A Demonstration of Stella: A Crowdsourcing-Based Geotagging Framework

Christopher Jonathan
Department of Computer Science and Engineering
University of Minnesota, MN, USA
cjonathan@cs.umn.edu

Mohamed F. Mokbel
Department of Computer Science and Engineering
University of Minnesota, MN, USA
mokbel@cs.umn.edu

ABSTRACT

This paper demonstrates $Stella$; an efficient crowdsourcing-based geotagging framework for any types of objects. In this demonstration, we showcase the effectiveness of $Stella$ in geotagging images via two different scenarios: (1) we provide a graphical interface to show the process of a geotagging process that have been done by using Amazon Mechanical Turk, (2) we seek help from the conference attendees to propose an image to be geotagged or to help us geotag an image by using our application during the demonstration period. At the end of the demonstration period, we will show the geotagging result.

1. INTRODUCTION

Attaching a real-world geographic location to any objects (e.g., image or tweet), a process known as *geotagging*, enables a myriad of important practical applications. For example, web search engines use geotagged images and websites for enhanced search experience [10] and news agencies place geotagged news items on a map for news analysis [6]. Meanwhile, several research efforts have used the locations of images or tweets to discover local events [2], identify scenery routes [3], find out flooded areas within hurricanes [8] and track food poisoning incidents [7], among many other applications. However, unfortunately, the lack of available geotagged data significantly reduce the efficiency and accuracy of all such applications. For example, only 0.7% of tweets are geotagged [4], around 4.8% of Flickr images are geotagged [5], while images searchable by web search engines are mostly not geotagged.

Motivated by the importance of applications that need geotagged data and the lack of such data, several approaches are proposed for geotagging from both commercial and research efforts, e.g., see [1, 11]. One may geotag a text-based data with natural language processing [1] and geotag images with computer vision [11] or commercial web search engine, e.g., Google Images . However, existing techniques suffer two major limitations: (1) each technique is tailored toward geotagging one specific data type and (2) existing techniques rely on having prior knowledge, e.g., a large training datasets, which still need another technique to provide them.

In this paper, we demonstrate $Stella$; an efficient crowdsourcing-based geotagging framework for any types of objects. The main idea is to recruit *domestic* workers, i.e., workers who are located near the object to be geotagged, to geotag the object. Our hypothesis is that *domestic* workers will have more knowledge about the accurate location of the object than *non-domestic* ones. Unfortunately, we do not know the location of the object and in fact, this is what we are looking for. $Stella$ overcomes this dilemma by using a novel crowdsourcing approach that aims to gradually understand the object location, called *adaptive crowdsourcing*. The main idea of adaptive crowdsourcing is that we do not use the whole crowdsourcing budget at once. Instead, we split the process into multiple iterations where the goal of each iteration is to narrow down the possible location of the object. For example, in the first iteration of the geotagging process, $Stella$ tries to infer the continent of the object by asking the workers worldwide. Then, we try to infer the country of the object by asking workers, however, now all recruited from the resulting continent of the previous iteration and so on. As we iterate more, we will be able to assign more *domestic* workers. Furthermore, $Stella$ is also equipped with a set of optimizations that are geared towards increasing the likeliness of recruiting more *domestic* workers, hence, obtaining more accurate answer.

In addition to returning location of the object, $Stella$ associates a *confidence* value to its final result. Such value indicates how confident $Stella$ is in geotagging the given object. The confidence is calculated based on various aspects that include the diversity in workers' solutions as well as the ratio of *domestic* workers. *Confidence* value adds an extra dimension for the requester to decide whether she can trust the result that is returned by $Stella$ or not. Contrast such approach with any existing crowdsourcing frameworks which only return back the result to the requester which she does not have any idea on how trustworthy the result is.

We demonstrate $Stella$ in two scenarios: (1) We ran a pre-simulated $Stella$ on Amazon Mechanical Turk while collecting the geotagging process, such as the workers' locations. Then we show the geotagging process of to the attendees via an interactive visualization. (2) We welcome the VLDB 2017 conference attendees to provide us an image that they want to geotag. Furthermore, we provide an interface for the attendees to help us in geotagging some of the images and experience the geotagging process with $Stella$. At the end, we will show the geotagging results as well as the score of each attendee's answer which is calculated based on the agreement between her answer and the other attendees' answers.

