Stella: Geotagging Images via Crowdsourcing

Christopher Jonathan
University of Minnesota, MN, USA
cjonathan@cs.umn.edu

Mohamed F. Mokbel[*]
Qatar Computing Research Institute, HBKU, Qatar
mokbel@hbku.edu.qa

ABSTRACT

Geotagged data (e.g. images or news items) have empowered various important applications, e.g., search engines and news agencies. However, the lack of available geotagged data significantly reduces the impact of such applications. Meanwhile, existing geotagging approaches rely on the existence of prior knowledge, e.g., accurate training dataset for machine learning techniques. This paper presents Stella; a crowdsourcing framework for image geotagging. The high accuracy of Stella is resulted by being able to recruit workers near the image location even without knowing its location. In addition, Stella also return its confidence about the reported location to help users in understanding the result quality. Experimental evaluation shows that Stella consistently geotags an image with an average of 95% accuracy and 90% of confidence.

CCS CONCEPTS

• **Information systems → Crowdsourcing**;

KEYWORDS

Spatial Crowdsourcing, Crowdsourcing, Geotagging Framework

ACM Reference Format:
Christopher Jonathan, Mohamed F. Mokbel. 2018. Stella: Geotagging Images via Crowdsourcing. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3274895.3274902

1 INTRODUCTION

Geotagging is the process of attaching a geographic location to an object (e.g., image or news item). Geotagging enables a myriad of important applications, e.g., web search engines use geotagged websites for enhanced search experience [1], location-based services analyze geotagged tweets to extract points-of-interest (POI) [19], and news agencies place geotagged news items on a map for enhanced user experience [21]. Meanwhile, several research efforts use geotagged data to discover local events [11], identify scenic routes [2], find out flooded areas [23], track food poisoning incidents [22], understanding city demographics [17], and many more. Due to the importance of such applications, academic and commercial systems were built to query, analyze, and visualize geotagged contents, e.g., see [3, 16, 18]. Unfortunately, the lack of available geotagged data significantly reduce the impact of such applications, e.g., only 0.7% of tweets [7] and 4.8% of Flickr images [8] are geotagged.

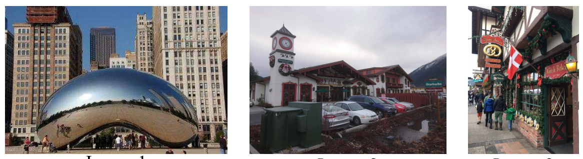


Figure 1: Example of Geotagging Images

Motivated by the importance of applications that need geotagged data, there were several commercial and academic efforts for geotagging, e.g., see [30, 31]. Unfortunately, most efforts are geared toward text-based data using natural language processing technique [30]. Meanwhile, there have been lack of research in image geotagging, where the state-of-the-art technique [31] suffers from low accuracy due to its over reliance with accurate training dataset. This can be seen from using major web search engines, e.g., Google Images [1], to geotag an image. Among the images in Figure 1, we could only geotag the first one. This is because web search engines rely on something like Google PlaNet [31], which geotag an image by training a convolutional neural network using millions of geotagged images, thus, unsuitable for new images. This is why it was successful only in geotagging our first image, a popular landmark in Chicago, USA, while other images are not that popular.

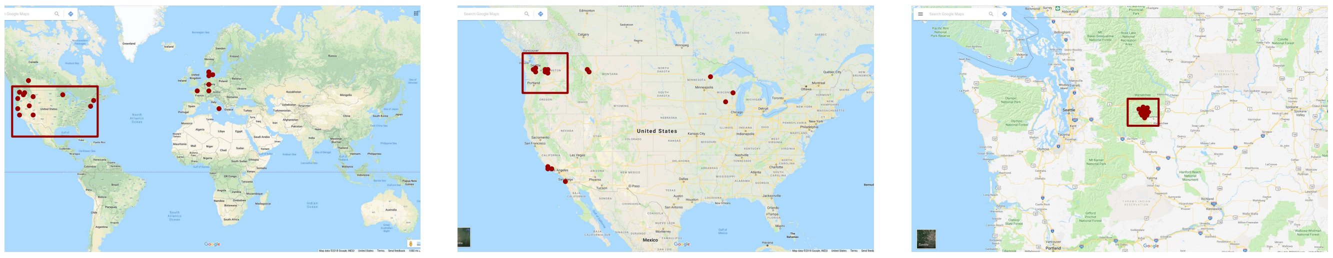
As the case with other machine-hard operations, researchers have turned to crowdsourcing to seek the wisdom of the crowd to solve those operations, e.g., POI labeling [9] and image search [32]. Following a similar approach, we tried to use crowdsourcing for image geotagging by running an experiment on Amazon Mechanical Turk [2] where we recruited 60 workers to identify the city of each image in Figure 1. It ends up that we were only able to identify the first image. This was expected as the workers were randomly recruited, thus, it would be difficult to geotag the other two images.

Should a crowdsourcing platform is smart enough, it would recruit workers who are familiar with these images. We believe that the closer a worker is to an image location, the more likely that he is able to geotag the image. We call such workers as *domestic* workers. To test our hypothesis, we ran another experiment where we recruited workers only from the state of each image. It ends up that a large majority of the workers were able to identify the city of each image. However, the challenge here is how to recruit those domestic workers, if we do not know the image location itself.

[*]Also affiliated with University of Minnesota, MN, USA.

This work is partially supported by the National Science Foundation, USA, under Grants IIS-1525953 and CNS-1512877.

[1]https://images.google.com/
[2]https://www.mturk.com/



(a) 1st Iteration (b) 2nd Iteration (c) 3rd Iteration

Figure 2: Overview of Stella Geotagging Process

In this paper, we present Stella; a crowdsourcing-based geotagging framework. Without loss of generality, we describe Stella in the case of image geotagging. Stella pushes the boundaries of crowdsourcing platforms to support geotagging by being able to recruit domestic workers without knowing the image location. Stella overcomes this dilemma by using an *iterative* approach to gradually understand the image location. Figure 2 shows an example of Stella in geotagging the second image of Figure 1 with a total budget of 60 workers. In the first iteration, Stella recruits only 20 worldwide workers out of its budget of 60. Figure 2(a) shows the answers we get from those 20 worldwide workers, where we ended up getting 20 different locations. While an existing crowdsourcing framework (e.g., Amazon Mechanical Turk) would conclude no answer here, Stella takes it further. Stella notices that the majority of the answers are within USA though they are from different locations and thus, concludes that this image must be somewhere in USA. In the second iteration, Stella recruits another 20 workers, all from USA rather than worldwide workers. Figure 2(b) shows the answers we get from those 20 USA workers, where again, we ended up getting 20 different locations within USA. However, Stella notices that the majority of the answers are within the Washington state. Hence, in the third iteration, Stella recruits its last 20 workers, all from the Washington state. Figure 2(c) shows the answers from those 20 workers, where there is a clear agreement on the whereabouts of the image. As a result, Stella can safely conclude the location of the image. The main idea is that Stella was finally able to recruit 20 domestic workers, i.e., those workers who live close to the image, even though the image location was unknown.

Stella does not only return the image location as an answer. Instead, it goes beyond to report on how confident it is in the answer. The main idea is to calculate the confidence based on the spatial diversity of the answers. The less spatially diverse answers we have, the more confident Stella is. Also, the more domestic workers we can recruit, the higher the confidence in the answer is. Extensive experiments of Stella using real deployment on Amazon Mechanical Turk shows that Stella is consistently able to geotag images with an average of 95% accuracy and 90% confidence.

