Qualitative Geocoding of Persistent Web Pages

Albert Angel*
Dept. of Computer Science
University of Toronto
Canada
(+1) 416-946-8878
albert@cs.toronto.edu

Chara Lontou
Dept.of Electrial Engineering and Computer Science
National Technical University of Athens, Greece
clontou@dblab.ece.ntua.gr

Dieter Pfoser
Alexandros Efentakis
RA Computer Technology Institute
University Campus Patras
26500 Rion, Greece
+30 210 6930 700
{pfoser|efedakis}@cti.gr

ABSTRACT
Information and specifically Web pages may be organized, indexed, searched, and navigated using various metadata aspects, such as keywords, categories (themes), and also space. While categories and keywords are up for interpretation, space represents an unambiguous aspect to structure information. The basic problem of providing spatial references to content is solved by geocoding; a task that relates identifiers in texts to geographic co-ordinates. This work presents a methodology for the semi-automatic geocoding of *persistent* Web pages in the form of collaborative human intervention to improve on automatic geocoding results. While focusing on the Greek language and related Web pages, the developed techniques are universally applicable. The specific contributions of this work are (i) automatic geocoding algorithms for phone numbers, addresses and place name identifiers and (ii) a Web browser extension providing a map-based interface for manual geocoding and updating the automatically generated results. With the geocoding of a Web page being stored as respective annotations in a central repository, this overall mechanism is especially suited for persistent Web pages such as Wikipedia. To illustrate the applicability and usefulness of the overall approach, specific geocoding examples of Greek Web pages are presented.

Categories and Subject Descriptors
H.2.8 [Database Applications] – Data Mining

General Terms
Algorithms.

Keywords
Multilanguage content, digital libraries, indexing, multilingual metadata, spatiotemporal databases

1. INTRODUCTION
"The map is the new search interface. Geography is another way to organize information. As human beings, we inherently understand geography." [1] In turn maps can become a user interfaces to many things. Geographic information, be it maps or 3D virtual worlds, are believed to be the future way for people to socialize, shop, and share information. In the foreseeable future, the map will become the interface of choice for the internet [29].

All of this works, however, only if information on the web is indexed geographically, i.e., if documents, paragraphs or key phrases thereof are annotated with the location information to which they refer.

Studies have shown that up to 10% of all Web pages contain references such as zip codes, complete address information and phone numbers that can be directly use to assign location information to the page [22]. Further, it is estimated that 60-80 percent of web pages contain overall geographically relevant information that can be used to geo-tag them [29]. The basic problem is to find such identifiers in text (geoparsing) and then to relate them to location information (geocoding).

This work presents a methodology for *the semi-automatic geocoding of persistent Web pages*, i.e., relating identifiers in texts to geographic co-ordinates using a combined automatic and human-centered approach. Specifically, we will focus on Greek Web pages and related geo information. The methods however are universally applicable. Automatic geoparsing and geocoding algorithms are successfully applied to identify phone numbers and addresses. When more generic geo identifiers are involved, automatic algorithms produce a significant number of false positives (Venizelos as a person) and false negatives (Venizelos as the name of Athens international airport). This work *advocates human intervention to improve on automatic geocoding results* (in the spirit of [9]) and develops therefore a Web browser extension that (i) allows for the manual geocoding of text portions and (ii) the updating, including deletion of automatically generated results. This proposed approach is especially helpful for *persistent* Web pages such as Wikipedia, i.e., pages that have a certain value to the community, are well cared for and change rather slowly. Here, geocoding can become a regular part of Web page authoring!

Location information extracted from a persistent Web page is stored in a central repository; for every page, identified by its

ACM GIS '08, November 5-7, 2008. Irvine, CA, USA
(c) 2008 ACM ISBN 978-1-60558-323-5/08/11...$5.00.

* Work done while at National Technical University of Athens, Greece.

[1] Quoting John Hanke, Director Google Earth and Maps.

URL, the repository maintains the geocoded text portions, in terms of their position on the page, the respective geographic co-ordinates, as well as a timestamp, indicating the version of the Web page. The geocoded portions of a Web page are highlighted, and their geographic extent displayed on a map interface (powered by Google Maps [16]); clicking on a text portion shows its position on the map. The respective functionality is accessible through a Firefox *browser extension.* This work also includes a set of use cases that illustrate the applicability and usefulness of the overall approach.

Related work in the area of geocoding is manifold, both in terms of research and commercial applications.

