Detecting Entities in Images using Metadata and the Crowd

Vasilis Verroios
Group No. 21
verroios@stanford.edu

1. INTRODUCTION

Search–by–image provides a new way for exploring the large collections of images hosted in the web. A user specifies an image as an input and the search engine finds a number of "similar" images using image processing, image metadata and/or human computation.

In this paper, we focus on a specific notion of "similarity" between images. We are interested in finding images that contain an entity tagged by the user in the input image. While entity inclusion can be considered as a concrete definition of similarity, the definition of an entity can become ambiguous depending on the case. For example, consider a user that tags a car in her input image. Is a car of the exact same brand and model the same entity, if it has a different color from the color of the input car? Is a car of the exact same model and color the same entity, if it has a different license plate? In order to partially resolve such ambiguities, we can ask the user to accompany his tagged input image with free text clarifying when two objects refer to the same entity. Thus, in our previous example, a accompanying free text could be "Any car of the exact same model; color does not matter".

In many applications image retrieval based on tagged entities can be of great value. Image artists or magazine photo editors looking for images that contain a specific object and terrorism/criminal investigation are examples of such applications. Compared to the traditional search–by–image retrieval, where results satisfy an abstract notion of similarity, retrieval by tagged entities provides a more well–defined interface to the end-user; either human or machine, e.g., web service.

Image retrieval by tagged entity can be based on mechanisms that exploit *a)* the content of images, *b)* the metadata describing properties of images, and *c)* human cognitive abilities. Unfortunately, all of the three approaches have significant drawbacks. Image processing mechanisms provide low–quality results compared to the human cognitive ability and their computational cost is prohibitive in the scale of the large image collections of today's web. Crowdsourcing can provide higher quality results but each *Human Intelligence Task (HIT)* implies a monetary cost. When the number of *HITs* to be performed are in the same order of magnitude with the number of images, there are very few, if any, applications that would be meaningful based on the total cost. Moreover, as the number of *HITs* increases, the time required for all of them to complete becomes prohibitive. On the other hand, algorithms that use image metadata as input can be efficiently applied in the large scale, but are unable to detect an entity in an image, if there is no description of such entity in the metadata.

Lately, a number of machine–human hybrid approaches [8, 9, 5, 2] have been proposed for entity resolution, which combine algorithms performed by machines and operations performed by humans. Nevertheless, the techniques proposed in [8, 9, 5, 2] can not be applied in the problem discussed here. The approaches of [8, 5, 2] issue a number of *HITs* proportional to the number of ambiguous matchings between pairs of entities/records. In the case of search by tagged entity in large-scale collections of images, the number of ambiguous matchings is high enough for making these approaches infeasible. The most relevant work to ours if that of [9]. However, the metric used in [9] for deciding which questions to ask the crowd, can not be applied in our problem. The reason is that this metric is bound to the notion of *transitive closure*, i.e., if records b and c are assumed to be the same entity with a record a, then b and c should also be the same entity. In our case, we can not apply *transitive closure* to compute the probability of three images containing the same entity.

Machine–human hybrid solutions have also been proposed for search–by–image [10]. Nevertheless, the approach of [10] focuses on the general, abstract notion of image similarity. Images are indexed based on features obtained through image processing, candidate search results are retrieved using this index and then are further filtered using crowdsourcing. Hence, this solution is not appropriate for the search by tagged entity problem, where the fine–grained entities specified by users can not be captured by today's image processing algorithms.

The approach we will follow in this paper, combines the metadata information describing an image with the information revealed by a small number of carefully selected *HITs*. Metadata information indicates the relationship between two images and is also related with the probability that the two images contain the same entity. For instance, consider two images taken by the same user, referring to the same location and date and having two persons tagged

in both of them. There is a considerable likelihood that a third person appearing in the first image, appears in the second image as well. Since our approach relies on metadata similarity between the initial image and other images, it is necessary that we have metadata available for the initial image. In practice, this means that either the initial image is part of the collection being searched or there is a preprocessing step that generates metadata for the input image, e.g., feature extraction through image processing [10, 4]. Note that in the notion of metadata we also include information generated by image processing.

Given a budget to be spent on *HITs* and a desired number of results, we examine which are the best questions to ask the crowd such that the expected number of matches is maximized. The obvious approach would be to ask the crowd for the images that have the highest probability of containing the input entity, based on the metadata. However, our approach goes beyond this simple observation. We also take into account the evidence that the outcome of crowdsourcing gives. Such evidence is important for selecting the rest of the images that will be included in the set of results. Hence, we ask the crowd for images that *a)* have a high probability of containing the input entity, and *b)* in case they are found to contain the entity, they will provide significant evidence for obtaining the rest of the results. For example, if two images have the same probability of being a match, and the second one has a lot more "connections" with other images than the first one, we would ask the crowd for the second image. Our approach can be of great benefit in cases where the desired number of results is too high for the available budget for crowdsourcing to support. In these cases, asking the crowd for a number of images higher than the desired number of results, is not a feasible solution.

In summary, we:

- formalize the problem of search by tagged entity using metadata and crowdsourcing, in section 2.
- prove that the problem is NP-*hard*, in section 3.
- propose heuristic algorithms that try to handle the problem based on three different concepts, *a)* low-hanging-fruits, *b)* image-centrality, and *d)* hubs and authorities, in section 4.
- experiment with a combination of synthetic and real data and reach to conclusions about which algorithms perform better in different settings, in section 5.