The rest of this paper is organized as follows: Section 2 gives an overview of Stella. A basic Stella framework is presented in Section 3. Section 4 gives a set of optimizations to enhance the performance of Stella. Section 5 discusses how we evaluate Stella's results and confidence. Extensive experimental results of Stella are presented in Section 6. Related work to Stella is highlighted in Section 7. Finally, Section 8 concludes the paper.

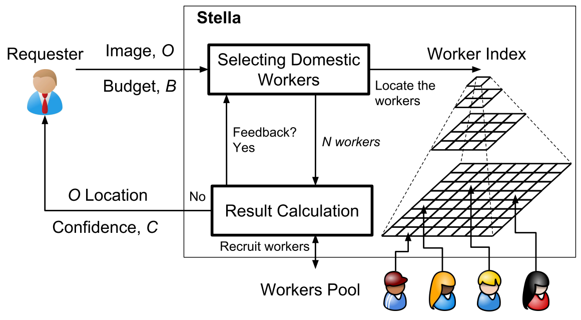


Figure 3: Stella System Architecture

2 SYSTEM OVERVIEW

Figure 3 gives the system architecture of Stella. A user submits an image O that needs to be geotagged and a budget B that the user is willing to pay. The answer is returned to the user as the location of O and a confidence value C that tells how much Stella is confident about the reported location of O. Meanwhile, workers who are registered in Stella indicate their willingness to participate in a given crowdsourcing task and are willing to share part of their location information. Studying the workers' location privacy and incentives is beyond the scope of this paper (see [26] and [20] for a study on worker's location privacy and incentives, respectively).

For efficient retrieval of workers within a certain area, Stella indexes the workers' locations in a multi-resolution non-overlapping spatial index structure. A worker's location is taken from the worker profile upon registering in Stella. Then, it is updated only when the worker explicitly updates it to get more matching tasks. So, there is no need to track any kind of worker's movement. The index can be a predefined spatial regions, such as the whole world in the first level, country in the second level, state in the third level, and so on, or a well-established space partitioning index structure, e.g., a pyramid index structure [24]. For ease of understanding, we use a pyramid structure as our multi-resolution index structure with height H where each worker's location information is stored in one of the cells on the lowest level of the pyramid.

Internally, the main challenge that Stella faces is how to find and recruit domestic workers without knowing the image location. The hypothesis is that domestic workers are more capable of geotagging an image than non-domestic workers. We have verified our hypothesis by running 150+ real crowdsourcing tasks with 600+ workers on Amazon Mechanical Turk (will be discussed further in Section 6.1). Stella addresses this challenge by introducing the idea of *adaptive*

Algorithm 1 Basic Stella

```
1: procedure GEOTAG($Index\ P$, $Image\ O$, $Budget\ B$)
2:     $S \leftarrow P.root$; $C \leftarrow 100\%$; $N \leftarrow B/P.H$
3:     while $S$ do
4:         $W \leftarrow$ SelectDomesticWorkers($N, S$)
5:         $(S, O.loc, C) \leftarrow$ ResultCalculation($W$)
6:     end while
7:     return $O.loc, C$
8: end procedure
```

crowdsourcing. Unlike conventional crowdsourcing platforms that assign a given task to all workers at once, Stella assigns the geotagging task to only a subset of the workers. Then, based on the result, Stella learns more knowledge about the image whereabouts and recruits another subset of the workers that are more domestic than the previous ones. This process is depicted in Figure 3 by the iterations over two main internal modules in Stella, namely, *Selecting Domestic Workers* and *Result Calculation*. The first module receives an amount of budget that it can spend on one iteration, N, as well as a hint about O location, provided as a feedback from running the second module from the previous iteration. The goal of this module is to find N domestic workers located within the provided hint, and send their information to the second module. Then, the second module sends the geotagging task to the assigned workers. Unless this is the last iteration, the objective is to find a smaller search space for the next iteration. If this is the last iteration, the module will provide the final answer, along with a confidence value that is computed incrementally across iterations.

3 BASIC STELLA FRAMEWORK

This section presents our *basic* Stella; the simplest form of the Stella framework (pseudocode shown in Algorithm 1). This is to focus on the main idea of Stella without digging into the optimization detail on every internal decision of Stella. Stella takes image O, budget B, and pyramid index structure P as its input, and outputs the image location $O.loc$ and a confidence value C. Then, Stella goes through H iterations, where H is the pyramid height, with $N = B/H$ workers assigned in each iteration. Each iteration is composed of two steps: (1) *Selecting domestic workers* (Section 3.1), where the objective is to select N workers within a search space, and (2) *Result calculation* (Section 3.2), where the objective is to predict the image location along with computing the confidence.

3.1 Step 1: Selecting Domestic Workers

This step takes a current search space S and the number of workers N as its input. The goal is to find N workers, inside and uniformly distributed over S, and output those workers to the second step. We do so by exploiting the structure of the pyramid index for the cells rooted at S and recruit $N/4$ workers from each child cell of S. For each cell, we recursively traverse its children until we do not have any workers left or until we reach the lowest level of P. For example, if $N = 4$, we select one worker from each of S children. If $N = 16$, we select one worker from each of the 16 cells that are two levels below S. In general, we will traverse $\lfloor \log_4 N \rfloor$ level(s) to uniformly assign workers. If N is not divisible by four, we assign the remainder of the workers randomly among the cells. When we decide to get workers from a certain cell, we randomly select them.

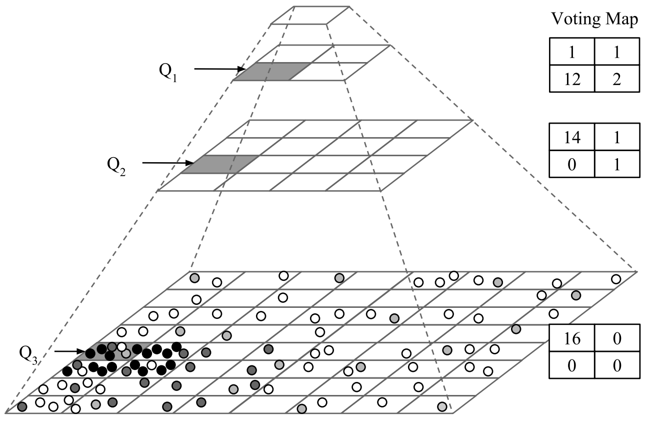


Figure 4: Basic Stella Example

3.2 Step 2: Result Calculation

This step takes the set of N workers from the previous step and sends them the image O. The answer is received from each worker in the form of (*latitude*, *longitude*). Unless this is the last iteration, we are not trying to infer the exact location of O. Instead, the goal is to identify which cell among the children of S that will be the new search space of the next iteration. So, instead of checking the exact (*latitude*, *longitude*) of every answer, we check which cell among the children of S that these coordinates are. Then, the answer is seen as if each worker is voting on which child of S contains O. By deploying a simple majority voting, we decide that the quarter that takes the most votes becomes the new S. If this iteration is the last one, we infer the exact location of O. There are many ways to do so, including, reporting the location as the minimum bounding rectangle (MBR) that includes the N results, finding the centroid of the N results, or using a density-based clustering technique such as DBSCAN [4]. Without loss of generality, Stella chooses to report back the MBR of the answers as the location of O.