*This work is partially supported by the National Science Foundation, USA, under Grants IIS-1525953, CNS-1512877, IIS-1218168, and IIS-0952977.

https://images.google.com/

Proceedings of the VLDB Endowment, Vol. 10, No. 12

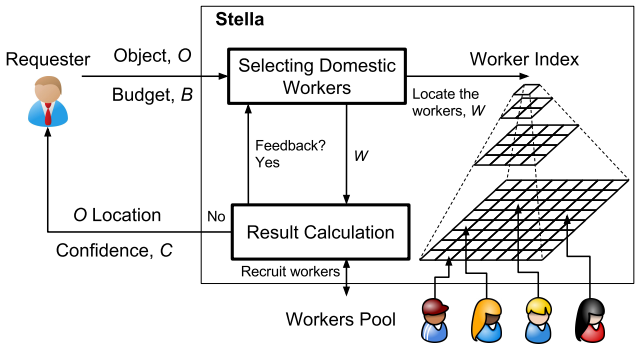


Figure 1: *Stella* **System Architecture**

2. SYSTEM OVERVIEW

Figure 1 gives the system architecture of *Stella*. A user submits an object O that needs to be geotagged and a budget B that she is willing to pay. The answer returned to the user contains the location of the object and a confidence value C that tells how much *Stella* is confident about $O.loc$ to be accurate. For efficient retrieval of workers within a certain area, *Stella* maintains its workers' locations in a pyramid index structure [9] with a system parameter H as its height. The first level (i.e., root) has only one cell, then, generally, the h^{th} level has 4^{h-1} non-overlapping equal cells covering the whole space. Rather than storing the exact location of each worker, we store each worker's location information in one of the cells on the lowest level of the pyramid in order to preserve the worker's location privacy.

Internally, the main challenge that *Stella* faces is how to find and recruit *domestic* workers without knowing the location of O. *Stella* addresses this challenge by introducing the idea of *adaptive crowdsourcing*. Unlike conventional crowdsourcing platforms that assign a given task to all B workers at once, *Stella* assigns the geotagging task into H iterations, where H is the height of the pyramid and to only a subset of B workers in each iteration. Then, based on the result of an iteration, *Stella* earns more knowledge about the whereabouts of O. Then, in the next iteration, it recruits another subset of the B workers that are more *domestic* than the previous ones by examining the result of the previous iteration. This process is depicted in Figure 1 by the iterations over two main internal modules in *Stella*namely, *Selecting Domestic Workers* and *Result Calculation* which will be discussed in next section.

As it is the case for any crowdsourcing platforms, workers need to have an incentive to participate in a crowdsourcing task. There are many well known methods for this, including monetary incentives where workers are paid by completing the task, gamification where workers participate in a task as part of playing a game, and badge awards where workers are getting higher badges the more they participate. The study of workers incentive methods is beyond the scope of *Stella*. *Stella* assumes that any given task has a budget B as number of workers that need to be assigned to this task, regardless of how to convince these B workers to participate.

3. BASIC STELLA

This section describes the our basic *Stella*: the simplest form of our proposed framework. Algorithm 1 gives the high level overview of *basic Stella*. First, *Stella* sets the possible search space S as the root cell of the pyramid index P, i.e., the object O could be anywhere in the whole space, with a 100% confidence value. With the concept of *adaptive crowdsourcing*, *Stella*

Algorithm 1 Basic *Stella*

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

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 main modules: (1) *Selecting Domestic Workers*: Given the current search space S and the number of workers N, this step finds a set W of N workers, all located within S, that will be assigned the geotagging task in this iteration. (2) *Result Calculation*: In this step, *Stella* sends the geotagging task to the N workers that are assigned by the previous step. The workers' answers are aggregated to form three outputs, namely, the potential location of O, the current iteration confidence value, and a new narrower search subspace S to be used in the next iteration to recruit more *domestic* workers.

