Location Extraction from Social Networks with Commodity Software and Online Data

George Valkanas, Dimitrios Gunopulos
Department of Informatics & Telecommunications
University of Athens
Greece
{*gvalk,dg*}*@di.uoa.gr*

Location is prevalent in most applications nowadays, and is considered a first class citizen in social networks. Locational information is of great significance since it can be used to map information from the online back to the physical world, to contextualize information, or to provide localized recommendations through Location-Based Services (LBS), and can be extended to trajectories and itineraries by adding a time-dimension. Despite all that, location extraction in social networks usually relies on GPS-enabled devices, that provide accurate geodetic coordinates. Nevertheless, most users provide general textual information about their surroundings, such as the city they live in, county or state (or equivalents), without using a GPS-enabled device, which is still valuable information for a number of applications. In this paper, we tackle the problem of extracting location information, usually referred to as *geocoding*, from additional user-provided content. Instead of using sophisticated and complex algorithms, which are common in online map services but require heavy development and tuning, we rely on software and data which are available online and publically accessible. We discuss the particularities of geocoding in online social networks and present a simple, lightweight, yet efficient approach for location extraction in such a setting. We finally evaluate our approach experimentally on a large corpus of Twitter users.

I. Introduction

Location is prevalent in most applications nowadays, and is becoming all the more popular. In fact, we can see a clear shift from the “spatial is special” argument made in the ’90s and early 2000 [1] to a general adoption of a “spatial is everywhere” statement. Locational information is highly important since it can be used to map information from the online back to the physical world, contextualize data and provide localized recommendations through Location-Based Services (LBS). Finally, such is its importance that concealing the location of a user is actively researched in privacy preserving data mining [2], [3], [4], [5].

Social networks are no exception, and location in this setting is rapidly gaining ground, giving rise to location-based social networks and location-enabled applications for socially connected entities. The idea is that people are not only socially connected but are also geographically dispersed and their social ties are a result of their current or past location(s). Therefore, it is no wonder that new-age recommendations rely on both social and geographical criteria [6], commonly referred to as “geosocial networking”, as much as they do on demographics.

Despite all that, applications (e.g., FourSquares, Gowalla) and research approaches [7], [8], [9], [10], [11] that rely on user location, utilize GPS-enabled devices, e.g., smartphones, that provide accurate geodetic coordinates. Unfortunately, users that share their location through a GPS-device are currently far less than those who do not, meaning that significant amount of information is lost, unless we can link them to a physical place as well. Figure 1 demonstrates the number of Twitter users, over a 10% unbiased sample of all users, who share their location through a GPS mechanism, during a 2-month period, as opposed to those who did not. Almost half of the users who provide GPS location (green area) have done so through a status update, whereas the other half (grey area) give GPS location in their profile. Regardless, these users make up only 1% of the Twitter users, whereas approximately 60% provide textual information. Even if we explicitly asked for GPS-enabled tweets, the figure practically states that most information on Twitter would remain idle, which is clearly underutilization of information.

On the bright side, most social network users share their

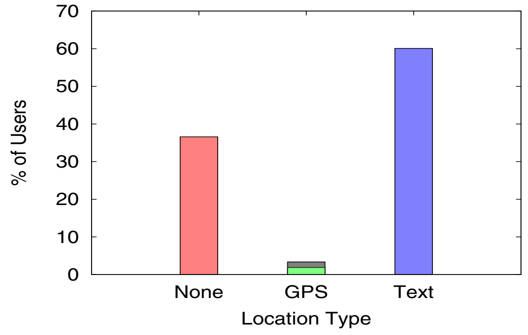

Figure 1. Distribution of users based on how they discolse their location

location publically on their profile in text form, including, for instance, the city, state, and country (or equivalents) they live in. Therefore, they provide a coarse view of their whereabouts in general, usually at city level granularity, or they attach such information to each of their status updates. Being able to pinpoint users at the city level is still important and sufficient for a variety of applications, such as event detection of high impact, spatiotemporal burstiness [12], sentiment analysis or even demographics, which has been historically handled at a higher level. Finally, providing city-level location could be a good indicator of how well a user knows a place, and we could use such users as query-answerers on location-related questions, e.g., sightseeing recommendation, itineraries etc.