Meanwhile, the confidence value of each iteration, i.e., the *local confidence*, is computed as the ratio of workers who vote for the new search space over N. With a pyramid of height H, there will be a total of H local confidence values. At the final iteration, we calculate the *overall confidence* as the geometric mean [25] of the H local confidence values. We decide to use geometric mean as it has been widely used to compare values that has different properties, e.g., in calculating the citation impact of research articles [25].

3.3 Detailed Example

Figure 4 gives an example of basic Stella, where the set of available workers are depicted by circles (regardless of their colors) in the lowest pyramid level. With a pyramid height of four levels ($H = 4$) and budget $B = 64$, there is a total of four iterations with 16 workers in each iteration ($N = 16$). At the start, the initial search space S is set as the root of the pyramid. In the first iteration, we select 16 workers uniformly distributed over S by picking one from each of the 16 cells at the third level (depicted as light gray circles). Then, these workers vote on which quarter O is located. An example of the voting result is depicted on the right of the second pyramid level. We select quarter Q_1 (depicted as gray cell) as the new S with local confidence of $12/16 = 75\%$.

Algorithm 2 Optimized Stella

```
1:  procedure GEOTAG($Index\ P$, $Image\ O$, $Budget\ B$)
2:      $S \leftarrow P.root$; $C \leftarrow 100\%$; $N \leftarrow B/P.H$
3:      $L \leftarrow$ Empty List;        /* OPTIMIZATIONS 1, 2 */
4:      while $S$ do
5:          if SkipIteration($L$, $S$) then /*OPTIMIZATION 2*/
6:              $(S, C, N) \leftarrow$ OfflineCalculation($S$, $L$)
7:              continue;
8:          end if
            /* Step 1: OPTIMIZATIONS 1, 4 */
9:          $W \leftarrow$ SelectDomesticWorkers($N$, $S$, $L$)
            /* Step 2: OPTIMIZATIONS 1, 2, 3, 4 */
10:         $(S, O.loc, C, L) \leftarrow$ ResultCalculation($W$)
11:     end while
12:     return $O.loc, C$
13: end procedure
```

In the second iteration, we select another 16 workers all within Q_1 by selecting one worker from each of the 16 cells in the lowest pyramid level that are the grandchildren of Q_1 (depicted as dark gray circles). The voting result is shown next to level 3 with a local confidence of $14/16 = 87.5\%$. In the third iteration, we select another 16 workers within Q_2. Since we only have one level left at the end, we end up selecting four workers from each of the four children of Q_2 (depicted by the black circles). Again, all the 16 workers agree that O is located in Q_3 with a local confidence of 100%. In the final iteration, we recruit our last batch of 16 workers (making a total of 64 workers) which all selected randomly from Q_3 (these workers are not depicted in the figure). Then, we report back the image location as the MBR that includes all exact locations returned from these last 16 workers with an overall confidence value of $(12/16 \times 14/16 \times 16/16 \times 16/16)^{\frac{1}{4}} = 90\%$.

4 OPTIMIZING STELLA

This section presents our *optimized* Stella; where we deploy four techniques to optimize our basic approach. All optimizations are generally geared towards ensuring that the recruited N workers are more domestic. Algorithm 2 gives the pseudo code of our optimized Stella, which follows the same skeleton of our basic Stella with the added optimizations annotated by comments.

4.1 Optimization 1: Domestic Workers

Objective. The objective of this optimization is to fully deploy the domestic worker concept behind Stella. Recruiting all workers uniformly distributed over the whole space will give low quality answer as there are only a small part of workers that are domestic to the image while other workers have a far distance to the image. That is why we only have subset of our workers chosen uniformly, then, we narrow down the search space, and recruit workers uniformly from the new smaller search space. In this optimization, we avoid recruiting workers uniformly from the new smaller search space in every iteration. We would like to always have our workers distributed in a skewed way biased towards the image location.

Main Idea. The main idea is that after retrieving the detailed workers' answers of exact (*latitude*, *longitude*) from an iteration, Stella passes this information to the next one. Then, the next iteration

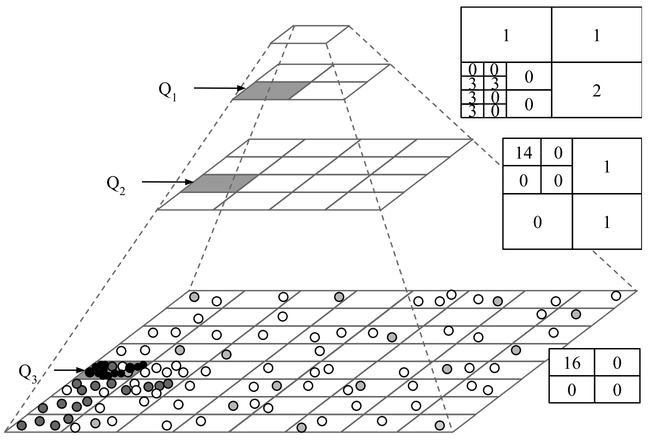


Figure 5: Optimized Stella Example

will use this information to select N workers that match the spatial distribution of the exact locations from the previous iterations by mapping them into the children of S rather than selecting them uniformly. With this, we are taking advantage of every bit of workers' answers that we have to predict the location of O to recruit workers where O is more likely to be.

Algorithm. We make the following three modifications which are annotated by OPTIMIZATION 1 in the algorithm: (1) We first initialize an empty list L to be used as a buffer to store the N workers' (*latitude*, *longitude*) answers. (2) We modify Step 1 to take an additional parameter L, and use it as a guide for the distribution of the N workers to be recruited within S. Since L is initially empty, the first iteration will select the N workers in a uniform way. (3) We modify Step 2 to return a fourth output, which is an updated list L that contains the N current iteration results. The list L and new search space S will be passed to the next iteration. The first step of the next iteration will traverse $\lfloor \log_4 N \rfloor$ level(s) and make use of L to decide on number of workers to get from each pyramid cell.

Example. Figure 5 shows the effect of this optimization on the example of Figure 4 with $H = 4$ and $N = 16$. In the first iteration, the first step of the algorithm is similar to basic Stella as we do not have any prior results. The second step of the algorithm comes up with a more detailed voting box that votes on the grandchildren of Q_1. In the second iteration, we recruit four workers from the four non-zero voted cells within Q_1 and none from other cells (depicted by dark gray circles). By contrasting Figures 5 and 4 for the dark gray circles, it is very clear that the recruited workers in the optimized Stella are much more localized than the ones in the basic Stella. Then, another detailed voting box is produced from this iteration, yet, with only votes on the children of Q_2, as there is only one pyramid level below it. In the third iteration, we recruit $\lfloor 14/14 \times N \rfloor$ = 16 workers from top left child (depicted by black circles). It can be visually seen that there are more black circles in Q_3, namely 16, in Figure 5 than that of Figure 4, namely four.

4.2 Optimization 2: Skipping Iterations

Objective. The objective of this optimization is to take advantage of cases where there are kind of an agreement on the whereabouts of the image O in one of the iterations and skip the iteration. Then,

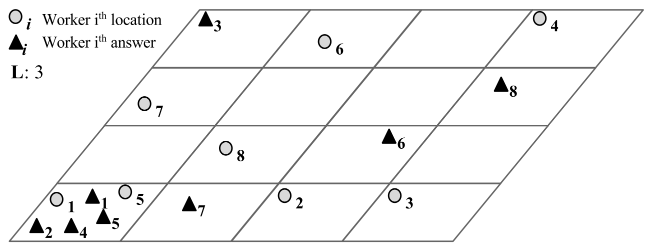


Figure 6: Optimization 3 Example

we reuse the budget from every skipped iteration to recruit more domestic workers in further iterations. For example, in geotagging an image of the Statue of Liberty in New York City, USA, it is likely that we can infer the city directly regardless of its exact location only from the first iteration. Thus, we can skip multiple iterations and use the skipped budget to recruit more domestic workers directly within the New York City to find the image exact location.

