by the inability to dynamically change the number of duplicates once a task is posted to the AMT, hence we chose a fixed threshold of five duplicates per task. However, this is suboptimal since we see that the first response is often a strong indicator of the overall validation result. (For example, if the first response is a YES, then the validation result is a YES with probability over 85%.) This observation can be used to adaptively tune the number of duplicates posted to the AMT, thereby reducing cost significantly.

Online Training of CrowdSearch Models: While the models we use in CrowdSearch are derived from a training dataset, they can be continually updated to reflect changes over time. Every validation task provides additional information about delay of responses, and hence inter-arrival time between responses. This data can be used to update the delay models to reflect the current behavior of AMT users. While we observed no time-of-day effects in our datasets, online training of the models can also help identify such behavior if it were to emerge. If so, a simple solution would be to use delay models with different parameters for different segments of a day. While any validation task can be used to

update delay models, online updates of the prediction tree requires ground truth information. One approach to update the prediction tree is to have a "gold standard" dataset where ground truth is available, and periodically introduce images from this dataset as validation tasks to the AMT.

Improving Automated Search Performance: In addition to filtering erroneous results generated by an automated search engine, CrowdSearch can also be used to improve the performance of the image search engine in terms of generating correct responses. This can be done by using human validated responses as feedback to the search engine to improve the quality of automatic search. One possibility is to use positive and negative feedback from CrowdSearch to adjust the scoring mechanism of the automated search. For instance, sometimes a background image, such as a grassy lawn can wrongly match several queries. CrowdSearch can be helpful to detect such background images and assign them lower weight to reduce false matches.

There are several other approaches to improve automated search results that we have not explored in this paper, the most obvious one being to use modalities such as GPS location, orientation, or user-provided text tags. A significant body of work in image search has addressed efficient and accurate multi-modal search methods (e.g. [31]). Similar methods can be applied to our search system as well. Despite these enhancements, we believe that CrowdSearch is an essential component to systems that seek to achieve close to 100% search accuracy.

Beyond Image Search: While our focus in this paper is on enhancing image search using crowdsourcing, we believe that the techniques are more broadly applicable to other instances of multimedia processing. One example where such techniques can work well is improving accuracy of audio transcription or machine translation, where human validators are used to verify the accuracy of transcription or translation. There is ample evidence that such audio-based tasks may be appropriate for crowdsourcing. For example, the Amazon Mechanical Turk has a significant fraction of tasks that involve audio to text transcription, and ChaCha [19] provides audio search that utilize people to answer search questions provided in the form of audio clips rather than text. CrowdSearch can be used to optimize the tradeoff of accuracy, delay, and cost in such audio search services.

CrowdSearch payment models: A practical consideration in deploying CrowdSearch on mobile phones is determining who pays for the cost of human validation. We envisage two possible payment models. One model is where the search provider pays for human validation in-order to provide a more accurate search experience for mobile users. This may be viable as long as the cost of validation is lower than the increased revenue to the search provider through targeted advertisements or user subscription. An example of a service that uses such a model is Amazon Remembers [27], which provides an image-to-product matching service that uses AMT. The price of human validation is small compared to the value of the product that may be purchased from Amazon by the user. An alternate model is where the mobile users pay directly for human validation through a micropayment account such as PayPal. This may be appropriate when the search is for something critical to the user, for example, lost items.

9. RELATED WORK

We now provide an overview of salient related work.

Image Search: Research in image search straddles advances in image processing and information retrieval. Techniques that we use in our system, such as SIFT, vistermextraction using vocabulary trees, and inverted index lookup are based on state-of-art approaches in this area [9, 13, 5, 12]. In our prior work, we have also applied such techniques in the context of sensor networks [30]. However, a well-known limitation is that these techniques work best for mostly planar images such as buildings, and are poor for non-planar images such as faces. This limitation is a reason why the recently released Google Goggle [21] system is primarily advertised for building landmarks. While our use of real-time human validation does not improve the performance of an image search engine, it can help filter incorrect responses to return only the good ones.

Much recent work such as [10, 1] has looked at mobile sensing and people centric sensing. Of relevance to this paper is the iScope system [31], which is a multi-modal image search system for mobile devices. iScope performs image search using a mixture of features as well as temporal or spatial information where available. While using multiple features can help image search engine performance, it does not solve the problem of low accuracy for certain categories. The crowdsourcing part of CrowdSearch is complementary to iScope, Google Goggles, or any other image search engine whose focus is on improving the performance of automated search. Any of these techniques can be augmented with CrowdSearch to achieve close to 100% precision at low cost.

Participatory Sensing: Sensing using mobile phones has become popular in recent years (e.g. Urban sensing [4], Nokia's Sensor Planet [22], MetroSense [7]). The emphasis of such efforts is to utilize humans with mobile phones for providing sensor data that can be used for applications ranging from traffic monitoring to community cleaning. Our work is distinct from these approaches in that we focus on designing human-in-the-loop computation systems rather than just using phones for data collection.

Crowdsourcing: Our system is by no means the first to exploit crowdsourcing for handling complex computation tasks. Luis Von Ahn's pioneering work on reCaptcha [29] uses humans to solve difficult OCR tasks, thereby enabling digitization of old books and newspapers. His another work on the ESP game [28] uses humans for finding good labels for images, thereby facilitating image search. The two systems use different incentive models — reCaptcha protects websites against robots, whereas ESP rewards participants with points if the players provide matching labels. Other popular crowdsourcing models include the auction-based crowdsourcing (e.g. Taskcn [23]), and simultaneous crowdsourcing contests (e.g. TopCoder [25]). Our work is inspired by these approaches, but differs in that we focus on using crowdsourcing to provide real-time search services for mobile users. To our knowledge, this model has not been explored prior to our work.