1.1 Running Example

The running example discussed here is used throughout the paper and serves to illustrate the challenges of the problem and to motivate our approach and solutions. Consider the network of images depicted in Fig. 1. A is the initial image, where the user has specified the entity to be searched over the network of images. Each link between two images represents the overlapping of their metadata. In the running example, we focus on only three types of common metadata between two images:

1. the common individuals that are tagged in both images.
2. a common date for the two images.
3. a common location for the two images.

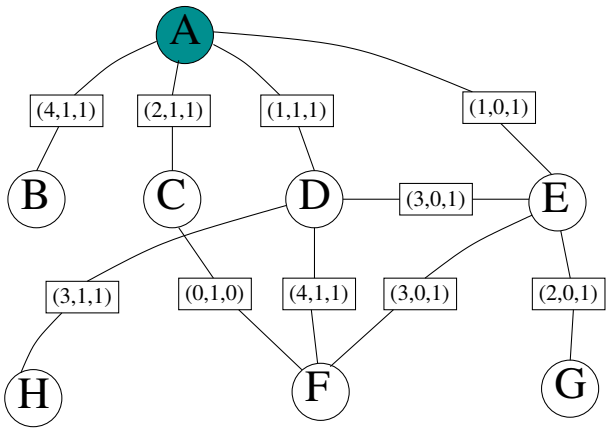

Figure 1: Image Network in Running Example

For instance, the first number in the triplet $(3, 0, 1)$ between images E and F, indicates that there are 3 individuals appearing in both images. Moreover, the second number indicates that the two images are not tagged as taken at the same location while the third number indicates that the two images are tagged with the same date.

Our goal is to find a subset of the images B, C, D, E, F, G, H, with the maximum number of matches, i.e, images containing the tagged entity the user specified in A. In the running example, we will assume that the number of results to be returned is 4 and that our budget for crowdsourcing is 2 *HITs*. A single *HIT* is formed by the comparison of the initial image with an image in question and the human is asked if the image in question contains the entity tagged in the initial image.

A possible strategy in this case, would be to create two *HITs* for images B and C, i.e., the two images that have the largest metadata overlaps with A; $(4, 1, 1)$ and $(2, 1, 1)$. If the outcome of the *HITs* points out that both images B and C contain the input entity of A, we will have to select two more images from D, E, F, G, H. However, images B and C do not have a significant metadata overlap with the rest of the images. In particular, there is only one link, the one with the triplet $0, 1, 0$ between image C and F. Therefore, the positive outcome for images B and C does not provide more evidence about the rest of the images that we need to include in the final 4 results.

A different strategy would be to create *HITs* for images D and E. While images D and E have a smaller metadata overlap with A than B and C, they have a significant overlap with images F, G, and H. Thus, in case the outcome of the two *HITs* points out that D and E contain the entity of A, we will have more evidence for the other two images that we will need to include in the final outcome.

A third strategy to consider, is to create *HITs* for B and D, taking into account that B has a large metadata overlap with A, and D has a large metadata overlap with the rest of the images in the network. Moreover, in the decision for which *HITs* to form, we must consider all the possible outcomes of the formed *HITs*. For example, how much will it help us if only B contains the input entity and how much of a help will it be if only D contains the input entity? Which is the probability of each of the four possible outcomes for the two *HITs*?

In the next section, we give a formal definition for the problem at hand and we describe our model about the *HITs* and the probability entailed by the metadata overlap between images.

2. PRELIMINARIES

A metadata overlap between an image and a set of images that contain an entity, implies a probability about the first image also containing this entity. Throughout the paper we use a probability function f over all such possible metadata overlaps, to denote this probability. Function f depends on the actual network of images that is searched over, and in this paper we will not examine how it can be derived. However, we will assume that we have access to this function during the definition of the problem and the description of our solutions, and in our experiments we will examine the effect of f by *a)* using different probability distributions, and *b)* using functions that include an error factor compared to the "true" function f.

We define an *HIT* as a comparison between the initial image containing the tagged entity, and a second image that the crowd must decide if it contains the entity or not. In large image networks the number of images that could potentially contain the input entity will be orders of magnitude higher than the number of *HITs* an application's budget for crowdsourcing can support. Therefore, in general, it won't be feasible to examine all the potential matches by grouping more than two images in a single *HIT*. For simplicity, we choose to include only two images in each *HIT* in our work. Moreover, we assume that humans do not make mistakes. Hence, if the outcome of an *HIT* is positive then the corresponding image will indeed contain the input entity. Otherwise, the image will not contain the input entity with probability 1.0. In practice, there are two main approaches for handling human errors in *HITs*. The first one is to use multiple *HITs* for a single comparison of an image with the initial image, and then to apply a mechanism for deciding if the first image contains the tagged entity or not, e.g., [7]. The second one is to rely on a crowdsourcing platform that supports worker quality control, e.g., [1].

Based on the probability function f over metadata overlaps and our definition for *HITs*, we examine two settings for the problem of finding the maximum number of matches using image metadata and crowdsourcing. In the first setting, we are interested in spending our complete crowdsourcing budget at once, i.e., we initiate all the *HITs* we can afford, in one run. In the second setting, which is a generalization of the first one, we want to spend only a part of our budget, taking into account the outcome of previous runs of *HITs* that we had put in place.