Main Idea. The main idea behind this optimization is to check on whether or not the previous iteration results agree on a certain child of S. To do so, we identify if there exists an outlier cell among S children that contains most workers' answers from the previous iteration. In Stella, we go with a simple high majority formula where, among the four children of S, we consider that a certain cell Q is an outlier cell if it has more than x% of the votes, with x is a system parameter with a default value of x = 80%. If there is such an outlier quarter Q, we reuse the results from the previous iteration as the results that are coming from workers in Q in the current iteration. Reusing these results saves us a budget of N workers that we can use in further iterations. However, we would still need to make *offline* computation to update the local confidence value for every skipped iteration. The number of workers of any skipped iterations will be distributed equally over the forthcoming iterations, which will make these workers more domestic than the case of having them in their original skipped iteration(s).

Algorithm. We make the following three modifications which are annotated by OPTIMIZATION 2 in the algorithm: (1) We initialize an empty list L to store N workers' answers, which we will use to decide on whether or not we can skip an iteration. (2) We start each iteration by performing a test that takes L and S as the inputs to check whether or not we can skip the current iteration. In the first iteration, the test will return *false* as L is still empty. In further iterations, the test will return *true* if and only if it finds an outlier cell. In that case, we skip the current iteration. Instead, we will only make offline calculation to update the search space S, confidence C, and number of workers N. (3) We will need to return the updated list L as the fourth output to be used in the next iteration.

Example. Consider the example in Figure 5 with $H = 4$ and $B = 64$. In the first iteration, the algorithm will proceed the same way as before. In the second iteration, we have the number of workers voting for each child of Q_1 where the two left children of Q_1 receive 6 votes each. Applying our threshold value of x = 80%, we find that no child in Q_1 has more than x% of the total votes, so we proceed as usual. In the third iteration, the number of workers voting of each child of Q_2 is (14, 0, 0, 0), with the top left child of Q_2 having a clear higher majority, i.e., $14/16$ = 87.5% of the total votes which is more than 80%. So, we skip this iteration and do not hire any workers here. Instead, we only do offline computation to update S to be Q_3, N to be 32, the local confidence value of 87.5%. As the next iteration is the final one, we recruit 32 workers from Q_3. This will clearly give a more accurate result as we will be recruiting workers that are more domestic than the original ones with the new overall confidence of: $(12/16 \times 14/16 \times 14/16 \times 32/32)^{\frac{1}{4}} = 87\%$.

4.3 Optimization 3: Weighted Confidence

Objective. The main objective of this optimization is to increase the confidence of Stella. Based on our hypothesis of domestic workers, there is a higher probability that a worker will return an accurate location when the worker selects his nearby locations rather than further locations. Thus, instead of valuing all workers' answers equally, we should assign a higher weight to an answer that is closely located with the worker's location, thus increasing the confidence of Stella. For example, when a worker who is living in Minneapolis geotags O to be in Minneapolis, it's more likely that the worker is more confident than geotagging O to be in Seattle.

Main Idea. The main idea behind this optimization is to weight each worker's answer based on the distance between the worker's location and his answer. The closer the distance is, the higher the weight will be. Without loss of generality, we use a simple cell distance to calculate the distance between two cells. For each worker w, the weight of w's answer is calculated as: d_{max} - d_{actual} + 1 where d_{actual} is the cell distance between the worker and his answer and d_{max} is the maximum cell distance at any pyramid levels, e.g., $d_{max} = 2$ in level 2 and $d_{max} = 6$ in level 3. Then, these weights are used to calculate the local confidence in each iteration as the ratio between the weighted sum of answers in the selected cell to the total weighted sum of all answers in this iteration.

Algorithm. The only modification for this optimization (annotated by OPTIMIZATION 3 in the algorithm) is that Step 2 of the algorithm will make use of the location of every worker it received from Step 1. We will retrieve the exact location of each worker that is stored in the index P and use it to calculate the weight of her answer. Similarly, the local confidence and overall confidence values of the iteration is calculated based on the total weighted answer on the new search space over the total weight of the iteration.

Example. Figure 6 gives an example that illustrates this optimization, where we only show the third level of a pyramid index. Here, we are recruiting eight workers, depicted by the gray circles, who have reported eight locations, depicted by the black triangles. For example, worker w_4 is physically located in the top right cell and has reported the image location in the bottom left cell. In this level, we have d_{max} = 6. Then, each worker answer will be weighted differently. For example, w_1's answer is located at the same cell as its location, i.e., d_{actual} = 0, hence, w_1's answer is weighted as $d_{max} - d_{actual} + 1 = 7$. Going on, w_2, w_3, w_4, w_5, w_6, w_7, and w_8 answers will be weighted as 5, 1, 1, 7, 4, 4, and 4, respectively. Out of these 33 total weights, 20 of them are for the bottom left cell (answers of w_1, w_2, w_4, and w_5). Hence, the local confidence value is $20/33$ = 60%, which is higher than basic Stella (50%).

4.4 Optimization 4: Widening Search Space

Objective. The objective of this optimization is to avoid the case where there is low confidence when identifying an image location.

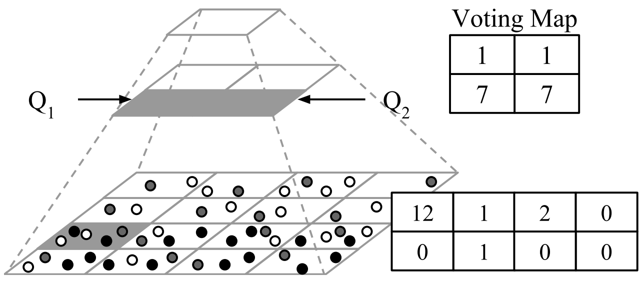


Figure 7: Optimization 4 Example

For example, consider the case where we have 100 workers who cast their votes on the four quarters that are the children of a cell as follows: (50, 45, 3, 2). In basic Stella, we will just select the first quarter as it has the majority of answers, namely 50, though the confidence is only 50%. For a case like this, it will be better if we expand our search horizon to also include the second quarter that receives 45 votes, as this quarter is still promising to contain the image. Should we be able to widen our new search space to include the first two quarters, we would be able to have a 95% confidence instead of the 50% confidence as in basic Stella.

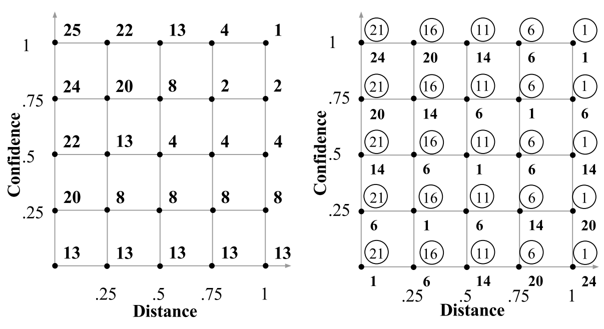
Main Idea. The main idea of this optimization is to go beyond the idea of narrowing down the search space S to only one quarter cell. Instead, we select all quarter(s) that contain more votes than the average votes per quarter as the new search space S, i.e., a quarter will be included in the new S if it has more than 25% of the votes.