The problem of extracting the location of a user in terms of (*latitude, longitude*) given a textual query is known in the literature as *geocoding*. Building a geocoding service is not easy, mostly because of the need for a complete reference database, as well as the heavy development and fine-tuning it entails [13], [14]. These reasons explain well enough why no effort has been performed, up to date, to extract locational information from the users' textual profile input. Moreover, despite the abundance of online geocoders, relying on these services as external resources is impractical and sometimes impossible as a result of their terms of use. Given the hard limits that the services pose on their daily query quota, it would take several months to geocode, e.g., all Twitter users, which are approximately 140 million [1] at the moment.

Motivated by the aforementioned problems, and a need for high volume of localized data, in this paper we tackle the problem of *geocoding* locations, provided by the users as textual information. Unlike existing sophisticated and complex algorithms, which are common in online map services [15], [16], [17] we rely on software and data which are publically available online, making our approach practical and easy to implement. We present a simple, lightweight, yet efficient algorithm to geocode user locations and place them on the map, at the best possible granularity. Finally, our approach is effective and is able to augment the pool of location-mapped users significantly, as we experimentally demonstrate on a large corpus of Twitter users.

To make a simple analogy, our approach ressembles for the geocoding task the well-known and powerful MapReduce [18] programming model. Map–Reduce has seen widespread adoption by alleviating the need for specialized hardware and fallback mechanisms, making it possible to run high-performance processes on commodity hardware. Similarly, our method does not need specialized software or datasets; on the contrary, we rely only on commodity software and data which can be found in online resources. Moreover, our approach is efficient, parallelizable, and runs locally on commodity hardware.

[1]https://business.twitter.com/basics/what-is-twitter/

The rest of the paper is organized as follows. Section II presents the related literature on the subject. Section III provides the specifics and the algorithmic description of our approach, followed by Section IV which includes the experimental section. Finally, Section V concludes the paper and discusses ideas on improvements and directions.

II. Related Work

The main goal of our paper is to return geodetic coordinates ("latitude, longitude"), which are the nearest to a location, presented as a textual query. The process is commonly known as *geocoding* and its common use refers to mapping street address locations to coordinates. Note that effective geocoding in that sense requires complete reference datasets of street names, which are unavailable to the public for all countries. Moreover, existing methods assume that the query location will be well-formed, or will adhere to some structure to guide the search, e.g. comma separation of administrative hierarchies: *Street No, Street, City, State*. Early works on geocoding investigated address tokenization techniques, employing rule-based or Hidden Markov Model [19], [20], because non-standard address formats were used. Note that according to a recent survey [21], the problem still persists in most countries.

These methods, however, used public health records, hence, sufficient information on the location of the patient (city, county, state) was present, and were constrained within a single country. On the contrary, our approach is challenged by high diversity, as social networking sites are used by people around the globe. Moreover, users hardly ever provide a street level location and write in their personal style. Location nicknames (e.g. "Fog City" for San Francisco), abbreviations ("LA" for Los Angeles, or "KCMO" for "Kansas City, Missouri") and (intentional) mispellings (e.g., "Laweezyana") are also customary. Finally, note that city-level granularity is sufficient for our purposes, yet we still need to overcome the other shortcomings.

The most relevant work to ours is [13], used in commercial online map service [17]. There are several technical differences in our approach and theirs, with the most basic being that we do not have access to a complete reference dataset, but rely on online resources as our primary data, making our approach easy to implement. Moreover, we employ lightweight, yet efficient algorithms, unlike the sophisticated algorithms presented in [13], which rely on mature commercial technology [14] and parameter fine-tuning. We stress that, although our goals overlap, our incentive is not to compete with these technologies, but provide simple algorithmic mechanisms to geocode information.

Exploting online resources for geocoding purposes has been researched in the past [22], but with a different goal in mind: to discriminate and accurately identify the location of a street address. The idea stems from the lack of fine-grained information in common online datasets (e.g. TIGER), and the authors counter this problem by using tax and real-estate sites, which provide additional information on the census blocks. Since address level location is scarce in online social networks, this methodology is inapplicable in our case.

Proposals have also been made [23], [24] to use online resources to automatically construct and maintain gazetteers, which are essentially the datasets of reference locations used in geocoding and other applications (e.g., named entity recognition). The advantage of these methods, using interlinked resources which is common in Wikipedia, is that document level linking (usually) corresponds to spatial proximity or relation (e.g., administrative hierarchy). Nevertheless, the goal of building gazetteers is orthogonal to ours, which is to geocode user locational information from short free texts. The more accurate gazetteers are, the better our approach can perform. Given that there are publically available datasets of good quality, and resources which are easier to crawl, we did not consider these methods in our current research.