2.1 Metadata Overlap

The metadata overlap between an image i and an image j is denoted as MO_{ij}. In our running example, in Fig. 1, $MO_{DF} = (4, 1, 1)$. As stated earlier, the probability function f is defined over all the possible sets of metadata overlaps between an image and a set of images that contain the input entity. In our running example, if images C, D and E contain the input entity, $f(MO_{CF} \cup MO_{DF} \cup MO_{EF})$ gives the probability that image F also contains the input entity.

2.2 Crowdsourcing

The budget available for crowdsourcing is denoted as β. In other words, the total number of images we can send to the crowd for comparison with the initial image, is β. In practice, we may have to form more than a single *HIT* for each image that is being compared with the initial one to reach a conclusion with probability close to 1.0. However, since in this paper we do not examine the accuracy of the workers, our crowdsourcing budget is defined in terms of image pair comparisons.

2.3 Problem

For a subset S_β of β images that are sent to the crowd, there are 2^β possible outcomes. If for one such outcome, a subset $S \subset S_\beta$ of images was found to contain the input entity, we need to find the subset of $k-|S|$ images that have the maximum expected number of matches, where $k > \beta$ is the number of results that must be returned to the user. We denote this subset of $k - |S|$ images as $top_{k-|S|}$. Note that $top_{k-|S|}$ can only contain images that were not evaluated by the crowd, i.e., do not belong in S_β.

In the running example, if we send images D and E to the crowd and both of them are found to contain the input entity, the $top_{k-|S|}$ will be images F and B; where $k = 4$, $\beta = 2$. Due to the linearity of expectation, the expected number of matches in that case will be $2 + f(MO_{DF} \bigcup MO_{EF}) + f(MO_{AB})$. (We assume, in the context of the running example [1], that f is increasing, i.e., $f(MO_1) > f(MO_2)$ if $MO_1 \succ MO_2$. $MO_1 \succ MO_2$ when *a)* they both refer to one link and MO_1 is higher for all the metadata types of its link than MO_2, e.g., $MO_{AB} \succ MO_{AC}$, or *b)* MO_1 refers to the same number or more links than MO_2 and there is a "1–1" matching between the "top" [2] links of MO_1 and MO_2 such that each link of MO_1 has a higher overlap than its match in MO_2, e.g., $MO_{DF} \bigcup MO_{EF} \succ MO_{AC}$.)

In order to find the overall expected number of matches for a subset S_β, we must take into account all of the 2^β possible outcomes. In the running example, if we choose to send D and E to the crowd, there are four possible outcomes. Finding both of them as matches has a probability of $f(MO_{AD}) * f(MO_{AE})$. Finding that none of them is a match has a probability of $(1-f(MO_{AD}))*(1-f(MO_{AE}))$. Similarly, finding that only D is a match has a probability of $f(MO_{AD}) * (1 - f(MO_{AE}))$, while finding that only E is a match has a probability of $f(MO_{AE})(1 - f(MO_{AD}))$. Taking everything into consideration, the overall expected number of matches if we send D and E to the crowd is:

$$\begin{aligned}
&(f(MO_{AD}) * f(MO_{AE}) * [2 + f(MO_{DF} \cup MO_{EF}) + f(MO_{AB})]) + \\
&(f(MO_{AD}) * (1 - f(MO_{AE})) * [1 + f(MO_{AB}) + f(MO_{DF}) + f(MO_{DH})]) + \\
&(f(MO_{AE}) * (1 - f(MO_{AD})) * [1 + f(MO_{AB}) + f(MO_{AC}) + f(MO_{EF})]) + \\
&((1 - f(MO_{AD})) * (1 - f(MO_{AE})) * [f(MO_{AB}) + f(MO_{AC}) + 2 * f(\emptyset)])
\end{aligned}$$

In general, the expected number of matches for a subset of images S_β sent to the crowd, is given by:

$$\phi(S_\beta) = \sum_{S \subset S_\beta} \left(p(S)[|S| + \sum_{i \in top_{k-|S|}} f(\bigcup_{j \in S \cup A} MO_{ij})] \right)$$

where A is the initial image, and $p(S)$ is the probability that

[1] in general we do not impose such an assumption for f, although, intuitively, it seems applicable in all cases

[2] with the higher metadata overlap

S is the subset of images found as matches by the crowd:

$$p(S) = \prod_{i \in S} f(MO_{iA}) \prod_{i \in S_\beta \setminus S} (1 - f(MO_{iA}))$$

PROBLEM 1 (CORE). *Given a budget β, an initial image A, and the number k of results that must be returned to the user, find the subset S_β that maximizes $\phi(S_\beta)$.*

In the general version of the problem, we have also available the outcome of previous crowdsourcing runs. Through these runs we have obtained a set S_p of images that also contain the input tag of the initial image A. The question now is which is the best way to spend an additional budget β' taking into account the existence of S_p. In the running example, assume that in the first crowdsourcing run we have sent image D to the crowd and it was found that D does contain the input entity. Therefore, $S_p = \{D\}$. Again, the number of results we must return to the user is $k = 4$ and the budget we want to spend in the next crowdsourcing run is $\beta' = 1$. In this case, if we choose to send to the crowd image E, the expected number of matches will be:

$$(f(MO_{DE} \cup MO_{AE}) * [1 + 1 + f(MO_{DF} \cup MO_{EF}) + f(MO_{AB})]) + ((1 - f(MO_{DE} \cup MO_{AE})) * [1 + f(MO_{AB}) + f(MO_{DF}) + f(MO_{DH})])$$