Algorithm. We make the following three modifications which are annotated by OPTIMIZATION 4 in the algorithm: (1) Step 1 is modified to select its workers from multiple cells rather than from only one cell as it was the case in basic Stella. In particular, the first action of this step is to divide the number of workers N by the number of cells in S to set the number of workers to be recruited from each cell in S. Then, selecting the set of workers from each cell in S is done in the same way as in basic Stella or in Optimization 1 of our optimized Stella. (2) Step 2 is modified to return back multiple cells inside S, rather than only one cell as it was in basic Stella. Those multiples cells in S are the ones that have more than the average votes per cell.

Example. Figure 7 gives an example of this optimization by using a pyramid of $H = 3$ and $N = 16$. The first root iteration will have 16 workers, depicted as gray circles, who will vote on quarters as (7,7,1,1). Since two of these quarters have more than the average votes per quarter (i.e., 4), we select these two quarters, namely Q_1 and Q_2, for the new S with a local confidence of $(7 + 7)/16 = 87.5\%$. In the second iteration, we allocate only eight workers to each of Q_1 and Q_2, i.e., two workers from each of the eight children of Q_1 and Q_2, depicted as black circles. The voting result is shown in the figure next to the lowest level, where only one cell, Q_3, is selected since it is the only one with above average votes. The third iteration finds the exact location of the image and returns the overall confidence of $(14/16 \times 12/16 \times 1)^{\frac{1}{3}} = 87\%$.

5 EVALUATING STELLA

One may evaluate the quality of a geotagging process based on how close is the resulting location to the actual image location. The closer the distance is, the better the accuracy. However, Stella returns the confidence of the answer in addition to the answer itself to help users to get a good idea on how much they can trust the answer. Yet, this makes it more challenging to evaluate the outcome of Stella. For example, it is clear that the best possible outcome of Stella is to have the exact correct location of an image with 100% confidence. Meanwhile, having a high confidence is not valuable, unless it comes with a highly accurate location as it can mislead the user to trust inaccurate result. For example, it is preferred to have both inaccurate location and confidence rather than inaccurate location with high confidence.



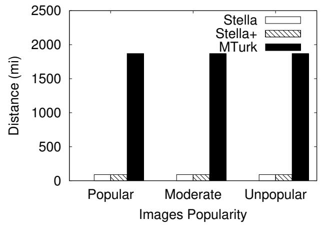
(a) Overall Accuracy Chart (b) Dist & Confidence Chart

Figure 8: Stella Accuracy Chart

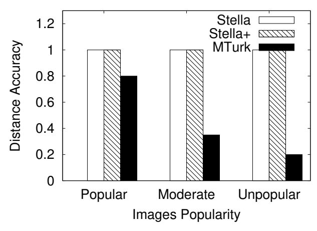
Figure 8(a) illustrates a spectrum of 25 possible answers and how we would evaluate each of them. The x-axis represents the distance accuracy of Stella answer, where 0 is the worst and 1 is the best. The y-axis represents the confidence returned from Stella, also ranging from 0 to 1. The 25 answer points depicted by black circles represent all the 25 combinations of distance accuracies and confidence values. On top of each point, we plot its ranking, with 1 is the best and 25 is the worst. For example, the top rightmost answer (1,1) has the best distance accuracy and confidence. The worst ranked answer (rank 25^{th}) is at (0,1), i.e., a very inaccurate answer with 100% confidence as the confidence misleads the user to trust inaccurate result. The rest of the section describes the concept of *distance accuracy*, *confidence accuracy*, and how we use both accuracies to come up with the *overall accuracy* of Stella, which is basically depicted by the ranking order in Figure 8(a).

Distance Accuracy. Distance accuracy evaluates the spatial location of an image O returned by Stella to the $truth$ location of O where the closer the distance is, the better the accuracy, regardless of the reported confidence. In the case when an answer is returned as an area, we consider $O.loc$ as the centroid of the area. We use a simple linear regression model to calculate the distance accuracy as: $1 - dist(O.loc, O_{actual})/d_{max}$ where $dist()$ is a euclidean distance function between two points and d_{max} is a system parameter that acts as a normalization upper bound distance. If $dist(O.loc, O_{actual})$ is greater than d_{max}, then the distance accuracy is set to 0. Figure 8(b) replicates the same setting of Figure 8(a) while the ranking of the possible answers is only based on the distance accuracy (depicted as numbers inside circles).

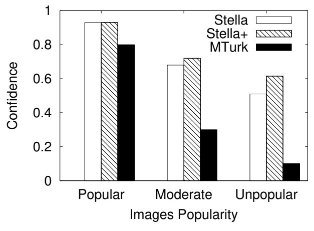
Confidence Accuracy. Given a distance accuracy and a confidence, confidence accuracy measures how accurate the confidence reflects the distance accuracy. For example, a high/low confidence with an



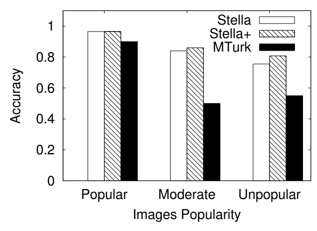
(a) Avg. Worker's Distance



(b) Distance Accuracy



(c) Confidence



(d) Overall Accuracy

Figure 9: Amazon Mechanical Turk Deployment

accurate/inaccurate distance accuracy should result in a good confidence accuracy. Hence, the confidence accuracy is calculated as: $1-|DistAccuracy-C|$, where C is the output confidence. This means that if a distance accuracy of 1 with 100% confidence, the confidence accuracy will be at its maximum value of 1. Similarly, with every combination that has equal distance accuracy and confidence. Meanwhile, the worst confidence accuracy is when a distance accuracy of 0 and C is 100% or when a distance accuracy of 1 and C is 0%. Figure 8(b) also shows the ranking based only on the confidence accuracy (depicted as bold number).

Overall Accuracy. Relying solely on distance accuracy allows two answers with the same distance from the image location to have the same quality, regardless of the resulting confidence. On one hand, if Stella reports an inaccurate location, it is strongly preferable that it gives low confidence. On the other hand, relying solely on confidence accuracy would equate two answers with same accuracy regardless of their distance accuracy. For example, in Figure 8(b), the points (0,0) and (1,1) have the same confidence accuracy, however, the latter one is strongly preferred as it provides the accurate location. We evaluate the *overall accuracy* as a linear combination of both accuracies as: $\alpha \times DistAccuracy$ + (1 - α) $\times$ $ConfAccuracy$, where $0 \leq \alpha \leq 1$ is a system parameter to weight the importance of each accuracy. The overall ranking depicted in Figure 8(a) is calculated based on having $\alpha = 0.5$.

6 EXPERIMENT

This section presents our experiment of Stella: basic Stella (termed Stella) and the optimized Stella (termed Stella+). To the best of our knowledge, Stella is the first framework that solves the geotagging problem without relying on having a rich set of training dataset. Hence, it is not comparable to other geotagging frameworks. Instead, we first evaluate the idea of Stella by deploying it on top of a commercial crowdsourcing platform, namely Amazon Mechanical Turk (termed MTurk), to compare its performance to the general deployment of Amazon Mechanical Turk in Section 6.1. Then, we evaluate the impact of each optimization of Stella in Section 6.2.