Similar in spirit are the works in [25], [26] in that they also try to derive a location from user tags. However, their approach differs in that users carefully select tags to describe a photogoraph because that will increase their photos visibility, unlike the social network setting. Moreover, photographs are usually about landmarks, which can be accurately identified and placed on a map. Finally, the Flickr service itself allows users to geocode their photos by selecting a place on a map, which can provide an accurate reference dataset of location-related tags.

III. Location Extraction

A. Problem definition

The problem of geocoding basically means to transform a location of type A, to another –usually equivalent– location of type B, through a meaningful mechanism. In our setting, the goal is to map textual locations, provided as user generated input, to a (*latitude, longitude*) pair or an equivalent identifier (that can be mapped back to these coordinates). The problem can then be defined as follows (tailored for our setting):

Problem 1: Given a user location $\mathcal{L}$, provided as a set of tokens (terms), find a set of geodetic coordinates (i.e., *latitude, longitude*), that can accurately describe $\mathcal{L}$.

Given this definition, our research focuses on the following desiderata:

1 Using available text descriptions from online resources
2 Investigate the accuracy and efficiency of simple techniques

B. Problem setting

Note that the geocoding definition is general enough and several transformation mechanisms could be employed. Moreover, the level of required accuracy is subjective, and usually is application-dependent. For event detection, achieving city level accuracy is sufficient. In that respect, returning a set of geodetic coordinates that describe the city (e.g., the center of the city) is considered enough. We are also highly interested in efficiency, due to the voluminous amount of data that is generated in social networks.

Unfortunately, existing online resources are notoriously known for being incomplete or inaccurate [27], even for the US. To circumvent this problem, we utilize the wealt of information available on the Web and fill in missing gaps. As we are not interested in street level accuracy, the GeoNames dataset [2] is sufficient for our purposes. Despite its richness, the dataset lacks a hierarchy of administrative units (e.g. county, state, etc) for most countries. To overcome this shortcoming, we can crawl online resources, such as the Flickr places dataset, or even mine Wikipedia, which will provide better information. For instance, we can use wikipedia to construct reference lists for abbreviations, synonyms or nicknames of locations (e.g. ”Fog City” or ”Frisco” for ”San Francisco”, 2-lettered US and Canada states abbreviations, etc.)

C. Online geocoding

A straightforward solution one might consider would be to query online geocoders, such as GeoNames server, OpenStreetMaps server, or even online mappings services. However, online mapping services pose a hard limit on the number of the allowed daily query quota. Even if such quotas were not present, crawling “politeness” should still be honored, which would eventually pose an upper bound on the number of submitted queries.

Although such an approach would work well for a few locations (say, a few thousand), it is infeasible for online social networks. Twitter currently claims 140 million users, whereas Facebook has nearly 1 billion active users per month [3]. Even if we take into account that 60% of the users provide their location in text form and assume a high query quota is allowed (Yahoo! allows 50K queries/day), it would take several months to geocode all of them. Finally, online geocoders are accessed through HTTP requests, which is known to be a slow communication protocol, and do not allow batch processing requests.

D. Data Cleaning

Unfortunately, users tend to write in their profile quotes such as *behind you*, *why do you ask*, etc., so as not to disclose their location. Therefore, it is mandatory to clean such data to come up with location information that will be of use.

For instance, users multiplex different encodings, charsets and fonts to make up their location. Though these locations are readable and well-understood by humans, they can not be mathced to an actual location, unless they are transformed to another encoding. Users also tend to surround their text

[2]http://www.geonames.org/
[3]http://newsroom.fb.com/content/default.aspx?NewsAreaId=22

with emoticons or various shapes (e.g. hearts, stars, bars, etc.) to make them fancy. We have manually compiled such lists and intend to make them publically available.

We also have lists to remove japanese, korean and arabic text. A major issue with japanese and korean is that text tokenizers and analyzers –including the one used by Lucene– are not well suited for these languages, which do not use whitespace delimiters. Assuming that an appropriate tokenizer is present, our algorithm is still applicable. Due to this fact, such locations resulted in a high rate of false positives. Given that we are unable to understand these languages, we removed all such locations, using custom built lists from online dictionaries, alphabets and the actual text from the locations.

Finally, given that (custom) data cleaning is a time consuming process, we could make use of machine learning approaches for this purpose. Sentences that do not refer to actual locations may emerge as topics and we could employ topic extraction algorithms (e.g. Mallet [4]). Alternatively, we could use co-occurrence relations or even maximal sets to identify these. We plan to thoroughly investigate these approaches as part of future work.