In general, the expected number of matches for a subset of images $S_{\beta'}$, when we are aware of a set S_p of matches, is:

$$\phi'(S_{\beta'}) = \sum_{S \subset S_{\beta'}} \left(p'(S)[|S_p| + |S| + \sum_{i \in top_{k-|S \cup S_p|}} f(\bigcup_{j \in S \cup S_p \cup A} MO_{ij})] \right)$$

with

$$p'(S) = \prod_{i \in S} f(\bigcup_{j \in S_p \cup A} MO_{ij}) \prod_{i \in S_{\beta'} \setminus S} (1 - f(\bigcup_{j \in S_p \cup A} MO_{ij}))$$

PROBLEM 2 (GENERAL). *Given a budget β', an initial image A, the number k of results that must be returned to the user, and a set S_p of images already found to contain the input entity, find the subset $S_{\beta'}$ that maximizes $\phi'(S_{\beta'})$.*

The general version of the problem can be applied in the following two cases:

- We have more than one initial images that contain the input entity and we want to spend the crowdsourcing budget in one run to get a set with the maximum expected number of matches.
- We iteratively send images to the crowd, until our total budget is over or until there is no more time left and we need to send a response back to the user. In this case, we apply GENERAL in each next crowdsourcing run for the budget β' we choose to spend each time.

3. COMPUTATIONAL COMPLEXITY

In this section we show that the CORE and GENERAL problems are NP–*hard*.

THEOREM 1 (NP-hardness of CORE). *Finding the subset S_β that maximizes $\phi(S_\beta)$ is NP–hard.*

PROOF. We only give a sketch of the proof here; the complete proof can be found in APPENDIX 1. Our proof is based on a reduction from *MAX–CUT*. Given an instance of the *MAX–CUT* problem, we first construct a virtual network of images that corresponds to the initial graph, $G = (V, E)$, of the *MAX–CUT* instance. Then, we solve a CORE instance for each $\beta \in [1, \frac{|V|}{2}]$. In each of these CORE instances we set $k = |V|$. Because of the way we construct the virtual network of images, the solution to each CORE instance will correspond to the subgraph G_β of the initial graph G that has β nodes and the maximum cut to the rest of G, i.e., the maximum number of edges (i, j) s.t. $i \in G_\beta$ and $j \in G \setminus G_\beta$. Therefore, each CORE instance gives us the max cut on the initial graph G, when G is partitioned into a component of β nodes and a component of $|V| - \beta$ nodes. The best solution out of the $\frac{|V|}{2}$ solutions, will be the optimal solution to our initial *MAX–CUT* instance.

COROLLARY 1 (NP-hardness of GENERAL). *Finding the subset $S_{\beta'}$ that maximizes $\phi'(S_{\beta'})$ is NP–hard.*

PROOF. Since CORE is NP–*hard* and since GENERAL is a generalization of CORE, GENERAL will also be NP–*hard*.

4. HEURISTIC ALGORITHMS

In this section, we present three heuristic algorithms which select the images to send to the crowd for evaluation.

The first heuristic follows the naive approach and selects the "low hanging fruits" for crowd evaluation, i.e., the images which have the highest probability to contain the input entity. This first heuristic serves as the baseline in our experiments.

The second heuristic goes a step further. It tries to select images that not only have a high probability to contain the input entity but also provide significant evidence for the rest of the images that must be included in the results; without being evaluated by the crowd. In order to achieve that, the second heuristic uses a notion of "centrality" that expresses how "well" an image connects other images to the input set of images that contain the tagged entity. We quantify this notion for an image I, by combining the probability of I to contain the input entity with the metadata overlap of I with other images that do not belong in the input set of images.

The third heuristic is an extension of the second. Besides the "centrality" scores for each image, the algorithm uses a second type of scoring, expressing how "good" an image is, as a candidate for being included in the final results without being evaluated by the crowd. The idea is based on the *hubs and authorities* algorithm [3], where the *hubs* are the candidates for inclusion without being evaluated by the crowd and the *authorities* are the candidates for being sent to the crowd for evaluation. The *authority* score of an image indicates that it has a significant metadata overlap with "good-quality" hubs, i.e., images that are likely to be included to the results, after the crowd evaluation takes place. The *hub* score indicates that an image has a significant metadata overlap with images that are likely to be send to the crowd for evaluation.

In the next sections, we give a detailed description of the three heuristics.

4.1 Low Hanging Fruits

This first heuristic sends to the crowd for evaluation, the images that have the highest probability to contain the input

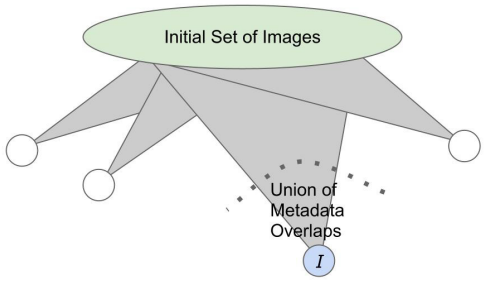

Figure 2: Graphical representation of Low Hanging Fruits

entity. Therefore, it does not consider the evidence that these images will provide for selecting the rest of the results, in case the crowd decides that they include the input entity. In our running example, in Fig. 1, and for a budget of 2 questions, the "low hanging fruits" would be images B and C.