In all our experiments, we use four evaluation metrics: (1) The average distance between the worker location and the true location of an image O. This gives an indication on how successful each technique is in recruiting domestic workers. (2) The distance accuracy, i.e., how far is the reported location of an image O from its true location. (3) The confidence value, where the higher the confidence is the better. This gives an indication of how each approach is confident with its result. (4) The overall accuracy, which combines both the distance and confidence accuracies.

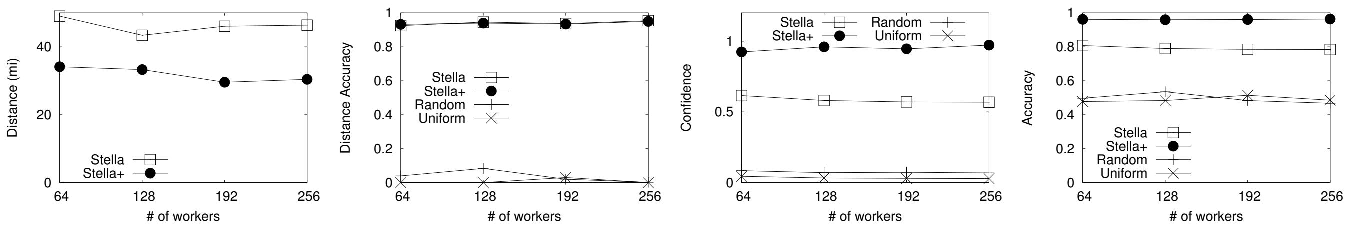
Note that we do not include latency in our evaluation metric. The main reason is that Stella is more about batch processing of geotagging and is not a real-time geotagging framework. While Stella may incur extra penalty in latency in comparison with other techniques, i.e., *upper-bounded* by the number of iterations $\times$ the latency of a single crowdsourcing deployment, it is currently the only framework that is able to geotag images with high accuracy while other approaches fail to do so. From our real experiment with Amazon Mechanical Turk, Stella often completes the geotagging task much earlier than its upper bound as we only deploy a subset of the total workers in each iteration, thus, resulting in less turn around time. Furthermore, for some images, Stella+ is able to minimize the latency further due to its second optimization.

6.1 Amazon Mechanical Turk Deployment

Experimental setup. We selected 20 images and geotagged them in a total of 150+ crowdsourcing tasks by using 600+ workers with a reward of \$0.05 per assignment. To ease the readability of the paper, based on the results, we categorize every image into three categories: *popular*, *moderately popular*, and *unpopular* images. A popular image is an image that more than 50% of the workers agree on its exact location, and thus MTurk can geotag the image accurately with majority voting. A moderately popular image is an image that more than 50% of workers are aware with its location, e.g., they are aware in which state of the US that the image is located. We categorize the rest as unpopular.

We first try to geotag every image using a major web search engine. However, it was only able to find the location of popular images, thus, having 0% accuracy for both moderately popular and unpopular images. To ensure a fair comparison, we simulated both Stella approaches using a real deployment on top of MTurk. Since MTurk only allows us to select workers from a certain country or state if the workers live in the US, we simulated Stella in a total of three iterations: (1) We recruit workers within the US and map their results into four non-overlapping regions. (2) We recruit workers all from the majority region to find the state of the image. (3) We recruit workers all from the majority state to find the city of the image. We also ask the workers to provide their zip-code to see how domestic the recruited workers are. For each image, we use a total of 30 workers to geotag with all three approaches.

Experimental Results. Figure 9(a) gives the average worker's distance for each approach. Both Stella and Stella+ use the average distance of the workers that we recruit at the last iteration as they are the ones that provide the final location of the images. For every image popularity, Stella and Stella+ are able to recruit significantly



(a) Avg. Worker's Distance (b) Distance Accuracy (c) Confidence (d) Overall Accuracy

Figure 10: Stella's Overall Performance

more domestic workers than MTurk. Specifically, both Stella variants and MTurk are able to recruit workers with an average distance of 88 miles and 1,879 miles, respectively. There is no performance difference between Stella and Stella+ as the interface of MTurk only allows us to recruit workers within a state, thus, they recruit a similar subset of workers for every image.

Figure 9(b) gives the distance accuracy in geotagging images across various image popularities. We set the maximum distance threshold, d_{max}, as the diagonal length of the state where each image is located. Instead of finding the centroid of an MBR that covers all workers' answers, we calculate the distance accuracy of MTurk as the average distance accuracy among the recruited workers. The reason is that the MTurk answers often give different locations which increases the size of the MBR that covers all of those answers. In such case, the centroid of the MBR is no longer represents the workers' answers accurately. For example, if 9 out of 10 workers choose Chicago, IL, and 1 worker chooses Boston, MA, as the location of an image, then the centroid of the MBR that covers all 10 answers is located somewhere around Cleveland, OH. As shown in the figure, both Stella and Stella+ significantly outperform MTurk for all image popularities. For popular images, while 80% of MTurk workers are able to provide the accurate city of the images, the rest of the workers provided locations that are very far. Furthermore, the distance accuracy of MTurk is worsen for both moderately popular and unpopular images where it has an accuracy of 35% and 20%, respectively. In contrast, Stella and Stella+ are able to maintain their distance accuracy across all image popularities. There is no difference in the distance accuracy between Stella and Stella+ as they are able to narrow down the accurate state of the images and recruit domestic workers to provide each image final location.

Figure 9(c) and Figure 9(d) give the confidence and overall accuracy of each approach across different image popularities, respectively. The confidence of MTurk is calculated as the ratio of workers who agree on the final answer within a city level over the total number of answers. Both Stella and Stella+ consistently outperform MTurk, as they have more agreement among workers. The improvement of Stella+ in all three image popularities is mainly due to its third optimization that assigns a different weight for each worker's answer. In Figure 9(d), we can see that Stella and Stella+ consistently have higher overall accuracy than MTurk. However, we also see that the overall accuracy of MTurk is better than its distance accuracy (Figure 10(b)) and confidence (Figure 10(c)). The reason is that MTurk has a good confidence accuracy where it reports a low accuracy result when it is not confident in its result. Contrast such result to both Stella approaches where they have a high overall accuracy due to its high distance accuracy and confidence.

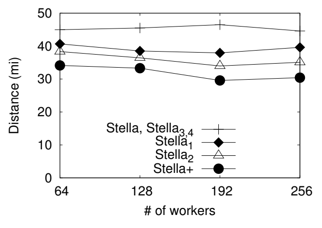
6.2 Standalone Stella Deployment

Experimental setup. Due to the limitation of the Amazon Mechanical Turk interface, we need to run our detailed evaluation of Stella in a different environment. Similar to prior research that used real non-crowdsourcing datasets as workers' locations [9], we use Foursquare dataset [12], consists of 1.7+ million users, as the workers that we can recruit. Then, we follow the *distance-aware quality* model, introduced in [9], which models the accuracy of a worker based on an exponential function. In particular, the input to the function is the distance between a worker w to an image, d_w, and the function will output the distance between w's answer to the image, a_w. $a_w = 0$ means that the answer of w is located at a zero distance from the image location, thus, is highly accurate. We generate such function by first plotting the accuracy of the 600+ workers from our Amazon Mechanical Turk experiment in Section 6.1 where the x-axis is the d_w and the y-axis is the a_w. Then, we generate an exponential trend line that fits the plot which results in an exponential function of: $a_w = 1.431e^{0.059d_w} - 1$.