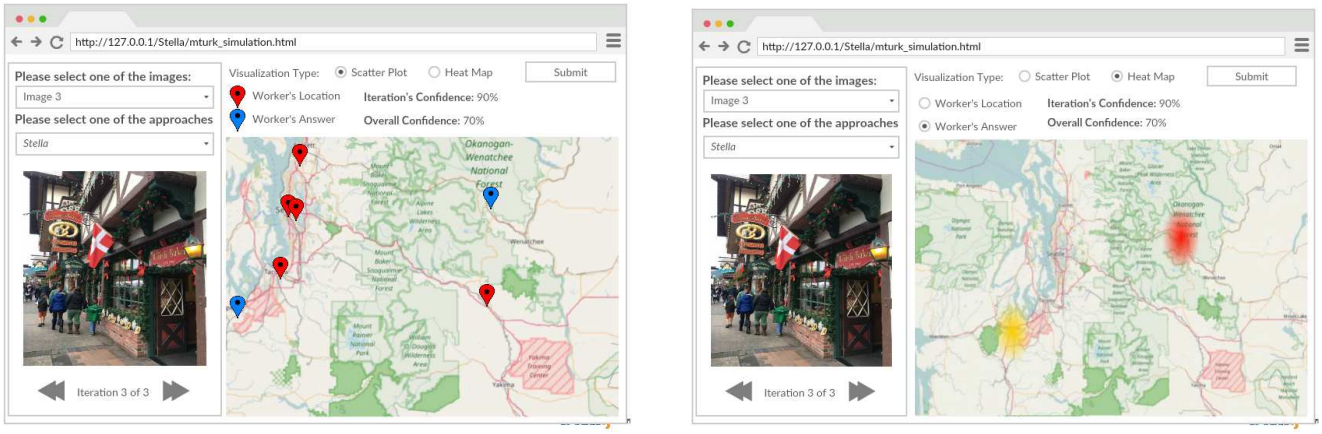
Each iteration in *Stella* has a search space S that includes the location of O and covers a subset of the search space of the previous iteration. In the last iteration, *Stella* will return an empty S, breaking out of the loop to output the latest location and the aggregated confidence values of all iterations. With a finer and smaller space that S covers in each iteration, *Stella* is able to *adaptively* understand the object location, and hence recruit more *domestic* workers. This gives more accurate location as well as better confidence value. The rest of this section discusses the details of the two modules of *Stella*.

3.1 Selecting Domestic Workers

This module takes two inputs, namely, the current search space for the object S and the number of workers that it can recruit in this iteration N. The goal is to find N *domestic* workers located within the provided search space S, and send their information to the second module. Our first knowledge of the location of O is that it can be anywhere within S. We do so by exploiting the structure of the pyramid index for the cells rooted at S and recruit $N/4$ workers from each of the children cells of S. For each child cell, we recursively traverse the children of each child cell until we do not have any workers that we can recruit from a cell or we reach the lowest level of the pyramid. 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 rather than selecting the workers from the center of the cell. The idea is to avoid overloading workers in the center and to avoid having workers fake their locations to be in a center of a cell in order to be assigned more tasks.

3.2 Result Calculation

This module takes the set of workers from the previous module and sends to them the object O that needs to be geotagged. The answer that is received from each worker in the form of (*latitude*, *longitude*). Unless this is the last iteration, the objective is to find a smaller search space of O based on the results that it receives from the workers and send this information back to the first module. In



(a) Scenario 1: Scatter Plot

(b) Scenario 1: Heat Map

Figure 2: Scenario 1: User Interface

particular, we will select one of the children cells of the current search space S that contains most of the workers' answers. Thus, we do not care much about the exact (*latitude*, *longitude*) of each answer. Then, we use a simple majority voting to choose the child cell of S that contains the most answers. If this is the last iteration, the module will provide the final location of the object by finding the minimum bounding rectangle (MBR) of the answers and return this MBR back to the user.

In each iteration, this module will also calculate the *confidence* value of each iteration, i.e., the *local confidence* value. The *local confidence* value is calculated based on the ratio of workers who voted for the selected child cell over the size of the results. With H iterations in total, there are H *local confidence* values, as one per iteration. At the final iteration, $Stella$ will calculate the *overall confidence* value by finding the geometric mean of the H *local confidence* values and return it to the user as the confidence of $Stella$ in geotagging the object.

4. OPTIMIZED STELLA

$Stella$ is equipped with optimizations that are geared to recruit more *domestic* workers the ones recruited by the basic $Stella$, which will end up in increasing the overall confidence and accuracy of the final result, described briefly below:

Recruiting More Domestic Workers. This optimization aims to recruit workers that are more *domestic* than the workers that are recruited by the basic $Stella$ in its first module. With this optimization, $Stella$ uses the workers' answers from the previous iteration to guide its decision in recruiting workers in each child cell of the current search space rather than uniformly recruiting workers for each child cell. Then, the number of workers that are recruited from each child cell of S will be based on the distribution of the workers' previous answers on these cells.