The general case is depicted in Fig. 2. For each image I that has common metadata with at least one image of the input set S_p, the algorithm evaluates its probability to contain the input entity, i.e., $f(\bigcup_{j \in S_p} MO_{Ij})$. Images are sorted based on this probability, and the algorithm selects the β with the highest probability, for a crowdsourcing budget of β questions.

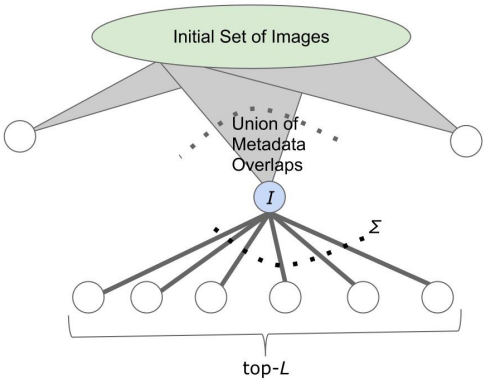

Figure 3: Graphical representation of Greedy Centrality

4.2 Greedy Centrality

The rationale of the algorithm is given by Fig. 3. For each image I that has common metadata with at least one image of the input set S_p, the algorithm computes a "centrality" score. This score is the product of two terms. The first term is the probability of I to contain the input entity. For the second term, the algorithm finds the images that have a metadata overlap with I and sorts them according to their probability to contain the input entity based on their overlap with I, i.e., for an image M, $f(MO_{IM})$. Then, the algorithm keeps only the top-L of the images that have an overlap with I and adds their expectations [3], i.e., $\sum_{j \in topL} f(MO_{Ij})$, to produce the second term for the centrality score of I. For example, in Fig. 1, for $L = 2$, the centrality score of D is $f(MO_{AD}) * (f(MO_{DH}) + f(MO_{DF}))$.

An important parameter involved in the definition of the centrality score, is the parameter L. The value of L is based on the expected number, E_m, of matches, out of the β images that will be sent to the crowd for evaluation. Then, if k is the overall number of images that must be returned as results to the end-user, $L = \frac{k-E_m}{E_m}$. Therefore, L defines that if E_m is the number of matches based on the crowd evaluation, each match will contribute to the rest of the $k - E_m$ results, with $\frac{k-E_m}{E_m}$ images. A rough estimation for E_m is the average probability to contain the input entity, computed over all the images that have common metadata with the input images' set.

Intuitively, the centrality score gives a rough estimation for the expected number of matches an image can contribute, if it is selected for crowd evaluation. The output of the algorithm is the top-β images based on their centrality scores.

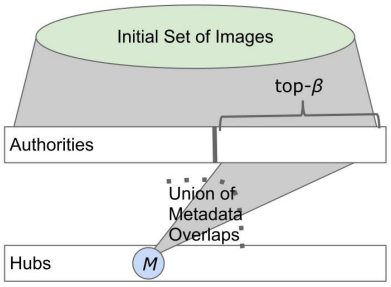

Figure 4: Graphical representation of a hub's score

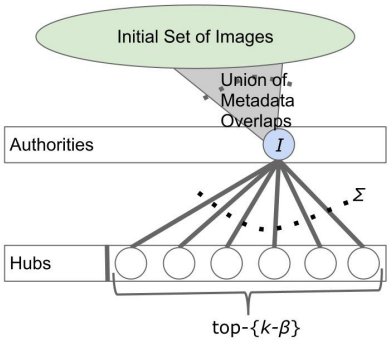

Figure 5: Graphical representation of an authority's score

4.3 topHITS

[3]because of linearity of expectation, this sum gives the expected number of matches I contributes

Greedy Centrality considers different top-L sets for each image, in order to compute the centrality scores. On the contrary, the *topHITS* algorithm attempts to detect a common set to be used in the centrality scores of all images. In order to quantify which are the nodes that should be part of this common set, the algorithm introduces a second type of scoring. In an analogy to the *hubs and authorities* algorithm, the *hub* score is the score indicating how good is a candidate node for being part of the common set and the *authority* score is the centrality score of each image based on the common set identified through *hub* scoring.

The algorithm starts by constructing two lists of images. The first list contains all the images that have a metadata overlap with at least one of the images of the input set. The second list contains the images that have a metadata overlap with at least one of the images of the first list. Note that some images may belong to both lists. As an initialization step, an *authority* score is computed for each image of the first list. The computation of the initial *authority* scores is identical with the procedure described in *Greedy Centrality*, for $L = k - \beta$; k is the overall number of results that must be returned to the end-user.

After the initialization step, the *topHITS* algorithm iteratively updates *hub* and *authority* scores using the following process. In each step, the *hub* score of each image is re-computed based on the top-β *authorities*. Specifically, the *hub* score of an image M in the second list, is $f(\bigcup_{j \in top\beta} MO_{jM})$, where *top*$\beta$ are the β *authorities* with the highest *authority* score. Fig. 4 gives a graphical representation of this computation. The second list is sorted based on the *hub* scores and then the *authority* scores of the first list is updated. For each image I in the first list, a new *authority* score is computed using as top-L, the $k - \beta$ *hubs* with the highest *hub* score. Fig. 5 gives a representation of this step.