E. Algorithmic description

Figure 1 presents a detailed algorithmic description of our lightweight geocoder. The algorithm takes as input the reference database (our *gazetteer*) $\mathcal{G}$ and the associated hierarchy of locations $\mathcal{H}$. In our approach a 4 level hierarchy was used: i) suburbs and towns, ii) counties, iii) states and iv) countries. Coarser or more fine-grained hierarchies could be employed, but we see that these levels work well in practice and align well with human intuition. The algorithm also takes the location $\mathcal{L}$ that we wish to geocode.

The entire process consists of 3 phases: *cleaning*, *tokenization* and *searching*. The cleaning process lower cases the characters to simplify subsequent steps. Due to the nature of the social networks datasets, cleaning also means that we need to correct fonts and encodings, and convert them to the one used in $\mathcal{G}$. Once this has been performed, we remove decoratory symbols (e.g., hearts, stars etc) that users insert in their location. Unfortunately, a drawback of using commodity software is that ordinary tokenization will not recognize these characters as text delimiters. The problem that then occurs is that tokens will not match valid locations in $\mathcal{G}$. As a final step, we remove characters which we can not process, e.g., japanese in our case. Note that in a complete system, this should be an optional step.

Once the cleaning process is complete, we tokenize our input location. Simple tokenization is performed, on whitespaces. We then check whether we can combine tokens with one another, into larger token sequences. Intuitively, the idea is to combine tokens which make up valid locations, e.g. "los angeles" instead of separately having "los" and "angeles". During this step, we do not care about the actual locations that can be retrieved, only whether additonal locations can be retrieved. This step creates a set of search tokens to use in the final step.

Finally, we perform the actual location mapping step. For each location from the search tokens, we get a list of id's, which we call *interpretations*. An *interpretation*

Algorithm 1 LightGeocoder

```
Input: Reference database (gazetteer) $\mathcal{G}$, Locations Hierarchy $\mathcal{H}$, Location to geocode $\mathcal{L}$
Output: Set of possible geodetic coordinates $\mathcal{C}$

1:  //Location preprocessing
2:  $cleanLocation \leftarrow lowerCase(\mathcal{L})$
3:  $cleanLocation \leftarrow CorrectFonts(cleanLocation)$
4:  $cleanLocation \leftarrow RemoveDecoration(cleanLocation)$
5:  $cleanLocation \leftarrow RemoveCharacters(cleanLocation)$
6:
7:  //Search term extraction
8:  $tokens \leftarrow tokenize(cleanLocation)$
9:  $searchTokens \leftarrow \emptyset$
10: $newToken \leftarrow tokens[0]$
11: for ( $i \leftarrow 1; i! = tokens.size(); i{+}{+}$ ) do
12:     $search \leftarrow newToken + "\ " + tokens[i]$
13:     if ( $\mathcal{G}.getResult(search).size()$ != 0 ) then
14:         $newToken \leftarrow search$
15:     else
16:         $searchTokens \leftarrow searchTokens \cup newToken$
17:         $newToken \leftarrow tokens[i]$
18:
19: //Location Matching
20: $matches \leftarrow \emptyset$
21: for ( $i \leftarrow 0; i! = searchTokens.size(); i{+}{+}$ ) do
22:     $search \leftarrow searchTokens[i]$
23:     $interpr \leftarrow \mathcal{G}.getResult(search)$
24:     if ( $matches.isEmpty()$ != 0 ) then
25:         $matches \leftarrow interpr$
26:         continue
27:     $newMatches \leftarrow matches$
28:     for ( $j \leftarrow 0; j! = interpr.size(); j{+}{+}$ ) do
29:         $ancestor \leftarrow \mathcal{H}.search(interpr[j], matches)$
30:         if ($ancestor! = null$) then
31:             $newMatches \leftarrow newMatches \setminus ancestor$
32:             $newMatches \leftarrow newMatches \cup interpr[j]$
33:     $matches \leftarrow newMatches$
34:
35: $\mathcal{C} \leftarrow \emptyset$
36: for ( $i \leftarrow 0; i! = matches.size(); j{+}{+}$ ) do
37:     $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{G}.coordiantes(matches[i])$
38: return $\mathcal{C}$
```

[4]http://mallet.cs.umass.edu/

is simply a possible location against which a token can be matched, without using other contextual information. For each new interpretation (lines 28-32), we check in the hierarchy $\mathcal{H}$ whether there is a child-parent relation with another interpretation that we obtained for previous search tokens. If we identify such a child-parent relation, we remove the parent id and insert the child's. This way, we increase the level of granularity for the location we have been given. Note that we maintain locations for which we can eventually establish a connection in the hierarchy. Once this process is complete, we query our database $\mathcal{G}$ for the actual coordinates of the locations that we have extracted, which we then return.