Skipping Iterations. This optimization goes beyond the idea of traversing the index one level in each iteration. Instead, we smartly skips several levels for those objects that there is an early agreement on their whereabouts. Then, we reuse our workers budget from the skipped iterations to recruit more *domestic* workers in further iterations, thus, increase the accuracy. For example, in geotagging an image that depicts the Liberty Statue in New York City, NY, USA; by just asking workers worldwide, we might be able to infer the city of the image right away without a need to ask workers to find the country and the state of the image. Then, we can use the budget of the skipped levels to recruit additional workers within the city of New York to find the exact location of the image.

Weighted Confidence. This optimization aims to achieve a higher confidence value by going beyond the idea of weighting all workers' answers equally. Instead, it considers that if w_i is closer to her answer than w_j to her answer, then w_i must be more confident than w_j in providing the answer. Hence, we will weight w_i answer higher when calculating the *local confidence* of an iteration which will also increase the *overall confidence*.

Expanding Search Space. This optimization aims to enhance the confidence for cases where the is no clear majority among workers' answers. For example, consider the case where we have 100 workers who cast their votes on the four quarter of a cell to be: (50, 45, 3, 2). By choosing only the cell that contains the majority voting will result in a *local confidence* of 50%. Thus, $Stella$ decides to expand its search space to also include the second quarter that receives 45 votes, as this quarter is still promising to contain the object. In particular, $Stella$ decides to include all cells that contain more votes that the average votes that each cell receive.

5. DEMONSTRATION SCENARIOS

This section presents our two demonstration scenarios for $Stella$. In the first scenario, we manually deploy $Stella$ on Amazon Mechanical Turk by running multiple geotagging tasks for different images. In this scenario, we will visualize the geotagging process of the basic $Stella$, the optimized $Stella$, as well as the regular Amazon Mechanical Turk deployment. In the second scenario, we welcome the VLDB 2017 conference attendees to provide us with an image that we will geotag with the help of the attendees. Near the end of the demonstration period, we will show the geotagging process of the image and release the resulted location of the image. Furthermore, we will also provide the score of each attendee's answer during the geotagging process.

5.1 Scenario 1: Amazon Mechanical Turk Deployment

In the first scenario, we manually simulate and deploy $Stella$ on Amazon Mechanical Turk (MTurk) to recruit its workers in geotagging different images. However, we are unable to do a live demonstration of the Amazon Mechanical Turk deployment during the

https://www.mturk.com/

demonstration period. The reason is that crowdsourcing environment is not geared towards real time application where user can submit a task and retrieve the result simultaneously as we need to wait for the workers to accept the task and do it. As a result, we decide to pre-run the geotagging process in offline while we record all the informations throughout the geotagging process, including each recruited worker's location, each worker's answer, and the confidence. Then, we will visualize the geotagging process where the conference attendees will be able to see the location of the workers that we recruited as well as their answers.

Since MTurk only allows selecting workers from a certain country or a certain state if they live in the United States, we simulated $Stella$ using MTurk through three iterations as follows: First, we recruit workers within the United States and map their results into four different regions. Then, we recruit workers all within the region with majority voting to find the state of the image. Finally, we recruit workers all from the state with majority voting to find the city of the image. We also ask the workers to provide their zip code to see how *domestic* the workers are.

Figure 2 depicts two screenshots from our user interface in simulating $Stella$ on Amazon Mechanical Turk. Users will be able to use the two drop down panels on the top left pane of the screen to list down different options of images that we were geotagging and the geotagging methods that we used. In this simulation, we used three geotagging methods, namely by using basic $Stella$, by using $Stella$ with its four optimizations, and by using Amazon Mechanical Turk as is which assigns random workers to geotag the image. Once the user chooses one of the images and one of the methods, the interface will visualize the geotagging process. In particular, the visualization shows the location of each worker that is recruited by each method as well as the location of each worker's answer. We provide two types of visualization that the user can choose: the scatter plot visualization as shown in Figure 2(a) and the heat map visualization as shown in Figure 2(b). By using the left and right arrows on the bottom left pane of the interface, user can choose which iteration that she wants to visualize.