The algorithm successively updates *hub* and *authority* scores until there is no change in the top entries of the two lists, or until a maximum number of iterations has been reached.

5. EXPERIMENTAL EVALUATION

Our evaluation is based on the flickr dataset used in [6]. However, while this dataset contains metadata information, we do not try to generate the function f that gives the probability of two images containing the same entity based on their metadata overlap. The reason is twofold. First, the information about the exact entities appearing in the images of this dataset is not available. Therefore, we would only be able to "learn" a rough estimate for the "true" probability function f. Second, our intent is to examine the performance of the different algorithms under different conditions with respect to the quality of the f used. Specifically, we focus on which algorithm performs better when there is a constant error between the f used and the "true" f.

Taking into account the above considerations, we combine the real-world structure and metadata of the flickr image network, with synthetic data about the underlying entities and the probability function f. In our experimental procedure, we use a "known" probability function f_k and a "true" probability function f_t. The procedure is the following:

1. Using f_k, run the algorithm-to-be-tested on the image network, and produce the set of images to send to the crowd.
2. Using f_t, synthetically generate matches and misses for the set examined by the crowd.
3. Using f_k and the matches found by the crowd, select the rest of the images to be included in the results.
4. Using f_t, synthetically generate matches and misses for the rest of the images that were included in the results.

For f_t we use the following family of functions:

$$f_t(\bigcup_i MO_i) = \sqrt[M]{\prod_{i \in topM} g(MO_i)}$$

For example, consider an image I that has a metadata overlap MO_i with each image i of a set of images containing the input entity. We apply a function $g()$ on each separate overlap MO_i, we select the top-M values [4], we multiply them together, and we get the M-th root of the the product as the outcome. Function $g()$ gives the probability of I containing the input entity based on the overlap MO_i with a single image i. The $g()$ we used in the experiments first produces a score for each of the 7 features an edge-overlap contains, based on the average value and the variance of all edges for each feature. Then, it takes the 7-th root of the scores' product as the outcome.

Each time f_k is used, we compute the real probability p_t, using f_t, we flip an unbiased coin, and we add or subtract a fixed error term ϵ from p_t, based on the coin outcome. In the experiments that appear here, ϵ is in the range of $[0.0, 0.3]$.

In the experiments of Fig. 6, we use a fixed crowdsourcing budget β of 20 questions, and we vary the ratio of K/β from 1.5 to 10; K is the total number of images that must be included in the results. For each combination of parameters, we run 40 experiments and we plot the average for the 40 runs. The top left plot of Fig. 6 is for $\epsilon = 0.0$, the top right is for $\epsilon = 0.1$, the bottom left is for $\epsilon = 0.2$, and the bottom right is for $\epsilon = 0.3$. *topHITS* provides the highest number of matches in all cases, while *Greedy Centrality* gives the second best outcome. The gap between the three heuristics slightly decreases as ϵ increases.

In the plots of Fig. 7, we use a fixed ratio of $K/\beta = 3$, and we vary β from 3 to 100. Again, we start with an $\epsilon = 0.0$ in the top left plot and we end up with an $\epsilon = 0.3$ in the bottom right plot. The results are similar with those of Fig. 7. *topHITS* consistently gives the best outcome and *Greedy Centrality* the second best. The number of matches decreases for all of the three heuristics as ϵ increases. Moreover, the gap between the three heuristics decreases.

The main result pointed out from Figs. 6 and 7 is that *topHITS* is consistently better than *Greedy Centrality*, which, in turn, is consistently better than *Low Hanging Fruits*. The difference between the three heuristics is small, indicating that there is room for improvement. However, the two heuristics remain better than the baseline approach, even when the error between the true probability distribution and the one used by the algorithms is large. Therefore, applying an algorithm that takes into account the evidence that the results of crowd evaluation will provide, is meaningful even when a rough estimation of the true probability distribution is available.