As a final note, the (worst case) algorithmic complexity is $O(\prod inter(token))$, where $inter(token)$ is the number of interpretations that each token may have. However, tokens are usually few (at most 7), and each one is matched to few locations. As we demonstrate in our experimental evaluation, the actual running time of the entire process is minimal, keeping the approach efficient for our purposes.

IV. Experiments

A. Experimental Setup

We have crawled Twitter through the Gardenhose between the start of April and end of May 2012, and have obtained a dataset of a little less than 300 million tweets. The tweets have been posted by 29 million unique users (identified by their id). After filtering users by language (english, french, german, greek, spanish) and timezone (USA and european timezones), we get nearly 11 million unique users. Note that several of them are in fact non-english speaking, however, the default language option is english. Since Twitter does not currently support all languages –which it uses internally for translation–, most users leave the default option. This is not a major issue as we can filter out such users further down the processing chain, based on the content of their tweets, using either lexicon-based or machine learning approaches, e.g. [28].

Out of the 11 million users, we have extracted approximately 2.5 million locations, which means that there is an average of 4 users per 1 location. If we allow each user to maintain a single location, then we obtain 2.35 million locations, meaning that during this 2-month period, about 150K users changed their profile location. This leads to the following conclusions:

* Users will update their profile location if they are inclined to do so.
* For the most part, for short periods of time (e.g., spanning a few months), the general location of a user remains unchanged. By "general" we mean a broad area where they have mostly been during that period.
* Since users do not change their profile location that often, geocoding it and storing it for future retrieval is meaningful and can be easily performed. Note that in case of users who have enabled a geo-tagging service, their location is available. Nevertheless, these services use a well-formed naming convention, making the geocoding problem much easier.

After cleaning the locations, using the process that we have described in the previous paragraph, we derive a dataset of 1.8 million locations. Within this dataset, the GPS-enabled locations are no more than 100K. Table I summarizes the basic properties of our dataset.

Table I
Location dataset characteristics

Variable	**Value**
#Initial users	29 million
#Users after filtering	11 million
#Uncleaned (unique) locations	2.5 million
#Cleaned (unique) locations	1.8 million
Average token count / location	3.147
Full gazetteer size	3,490,337 entries
Small gazetteer size	1,042,742 entries

We also created two reference databases (i.e. gazetteers), both of which include all of the data that we crawled from the Flickr Places website. Their difference lies in the entries used from the GeoNames dataset, for the countries we want. The first one contains all of the GeoName entries, and we denote this one as *Full*. The second one uses only entries that have been marked as "administrative" (e.g., a county) or "populated area" (e.g., village). We denote the latter dataset as *Small*. Each GeoName entry has been assigned to the closest location of similar administrative type from the Flickr dataset, as we have described in the previous sections, to be able to efficiently exploit the hierarchy. Finally, we have indexed each reference database using Lucene [5], to allow for IR-style retrieval of locations.

B. Efficiency Evaluation

In this subsection, we evaluate our system in terms of efficiency. The geocoding process is written in Java 1.6 and run on a single node, with Quad Core @3GHz. We measure the time taken to run the LightGeocoder algorithm for each user and report the cumulative time required for all 1.8 million unique, cleaned locations. Therefore, the reported values show how much time is required for the geocoding process in a real system implementation.

Figure 2 shows the efficiency of our geocoding scheme, with each of the two gazetteers. Clearly, when the *Full* gazeetteer is used the runtime increases, because each token has more possible interpretations and the index contains more entries to search against. On the other hand, the *Small* gazeetteer is more lightweight and requires less time to come