Many applications have begun to utilize micro-payment crowdsourcing systems such as AMT. This includes the use of crowdsourcing for labelling images and other complex data items [8, 15, 14]. For example, Sorokin et al. [15] show that using AMT for image annotation is a quick way to annotate large image databases. However, this work is

done in an offline manner and not in the context of a real-time search system. We also find that image annotation is quite noisy in comparison to validation of candidates from a search engine.

A central contribution of our work is modeling delay from crowdsourcing of validation tasks. While there have been several studies to understand quality of results from AMT solvers [8] [14], we are not aware of any other work that attempts to model delay from the system. We believe that our models can have broader applicability for other applications that use crowdsourcing systems.

Amazon Remembers [27] is an application that takes phone-based queries and uses crowdsourcing for retrieving product information from the queries. While Amazon Remembers combines mobile phones with crowdsourcing, CrowdSearch is considerably more sophisticated in its use of crowdsourcing since it enables *real-time* responses by specifying deadlines and combines automated and human processing.

10. CONCLUSIONS

Multimedia search presents a unique challenge. Unlike text that provides sufficient context to facilitate search, search using images is difficult due to the unavailability of clear features. The explosive growth of camera-equipped phones makes it crucial to design precise image search techniques. However, despite significant research in the area, a general image search system is still far from reality.

In this paper, we propose a new paradigm for real-time human-in-the-loop image search systems. Humans are excellent at distinguishing images, thus human validation can greatly improve the precision of image search. However, human validation costs time and money, hence we need to dynamically optimize these parameters to design an real-time and cost-effective system. CrowdSearch uses adaptive techniques based on accurate models of delay and accuracy behavior of human solvers. Our system is designed on iPhones, a backend server, and crowdsourcing system, such as Amazon Mechanical Turk. We demonstrate an 95% and above search precision, while being adaptive to the deadline needs for queries. Compare to alternative approaches that with similar search delay, our scheme saves the monetary cost up to 50%. While our work focuses on image search, our techniques, especially the AMT behavior models and prediction algorithms, open up a spectrum of possibilities and can be applied to areas beyond images to any multimedia search from mobile phones.

Acknowledgements

We are very thankful to Prof. Mark Corner for discussions on AMT, and to Prof. R. Manmatha for assistance with the automated image search system. We thank Prof. Romit Roy Choudhury for shepherding and the anonymous reviewers for their comments. This research was supported by NSF grants CNS-0546177, CNS-0546177, and CNS-0910900.

11. REFERENCES

[1] M. Azizyan, I. Constandache, and R. Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. *In Proceedings of MobiCom 09*, Sep 2009.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[3] K. P. Burnham and A. D. R. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, Second Edition*. Springer Science, New York, 2002.

[4] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In *WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet*, page 18, New York, NY, USA, 2006. ACM.

[5] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proceedings of CIVR '07*, pages 549–556, New York, NY, USA, 2007.

[6] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: Making smartphones last longer with code offload. In *In Proceedings of ACM MobiSys*, 2010.

[7] S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, G. seop Ahn, and A. T. Campbell. Metrosense project: People-centric sensing at scale. In *In WSW 2006 at Sensys*, 2006.

[8] A. Kittur, E. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. *CHI 2008*, Jan 2008. Crowdsourcing applied to user study.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints, 2003.

[10] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *MobiSys*, pages 165–178, 2009.

[11] M.-E. Nilsback. *An automatic visual Flora - segmentation and classification of flowers images*. PhD thesis, University of Oxford, 2009.

[12] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[13] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

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

[15] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. *Computer Vision and Pattern Recognition Workshops*, Jan 2008.

[16] `http://images.google.com/imagelabeler/`. Google Labeler.

[17] `https://www.livework.com/`. LiveWork: Outsource Business Tasks To Teams of On-Demand Workers.

[18] `http://www.abiresearch.com/research/1002762-US+Mobile+Email+and+Mobile+Web+Access+Trends`. US Mobile Email and Mobile Web Access Trends - 2008.

[19] `http://www.chacha.com/`. ChaCha: Real people answering your questions.

[20] `http://www.crowdspirit.com/`. CrowdSpirit: enables businesses to involve innovators from outside the company directly in the design of innovative products and services.

[21] `http://www.google.com/mobile/products/search.html#p=default`. Goggle: Google image search on mobile phones.

[22] `http://www.sensorplanet.org/`. Sensor Planet: a mobile device-centric large-scale Wireless Sensor Networks.

[23] `http://www.taskcn.com/`. Taskcn: A platform for outsourcing tasks.

[24] `http://www.theextraordinaries.org/crowdsourcing.html`. The Extraordinaries.

[25] `http://www.topcoder.com/`. www.topcoder.com.

[26] `http://www.vlfeat.org/~vedaldi/code/siftpp.html`. SIFT++: a lightweight C++ implementation of SIFT detector and descriptor.

[27] `http://www.wired.com/gadgetlab/2008/12/amazons-iphone/`. Amazon Mobile: Amazon Remember.

[28] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM Press.

[29] L. von Ahn, B. Maurer, C. Mcmillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, August 2008.

[30] T. Yan, D. Ganesan, and R. Manmatha. Distributed image search in camera sensor networks. *In Proceedings of SenSys 2008*, Jan 2008.

[31] C. Zhu, K. Li, Q. Lv, L. Shang, and R. Dick. iscope: personalized multi-modality image search for mobile devices. *In Proceedings of Mobisys '09*, Jun 2009.