[4] a minimum value is used if less then M overlaps are available

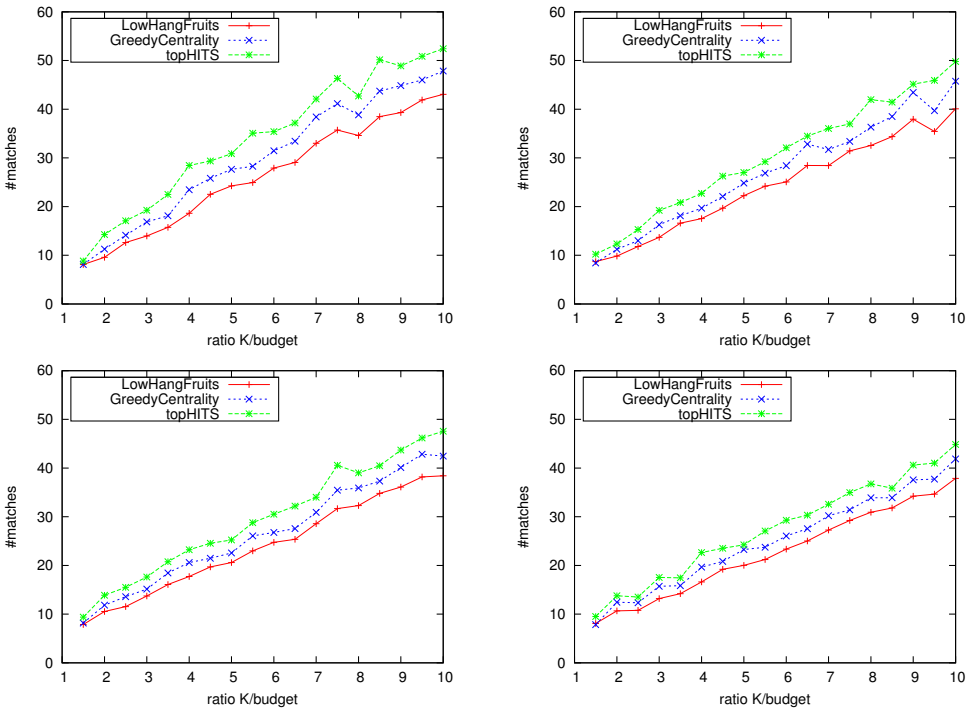

Figure 6: #Matches found as we increase *a)* the ratio of K to β, *b)*the error between f_k and f_t from 0.0(top left) to 0.3(bottom right)

The better outcome of *topHITS* comes with a cost. Fig. 8 gives the average computation time per run, when $\beta = 20$ and K varies from 30 to 100. Fig. 9 gives the average computation time per run, when $K = 50$ and β varies from 3 to 30. Clearly, *topHITS* implies a higher overhead. Nevertheless, such overhead is of less importance, since the dominant overhead is the crowdsourcing step. Besides the three heuristic algorithms, we implemented the exact algorithm that evaluates all possible image subsets os size β. The computational complexity of the exact method didn't allow us to run any meaningful experiments that would indicate how far our heuristics are from the optimal.

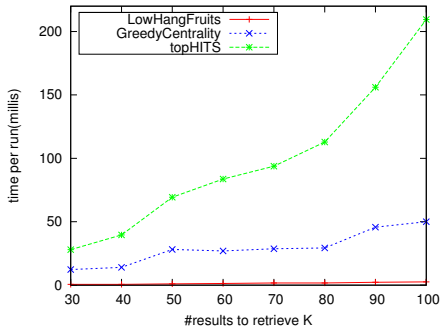

Figure 8: Time per run as we increase K

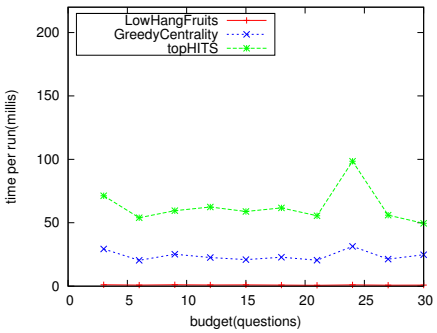

Figure 9: Time per run as we increase β

6. CONCLUSIONS

We proposed an approach for the problem of search-by-tagged-entity, that combines the interconnection of images through metadata information and human intelligence through crowdsourcing. In addition, we proposed two heuristics that perform slightly better than the naive solution. More importantly, our results show that the approach of combining crowdsourcing with structural information about the network, is a promising direction for this problem, even when only a rough estimation of the probability function for the existence of entities is available.

7. REFERENCES

[1] Crowdflower. http://crowdflower.com.
[2] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking.

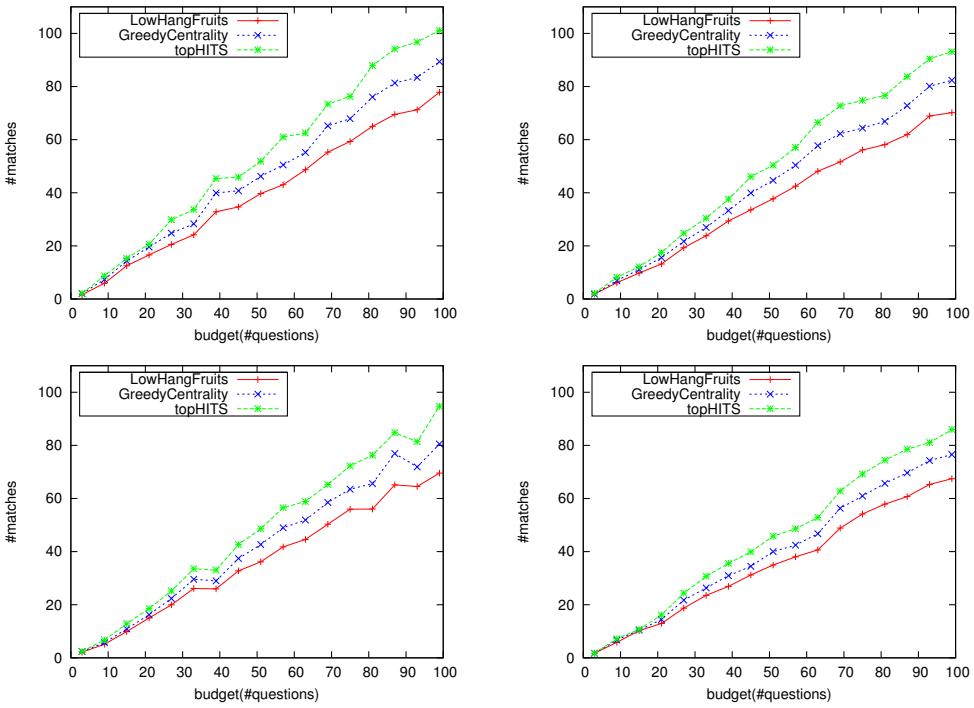

Figure 7: #Matches found as we increase *a)* budget β, *b)*the error between f_k and f_t from 0.0(top left) to 0.3(bottom right)

In *WWW*, 2012.