[5]http://lucene.apache.org/core/

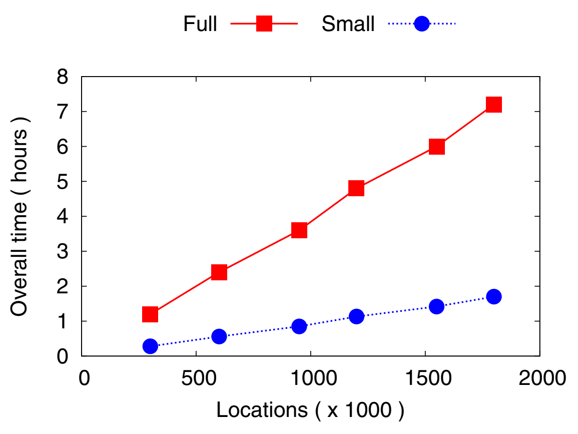

Figure 2. Efficiency of geocoding using the Full and the Small gazetteers

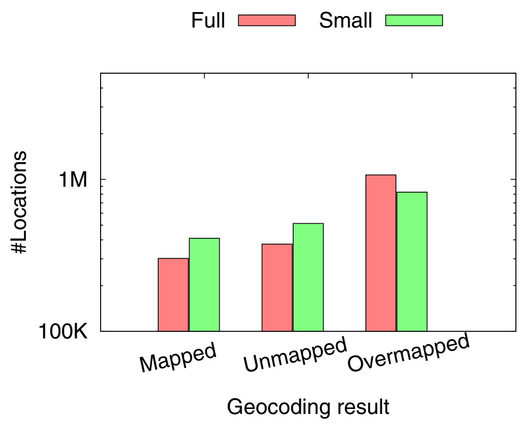

Figure 3. Number of mapped, unmapped and overmapped locations by gazetteer usage

up with the set of possible locations. Although there is a 3:1 ratio in the gazetteer sizes, the processing times differ by a factor of 10; this is can be explained as the hierarchy consists of 4 levels (including countries).

Not surprisingly, both approaches are linear to the number of locations mapped. Therefore, we can cut down the total elapsed time by using multi-threaded solutions or by distributing the workload among more computers. Regardless, the average (per location) geocoding runtime is 3.4 ms when using the *Small* database, whereas it is 14.4ms with the *Full* database, which still allows us to do the geocoding in real time. Finally, the cleaning process takes a negligible amount of time (in the order of 10^{-5} sec), and does not influence the overall efficiency, as it is linear in the text's length and performed through lookups.

C. Effectiveness Evaluation

A basic goal of our research was also to investigate the effectiveness using simple techniques and publically available data. To measure our technique's effectiveness, we performed a series of experiments. First off, and in order to simplify the evaluation process, we assume that if our geocoder returns more than 5 possible matches, then we can not map the corresponding location at the granularity we desire. However, we discriminate between "*overmapping*" a location and not mapping it at all. Effectively, we can see the number of locations that have been mapped, the ones that have been "*overmapped*" and the remaining ones that have not been mapped. Figure 3 shows the number of locations (in log scale) that fall inside each of these categories when either of the gazetteers is considered.

The interesting outcome of this experiment is that, contrary to what one might expect, the *Small* gazetter maps more locations than the *Full* gazetteer, especially when it leaves more locations unmatched! Apparently, the large number of indexed entities present in the *Full* gazetteer introduce more disambiguation and make it harder for the algorithm to select a small set of possible matches. This leads us to the conclusion that it is not only important to clean the incoming locations, but we should also pay special care to the reference dataset.

Mapping (or not) more or fewer locations may be statistically useful, but does not realy hint us on how well our system performs. For this reason, we have conducted an *oracle* experiment, querying a service which is particularly suited to the geocoding task. Specifically, we queried the Yahoo! maps geocoding service with 10K locations, randomly sampled from the 1,8M locations that we experimented with. The service returns its response in XML format, with a ranked list of potential matches. If the location could not be matched, an empty ranked list is returned.

We used the default parameters which return at most the top-10 possible locations, along with a GPS location and a small textual description. Each result is also accompanied by a quality element of integer type, which is essentially the granularity (or precision) of the resulting location. Higher values indicate greater precision. For instance, quality values greater than 80 correspond to points, whereas values between 31 and 40 are town level areas.

We initialy present a confusion matrix, which measures 4 values:

- *True positives*: the number of textual description identified as a location by both approaches
- *True negatives*: the number of textual information that have been correctly identified as **not** being locations
- *False positives*: the number of textual information which (our approach) identifed as being possible locations, whereas Yahoo! maps did not consider them as such.
- *False negatives*: the number of locations which we failed to identify as being a valid location, whereas Yahoo! maps returned a ranked result set.

Table II contains these four measures for both the reference databases. The matrix is read as a combination of its row and column, for the corresponding gazetteer. For instance, the upper right corner reveals the *true* (vertical) *positive* (row) for the *Simple* gazetteer. Two important things to mention are: i) *overmapped* locations are handled as *unmapped*, on the basis that we can not discriminate between them easily with our current approach, and ii) the *true positives* metric only measures whether both techniques identified a description as being a location or not. It does not consider, whether the same location has been indeed selected, which we discuss followingly.