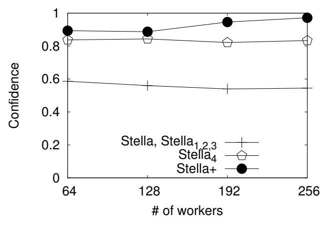
Unless mentioned otherwise, we use a total of 128 workers, a pyramid index of six levels which covers the mainland of USA, and a default $\alpha = 0.5$ for our accuracy evaluation to weight distance and confidence accuracy equally. For our second optimization, we set $x = 80\%$. To minimize the inconsistency that may be resulted, we run each experiment 100× on a machine with Intel Quad Core i7-4790 3.6Ghz, two threads per core, and 32GB of RAM running 64-bit Ubuntu 16.04.

Overall Performance. We compare the end to end performance of Stella and Stella+, with two basic crowdsourcing techniques: *random* and *uniform* workers assignment. Random selects workers randomly. Meanwhile, uniform first divides the space into equal grids, where the number of grid cells is equal to the number of workers that it will assign, then it randomly selects one worker within each grid cell. For each experiment, we randomly select a location as the image location and varies the number of workers that we recruit from 64 workers to 256 workers.

Figure 10(a) gives the average recruited worker's distance to an image. Stella and Stella+ recruit workers with an average distance of 46 and 32 miles, respectively. We omit the the average worker's distance of the two basic approaches as random and uniform has a very large distance of 958 and 1,341 miles, respectively. Stella+ is consistently able to recruit 30% more domestic workers than Stella which is resulted from its first two optimizations. We also see that there is not much difference in the average recruited worker's distance when varying the number of workers as both Stella approaches still recruit workers in the same search space.

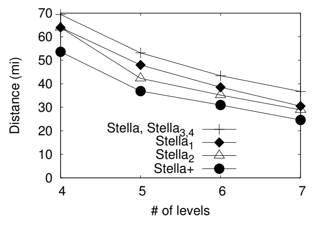


(a) Avg. Worker's Distance

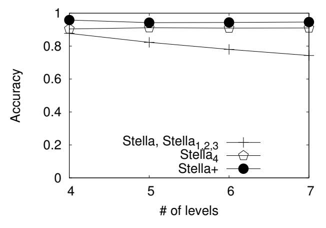


(b) Confidence

Figure 11: Internal Performance Varying # Workers



(a) Avg. Worker's Distance



(b) Overall Accuracy

Figure 12: Internal Performance Varying # Levels

Figure 10(b) gives the distance accuracy of every approach. Both Stella approaches are consistently able to outperform both basic approaches by having an average 90% more accuracy than random and uniform. Both Stella and Stella+ also have a similar distance accuracy between each other. The reason is that distance accuracy is calculated only based on the last iteration results and since both approaches are able to narrow down search space to the accurate one, there is no difference in their distance accuracy.

Figure 10(c) gives the overall confidence of every approach. We calculate the confidence of the basic approaches by projecting their results into the lowest pyramid level and taking the ratio of answers that fall into the cell that contains the majority of answers. Both Stella and Stella+ outperform the two basic approaches by achieving 50% and 90% more confidence, respectively. Stella+ outperforms Stella by having 40% more confidence as Optimizations 1 and 2 allow Stella+ to recruit more domestic workers while Optimization 3 gives higher weight to their answers. The confidence also directly impact the overall accuracy of each approach as can be seen in Figure 10(d), where Stella, Stella+, and the two basic approaches have 96%, 79%, and 50% overall accuracy, respectively.

Internal Performance. We compare six variants of Stella in this section: the basic Stella, the optimized Stella+, Stella with Optimization 1 only (Stella$_1$), with Optimization 2 only (Stella$_2$), and so on. For ease of readability, we combine several variants of Stella into one series if they have a similar performance. In this experiment, we also study the effect of varying the number of iterations of each approach in addition to varying the number of workers.

Figure 11(a) gives the average worker's distance to an image by varying the number of workers. Both Stella$_1$ and Stella$_2$ are able to assign more domestic workers than basic Stella. This confirms our idea of Optimization 1 that by recruiting workers distributed in a skewed way that matches the answers distribution, we will be able to get more domestic workers. Optimization 2 is able to assign more domestic workers than Stella as it is able to skip some of the iterations and use the skipped budget for further iterations to recruit 25% more domestic workers. Combining these two optimization results in a better performance as shown by Stella+ which outperforms Stella by recruiting 35% more domestic workers.

Figure 11(b) gives the overall confidence for each variant of Stella. Stella$_4$ is able to achieve 20% more confidence than Stella as it is able to select more than one cell as the new search space. However, Stella$_3$ does not have any impact to the overall confidence. The reason is that there is less agreement between workers at the lower pyramid level which is caused by from the accuracy model that we extract. For example, even with $d_w = 0$, the answer will have an error distance of $a_w = 0.431$ in lat/lon degree which, in our case, is roughly 23.4 miles while each cell has a diameter of 55 miles. However, by combining Optimizations 3 and 4 together, we achieve a better confidence as shown by Stella+. The reason is that Optimization 3 now has an impact in calculating the confidence as workers' answers can agree on more than one cell due to Optimization 4.

Figure 12(a) gives the average worker's distance that each variant of Stella assigns by varying the number of iterations. All variants are able to recruit more domestic workers as they traverse deeper into the pyramid which confirms our adaptive crowdsourcing concept. Both Stella$_1$ and Stella$_2$ are able to assign more domestic workers than Stella due to their optimizations which are geared towards recruiting more domestic workers. Meanwhile, Stella+ outperforms other variants of Stella as its third optimization weights those domestic workers' answers more. Figure 12(b) gives the overall accuracy of all variants. The overall accuracy of Stella, Stella$_1$, Stella$_2$, and Stella$_3$ decrease with deeper pyramid levels as there is less agreement between workers due to the the accuracy model that we extract. However, Stella$_4$ is able to maintain its accuracy as it expands its search space in the lower pyramid levels. However, by combining Optimizations 3 and 4 together, Stella+ consistently outperforms Stella$_4$.

7 RELATED WORK

Crowdsourcing Frameworks. Many crowdsourcing efforts are focused on providing efficient techniques to solve different types of machine-hard tasks. Examples include integrating crowdsourcing into the query plan of a Database [6, 13], using the crowd to sort and join data [13], to compute skyline over noisy dataset [15], for real-time image search [32], and many more. However, up to our knowledge, crowdsourcing has not been used for geotagging as it is first deemed unfit to be solved by the crowd due to the low accuracy of the result. In this paper, we provide the first framework that leverages crowdsourcing for geotagging. Task assignment, i.e., studying on how to assign a task to a set of workers, is a very important problem and has been widely studied [5, 29, 33]. [5] assigns a task to a worker that has completed similar tasks with high performance. In geotagging, the only information to predict such performance metric is to check the worker's performance in solving other tasks that are co-located with the new task. However, we do not know the location of the new task and in fact, this is what we are trying to find. [33] assigns tasks that results in the highest improvement in quality by representing the possible answer of a task in a matrix. This approach will only work when there is a limited amount of answers for a task, e.g., a label of "equal" or "not equal" in an entity resolution, which is not the case in geotagging as every location can be treated as an answer. Our task assignment technique is closely related to

[29] where it tries to maximize the overall utility, i.e., the location coverage in our case, given a fixed budget. The main difference is that [29] spends all of its budget at once and thus, the best result it can get is the first iteration of Stella.