5.2 Scenario 2: VLDB 2017 Live Geotagging

In the second scenario, we welcome the VLDB 2017 attendees to help us geotag an image as well as provide us an image to geotag. For each submission, the user will provide us four inputs: the user's name, the city, the country where she lives, and the predicted location of the image. We will use the city and the country information of the user as the worker's location during the geotagging process. Figure 3 shows the user interface for a user to geotag an image. The user will first provide us her name as well as the city and country where she lives by using the three input boxes on the top left pane. Then, she can choose the image that she wants to geotag from the drop down list below it. Once the user decides on an image, we ask the user to drop a pin on a map on the right side of the screen. The user can zoom in and zoom out of the map to find the desired location. The user can also upload the picture that she wants to geotag with the help of the VLDB attendees by using the upload button located on top right of the screen.

All attendees' submissions will be recorded in our framework and we will use these to simulate the geotagging process for each image near the end of the demonstration period. In particular, in the first iteration, the framework will first choose a subset of users' answers based on their reported locations and use their responses to get the clue of the image for the second iteration. In the second iteration, it will choose another subset of records, however, now all within the boundary of the first iteration's result to get a better clue on the location of the image and so on.

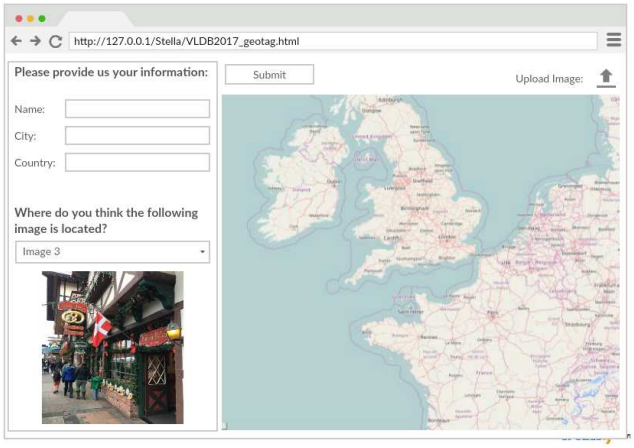


Figure 3: Scenario 2

Near the end of the demonstration scenario, we will show the final location of each image, the resulting confidence, as well as whole the geotagging process that are resulted from the efforts of the conference attendees. To show the geotagging process, we will use the visualization interface that is used in the first scenario (Figure 2). Furthermore, each conference attendee will also be able to check the score of her answer which is calculated based on the distance of her answer to final location of the image.

6. REFERENCES

[1] M. Abchir, I. Truck, and A. Pappa. Dealing with natural language interfaces in a geolocation context. *CoRR*, 2013.

[2] H. Abdelhaq, C. Sengstock, and M. Gertz. Eventweet: Online localized event detection from twitter. *PVLDB*, 2013.

[3] M. Alivand and H. Hochmair. Extracting scenic routes from vgi data sources. In *SIGSPATIAL*, 2013.

[4] M. Graham, S. A. Hale, and D. Gaffney. Where in the world are you? geolocation and language identification in twitter. *The Professional Geographer*, 2014.

[5] C. Hauff. A study on the accuracy of flickr's geotag data. In *SIGIR*, 2013.

[6] H. Samet, J. Sankaranarayanan, M. D. Lieberman, M. D. Adelfio, B. C. Fruin, J. M. Lotkowski, D. Panozzo, J. Sperling, and B. E. Teitler. Reading news with maps by exploiting spatial synonyms. *Communications of the ACM*, 2014.

[7] C. W. Schmidt. Using social media to predict and track disease outbreaks. *Environmental health perspectives*, 2012.

[8] D. Sun, S. Li, W. Zheng, A. Croitoru, A. Stefanidis, and M. Goldberg. Mapping floods due to hurricane sandy using npp viirs and atms data and geotagged flickr imagery. *International Journal of Digital Earth*, 2016.

[9] S. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *Computer graphics and image processing*, 1975.

[10] Web search geolocation. https://www.wired.com/2010/03/location-isnt-just-a-feature-anymore-its-a-platform/.

[11] T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. *arXiv preprint arXiv:1602.05314*, 2016.