Table II
CONFUSION MATRIX FOR THE "ORACLE" EXPERIMENT

	Full		**Small**	
	True	*False*	*True*	*False*
Positive	1699	114	2363	75
Negative	656	7551	695	6887

For the **True** columns, higher values are preferred, whereas for the **False** ones, lower values are preferred. Once again, we derive the conclusion that "simpler is better". The small gazetteer performs better than the full gazetteer in all four occasions. In fact, if we carefully observe the matrix, it appears that when we move from the small to the full gazetteer, there is a transfer of true positives to false negatives (a result of "overmapping" locations), and a similar transfer of true negatives to false positives. Clearly, this artifact has been introduced by the GeoNames dataset, since everything else has remained unchanged.

We now turn our attention to a more fine-grained notion of effectiveness, regarding the true positives which we identified with each approach. Our interest is to see if the locations extracted by our geocoder fall within the top most locations returned by the Yahoo! maps. To this end, we measure the percentage of documents retrieved by our approach that can be matched to one of the top-n results by Yahoo!. For instance, if the top-1 result is present in our returned list, we assume that we have a match at position 1. Because we currently avoid ranking, this essentially means that we have the prospect of returning one of our results as a top-1 value (and stop at that point).

We select up to the top-5 results, returned by Yahoo! maps, which have already been provided as a ranked list. For each result, provided that its quality is up to a city level ($quality > 30$), we extract the closest city location from our crawled Flickr dataset, which is the basis for the hierarchy. Followingly, we check if at least one of our geocoder's proposed locations is within Yahoo! map's top results.

Figure 4 demonstrates the cumulative measure we discussed in the previous paragraph. We observe that the simple gazetteer did not only achieve a high true positive ratio (see Table II), but it has been able to correctly identify the correct location at a percentage of 36% with simple techniques. It

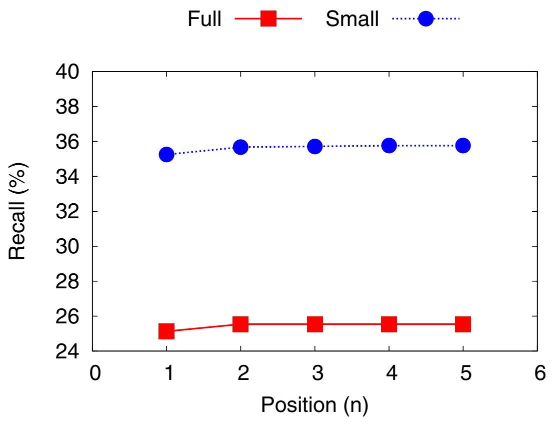

Figure 4. Effectiveness of our approach using the two gazetteers

also performs better than the *full* gazetteer technique, by at least 10%. In actual number, the simple gazetteer matched correctly around 845 locations, as opposed to a mere 434 that the full one did.

As a final note, it is important to mention that, despite the many "(mis)classifications" (false positives, false negatives), even with the full gazetteer it appears that we can increase our geo-tagged users data pool by roughly 5% of present **locations**. This translates to several users especially, if we consider the average 4:1 user-location ratio. Clearly, our gold-standard data is quite small, and generalizations should be cautioned, but it still provides good evidence that profile location is useful and can be extracted with lightweight approaches, such as the one presented in this paper.

V. CONCLUSION

Location is of paramount importance in today's applications, and social networks are no exception. Several crucial and practical applications rely on locating users, but do so using GPS-enabled devices. Social network users tend to provide a coarser view of their whereabouts, e.g., the city where they live, which is still sufficient information for numerous applications and demographics.

In this paper we addressed the problem of *geocoding* social network user locations, at a city / town granularity level. Instead of implementing sophisticated algorithms, which require a complete reference databases, advanced data structures and parameter fine-tuning, we used resources which are publically available online, with respect to both data and software. Therefore, our approach is easy to implement and extend. Finally, we demonstrated experimentally the efficiency and efficacy of our approach, contrasting the results we obtain against Yahoo!'s online geocoder. We generally observe a significant improvement in the number of user locations we can extract and utilize, as opposed to GPS-only locations, with minimal computational overhead.

Acknowledgements This research has been supported by an “Heraclitus II” scholarship, co-financed by the E.U. and Greek national funds, by EU-funded projects DISFER, INSIGHT, MODAP, and the MMD project, funded by GSRT program “ARISTEIA”.