[3] J. M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 1999.

[4] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[5] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Human-powered sorts and joins. *PVLDB*, 2011.

[6] J. McAuley and J. Leskovec. Image labeling on a network: Using social-network metadata for image classification. In *ECCV*, 2012.

[7] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, 2012.

[8] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 2012.

[9] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. In *Technical Report*, Stanford, 2012.

[10] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: Exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, 2010.

APPENDIX 1

`THEOREM 1 (NP-hardness of CORE).` *Finding the subset S_β that maximizes $\phi(S_\beta)$ is NP–hard.*

`PROOF.` We will do a reduction from *MAX–CUT*. In the *MAX–CUT* problem, we are given a graph $G = (V, E)$, and we want to partition it to two components G_1 and $G \setminus G_1$ s.t. the number of edges (i, j) with $i \in G_1$ and $j \in G \setminus G_1$ are the maximum possible.

First, construct a graph $G' = (V', E')$, $V' = V \cup a$, $E' = E \cup \{(i, a), \forall i \in V\}$. The newly inserted node a corresponds to the virtual input image. Give weights to edges: $\forall (i, j) \in E'$, $w_{ij} = \frac{1}{2|V|}$ if $i \neq a$ AND $j \neq a$, or $w_{i,j} = \frac{1}{2}$ otherwise. The probability function f used is $f(E_S) = \sum_{(i,j)\in E_S} w_{ij}$, where E_S is any set of edges.

For each $\beta \in [1, \frac{|V|}{2}]$, solve a `CORE` problem with input β, G', $k = |V|$, and a. Lets focus on the value of one of these solutions. For a specific β, the optimal solution is a subset of nodes S_β^{opt} with $\phi(S_\beta^{opt})$

$$= \sum_{S \subseteq S_\beta^{opt}} \left(p(S) * [|S| + \sum_{i \in top_{k-|S|}} f(\bigcup_{j \in S \cup A} MO_{ij})] \right)$$

$$= \frac{1}{2^\beta} * \sum_{S \subseteq S_\beta^{opt}} \left([|S| + \sum_{i \in top_{k-|S|}} f(\bigcup_{j \in S \cup A} MO_{ij})] \right)$$

$$= \frac{1}{2^\beta} \left(\sum_{i=1}^{\beta} \binom{\beta}{i} i + \sum_{S \subseteq S_\beta^{opt}} [\sum_{i \in top_{k-|S|}} f(\bigcup_{j \in S \cup A} MO_{ij})] \right)$$

$$= \frac{1}{2^\beta} \left(\sum_{i=1}^{\beta} \binom{\beta}{i} i + \sum_{j \in S_\beta^{opt}} [\sum_{k=1}^{\beta} \frac{(\beta-1)!}{(\beta-k)!} * \sum_{\substack{(i,j): i \neq a \\ i \in V \setminus S_\beta^{opt}}} \frac{1}{2|V|}] + \sum_{i=1}^{\beta} \binom{\beta}{i} (|V| - \beta) \frac{1}{2} \right)$$

$$= \frac{1}{2^\beta} * \sum_{i=1}^{\beta} \binom{\beta}{i} i + (|V| - \beta) \frac{1}{2} + \frac{1}{2^\beta} * \sum_{k=1}^{\beta} \frac{(\beta-1)!}{(\beta-k)!} * \frac{1}{2|V|} * \sum_{j \in S_\beta^{opt}} |Cut_j|$$

where $Cut_j = \{(i,j) \in E : i \in V \setminus S_\beta^{opt}\}$

Transition from (1) to (2) is because $\forall S \subseteq S_\beta$, $p(S) = \frac{1}{2^\beta}$. Transition from (2) to (3) is because $\sum_{S \subseteq S_\beta} |S| = b * 1 + \binom{\beta}{2} * 2 + \ldots \binom{\beta}{\beta} * \beta = \sum_{i=1}^{\beta} \binom{\beta}{i} i$. Transition from (3) to (4) follows from two facts. First, the set $top_{k-|S|}$ is the same for any $S \subseteq S_\beta$. Since $|S_\beta| \geq |S|$ and since a node that belongs in S_β can not also belong in $top_{k-|S|}$ by definition, $top_{k-|S|}$ is always equal to $V \setminus S_\beta$. Second, each node in S_β contributes in $\sum_{i \in top_{k-|S|}} f(\bigcup_{j \in S \cup A} MO_{ij})$ one time for $|S| = 1$, $\beta - 1$ times for $|S| = 2$, $(\beta-1)(\beta-2)$ times for $|S| = 3$ and so on. In each contribution a node adds to the sum the number of edges that have that node as one endpoint and the other one on $V \setminus S_\beta$, multiplied by $\frac{1}{2|V|}$. Note that the above hold for any candidate set S_β and not only for the optimal solution S_β^{opt}.

From (5) we can conclude that S_β^{opt} is the subset of nodes that maximizes $\sum_{j \in S_\beta} |Cut_j|$. Thus, S_β^{opt} gives the maximum cut in G where one component must contain β nodes and the second one $|V| - \beta$ nodes. By solving the problem for all $\beta \in [1, \frac{|V|}{2}]$ we can find the overall max-cut of G.