Spatial Crowdsourcing. Several works have been conducted in studying spatial crowdsourcing where each crowdsourced task contains a spatial information about the task and/or the workers. Spatial crowdsourcing frameworks [10, 14, 27, 28] require workers to physically go to the location of the task, e.g., take a picture of an object. However, these applications use one main assumption that the task's location to be known in advance. Similar to these techniques, our work tries to find the workers who are located near the task. However, without knowing the location of the image in advance, existing techniques are not fit to solve the geotagging problem.

Object Geotagging Techniques. Many works have been conducted to geotag an object by using natural language processing (NLP) [30], incorporating user profile [7], or using machine learning techniques [31]. However, they suffer from two main limitations: (1) Each object type, e.g., text and image, requires its own tailored solution. For example, NLP techniques cannot geotag images while computer vision technique cannot geotag tweets. (2) The answer quality mainly rely on having prior knowledge, e.g., an accurate training dataset. With Stella, we do not need to create a specific solution for each object type and we do not rely on having a large accurate training datasets. In fact, some of these approaches can use Stella to create an accurate training dataset for their techniques.

8 CONCLUSION

We presented Stella; a crowdsourcing-based framework for image geotagging. Given an image and budget, Stella finds the spatial location of the image by asking the crowd within the given budget. In order to geotag an image accurately, Stella identifies and recruits domestic workers, i.e., workers who live nearby the location of the image, despite of its unknown location. Stella overcomes this dilemma by introducing a novel crowdsourcing technique, called adaptive crowdsourcing, that gradually understands the image location. The main idea is to split the crowdsourcing process into multiple iterations and use a subset of the total workers in each iteration. Then, based on the result, Stella earns more knowledge about the image whereabouts and recruits another subset of the workers that are more domestic than the previous ones. In the last iteration, Stella will use the last set of workers to find the exact location of the image. Stella also reports back its confidence in geotagging the image to help the requester in understanding the result quality. Stella is equipped with four optimization techniques to further enhance its accuracy and performance. Extensive experimental evaluation of Stella using real settings and real datasets show that Stella is able to achieve an average of 95% accuracy as well as 90% confidence.

REFERENCES

[1] 2010. Location Isn't Just a Feature Anymore, It's a Platform. https://www.wired.com/2010/03/location-isnt-just-a-feature-anymore-its-a-platform/.

[2] Majid Alivand and Hartwig Hochmair. 2013. Extracting scenic routes from VGI data sources. In *SIGSPATIAL*.

[3] Jeremy W Crampton, Mark Graham, Ate Poorthuis, Taylor Shelton, Monica Stephens, Matthew W Wilson, and Matthew Zook. 2013. Beyond the geotag: situating 'big data' and leveraging the potential of the geoweb. *Cartography and geographic information science* 40, 2 (2013), 130–139.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*.

[5] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. 2015. iCrowd: An adaptive crowdsourcing framework. In *SIGMOD*.

[6] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: Answering Queries with Crowdsourcing. In *SIGMOD*.

[7] Mark Graham, Scott A Hale, and Devin Gaffney. 2014. Where in the World Are You? Geolocation and Language Identification in Twitter. *The Professional Geographer* (2014).

[8] Claudia Hauff. 2013. A study on the accuracy of Flickr's geotag data. In *SIGIR*.

[9] Huiqi Hu, Yudian Zheng, Zhifeng Bao, Guoliang Li, Jianhua Feng, and Reynold Cheng. 2016. Crowdsourced POI Labelling: Location-aware result inference and task assignment. In *ICDE*.

[10] Christopher Jonathan and Mohamed F Mokbel. 2017. Towards a Unified Spatial Crowdsourcing Platform. In *SSTD*.

[11] John Krumm and Eric Horvitz. 2015. Eyewitness: Identifying local events via space-time signals in twitter feeds. In *SIGSPATIAL*.

[12] Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F Mokbel. 2012. Lars: A location-aware recommender system. In *ICDE*.

[13] Guoliang Li, Chengliang Chai, Ju Fan, Xueping Weng, Jian Li, Yudian Zheng, Yuanbing Li, Xiang Yu, Xiaohang Zhang, and Haitao Yuan. 2017. CDB: Optimizing Queries with Crowd-Based Selections and Joins. In *SIGMOD*.

[14] Wei Li, Haiquan Chen, Wei-Shinn Ku, and Xiao Qin. 2017. Scalable Spatiotemporal Crowdsourcing for Smart Cities based on Particle Filtering. In *SIGSPATIAL*.

[15] Christoph Lofi, Kinda El Maarry, and Wolf-Tilo Balke. 2013. Skyline queries in crowd-enabled databases. In *EDBT*.

[16] Amr Magdy, Louai Alarabi, Saif Al-Harthi, Mashaal Musleh, Thanaa M Ghanem, Sohaib Ghani, Saleh Basalamah, and Mohamed F Mokbel. 2015. Demonstration of Taghreed: A system for querying, analyzing, and visualizing geotagged microblogs. In *ICDE*.

[17] Omar Montasser and Daniel Kifer. 2017. Predicting Demographics of High-Resolution Geographies with Geotagged Tweets. *AAAI* (2017).

[18] Yusuke Nakaji and Keiji Yanai. 2012. Visualization of real-world events with geotagged tweet photos. In *ICMEW*.

[19] Barak Pat and Yaron Kanza. 2017. Where's Waldo? Geosocial Search over Myriad Geotagged Posts. *SIGSPATIAL* (2017).

[20] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. *ICWSM* (2011).

[21] Hanan Samet, Jagan Sankaranarayanan, Michael D Lieberman, Marco D Adelfio, Brendan C Fruin, Jack M Lotkowski, Daniele Panozzo, Jon Sperling, and Benjamin E Teitler. 2014. Reading news with maps by exploiting spatial synonyms. *CACM* (2014).

[22] Charles W Schmidt. 2012. Using social media to predict and track disease outbreaks. *Environmental health perspectives* (2012).

[23] Donglian Sun, Sanmei Li, Wei Zheng, Arie Croitoru, Anthony Stefanidis, and Mitchell Goldberg. 2016. Mapping floods due to Hurricane Sandy using NPP VIIRS and ATMS data and geotagged Flickr imagery. *International Journal of Digital Earth* (2016).

[24] Steven Tanimoto and Theo Pavlidis. 1975. A hierarchical data structure for picture processing. *Computer graphics and image processing* (1975).

[25] Mike Thelwall. 2016. The precision of the arithmetic mean, geometric mean and percentiles for citation data: An experimental simulation modelling approach. *Journal of informetrics* 10, 1 (2016), 110–123.

[26] Hien To, Cyrus Shahabi, and Li Xiong. 2018. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *ICDE*.

[27] Yongxin Tong, Lei Chen, and Cyrus Shahabi. 2017. Spatial crowdsourcing: challenges, techniques, and applications. *PVLDB* (2017).

[28] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*.

[29] Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R Jennings. 2014. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence* 214 (2014), 89–111.

[30] Isis Truck and Mohammed-Amine Abchir. 2017. Natural Language Processing and Fuzzy Tools for Business Processes in a Geolocation Context. *Advances in Artificial Intelligence* (2017).

[31] Tobias Weyand, Ilya Kostrikov, and James Philbin. 2016. PlaNet-Photo Geolocation with Convolutional Neural Networks. In *ECCV*.

[32] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. 2010. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*.

[33] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. 2015. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*.