References

[1] L. Anselin, “What is special about spatial data? alternative perspectives on spatial data analysis,” in *Symposium on Spatial Statistics, Past, Present and Future*, 1989, pp. 63–77.

[2] R. Agrawal and R. Srikant, “Privacy-preserving data mining,” in *Proc. SIGMOD ’00*, 2000, pp. 439–450.

[3] A. Gkoulalas-Divanis, P. Kalnis, and V. S. Verykios, “Providing k-anonymity in location based services,” *SIGKDD Explor. Newsl.*, vol. 12, no. 1, pp. 3–10, Nov. 2010.

[4] T. Xu and Y. Cai, “Location anonymity in continuous location-based services,” in *Proc. GIS ’07*, 2007, pp. 1–8.

[5] P. Golle and K. Partridge, “On the anonymity of home/work location pairs,” in *Proc. Pervasive ’09*, 2009, pp. 390–397.

[6] P. Symeonidis, A. Papadimitriou, Y. Manolopoulos, P. Senkul, and I. H. Toroslu, “Geo-social recommendations based on incremental tensor reduction and local path traversal,” in *Proc. GIS-LBSN*, 2011, pp. 89–96.

[7] J. Eisenstein, B. O’Connor, N. A. Smith, and E. P. Xing, “A latent variable model for geographic lexical variation,” in *Proc. EMNLP ’10*, 2010, pp. 1277–1287.

[8] L. Ferrari, A. Rosi, M. Mamei, and F. Zambonelli, “Extracting urban patterns from location-based social networks,” in *Proc. LBSN ’11*, 2011, pp. 9–16.

[9] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, “Discovering geographical topics in the twitter stream,” in *Proc. WWW ’12*, 2012, pp. 769–778.

[10] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in *Proc. WWW ’10*, 2010, pp. 851–860.

[11] S. Wakamiya, R. Lee, and K. Sumiya, “Crowd-based urban characterization: extracting crowd behavioral patterns in urban areas from twitter,” in *Proc. LBSN ’11*, 2011, pp. 77–84.

[12] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras, “On the spatiotemporal burstiness of terms,” *PVLDB*, vol. 5, no. 9, pp. 836–847, 2012.

[13] V. Sengar, T. Joshi, J. Joy, S. Prakash, and K. Toyama, “Robust location search from text queries,” in *Proc. GIS ’07*, 2007, pp. 1–8.

[14] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, “Robust and efficient fuzzy match for online data cleaning,” in *Proc. SIGMOD ’03*, 2003, pp. 313–324.

[15] http://maps.yahoo.com/.

[16] http://maps.google.com/.

[17] http://www.bing.com/maps/.

[18] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[19] T. Churches, P. Christen, K. Lim, and J. Zhu, “Preparation of name and address data for record linkage using hidden markov models,” *BMC Medical Informatics and Decision Making*, vol. 2, no. 1, 2002.

[20] J. Murphy and D. R. Armitage, “Merging the modelled and working address database: A question of dynamics and data quality.” http://limerickcity.ie/IT/GIS-GeographicalInformationSystems/AssociatedDocuments/NewsletterFile,3692,en.pdf, Accessed on Sept. 2, 2012.

[21] D. Goldberg, J. Wilson, and C. Knoblock, “From text to geographic coordinates: the current state of geocoding,” *URISA Journal*, vol. 19, no. 1, pp. 33–47, 2007.

[22] R. Bakshi, C. A. Knoblock, and S. Thakkar, “Exploiting online sources to accurately geocode addresses,” in *Proc. GIS ’04*, 2004, pp. 194–203.

[23] Z. Zhang and J. Iria, “A novel approach to automatic gazetteer generation using wikipedia,” in *Proc. People’s Web ’09*, 2009, pp. 1–9.

[24] A. Toral and R. Munoz, “A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia,” *EACL*, 2006.

[25] D. J. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg, “Mapping the world’s photos,” in *Proc. WWW ’09*, 2009, pp. 761–770.

[26] P. Serdyukov, V. Murdock, and R. van Zwol, “Placing flickr photos on a map,” in *Proc. SIGIR ’09*, 2009, pp. 484–491.

[27] J. H. Ratcliffe, “On the accuracy of tiger-type geocoded address data in relation to cadastral and census areal units,” *International Journal of Geographical Information Science*, vol. 15, no. 5, pp. 473–485, 2001.

[28] S. Sagiroglu, U. Yavanoglu, and E. N. Guven, “Web based machine learning for language identification and translation,” in *Proc. ICMLA*, 2007, pp. 280–285.