Geocoding Web pages can be a means for indexing content, as in [18]. In addition, the spatial extent of the geocoding is used to reason about the importance of the content, e.g., a web page about a small village in Greece may interest less people than one about the Balkans. Web-a-Where [1] is another system for geocoding Web pages. It assigns to each page a geographic *focus* — a locality that the page discusses as a whole. The tagging process targets large collections of Web pages to facilitate a variety of location-based applications and data analyses. Their experiments show that 80% of individual name place occurrences can be tagged correctly and the correct focus of a page can be established 91% of the time. One of the first works on geocoding [23] describes a navigational tool for browsing web resources by geographic proximity as an alternative means for Web navigation. The approach presented in [4] proposes the use of the Web's geographic information to populate address databases, i.e., parse Web pages for useful address information and populate an address database with the available information. A method for calculating the geographic breadth of a Web page is given in [10]. A geographic search tool is used in the context of personalized search, where user's position is an element of her profile. The proximity of the position of the Web page in relation to the user's position is a criterion for the ranking of the page in the search results.

In the realm of geocoding, a range of related commcercial products exist. Google has developed the web service Google Local (integrated now in Google Maps) [16]. This service offers a search which is based on a combines geographic and keyword-based search (e.g., Pizza in Athens). A similar service is Yahoo Yellow Pages [30]. MetaCarta [22] provides tools and services that geoparse and geocode text content using natural language processes and highly refined geodata. The results can be used for geographic search on the Web, in GIS applications, for categorizing documents, etc. Beyond those service is the GazDB [2] that enables the efficient production of customizable gazetteers. The GazDB separates names from features while storing the relationships between them. Geographic names are stored in a variety of resolutions to allow for internationalization and for multiplicity of naming. Geographic features are categorized along several axes to facilitate selection and filtering. The purpose of the MetaCarta GazDB is to provide both a place and supporting mechanisms for storing, maintaining, and exporting everything we know about our collection of geographic entities. A converter for RSS to GeoRSS [12] is available as an open source service that finds geographic identifiers in RSS feeds using natural language processes (for information extraction) and machine learning processes (for updating its results as it interacts with the users). It is mainly applicable to news articles, presenting a global map with the event places marked and finally offering the ability to search or read the events using geographic criteria.

What is common to all existing work is that it typically advocates an ad-hoc automatic geocoding approach with little or no quality control besides the established accuracy of the algorithm. This work goes beyond that in that we advocate automatic geocoding, user/community-based control and a public repository to store the geocoding (cf. [9]).

The outline of the paper is as follows. Section 2 discusses the employed automatic geocoding approach customized for Greek Web pages. Section 3 introduces the semi-automatic, collaborative approach for geocoding persistent Web pages. Section 4 gives an experimental evaluation. Finally Section 5 presents conclusions and directions for future work.

2. AUTOMATIC GEOCODING

Geocoding refers to (i) identifying text portions that might have a spatial aspect (geoparsing) and (ii) linking this text to location information, typically coordinates (geocoding). Ideally, this overall process can be automated as discussed in the following and illustrated in Figure 1.

This section presents the specific *approach that was developed for the automatic geocoding of Greek Web pages*. Although representing a standalone solution, Section 3 will show how this solution can be wrapped in browser extension to allow for a semi-automatic geocoding approach for Web pages.

The overall automatic geocoding framework includes (i) an HTML Parser for removing tag information and detecting the character encoding of the page, (ii) a module for the standardization, composition and presentation of the geocoding, (iii) a module for approximate string matching and a (iv) a database containing geo data needed for relating, e.g., address information to geographic co-ordinates.

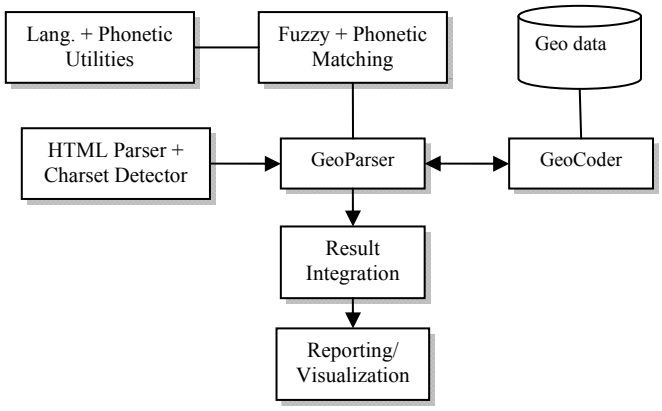


Figure 1: Automatic geocoding system architecture

The following sections will discuss each module in more detail.

2.1 HTML Parser

One of the main issues that arise during the parsing of Web pages is that most of the times the HTML code is malformed and does not usually conform to the stricter XHTML standard. Most programming languages (including Java that was used during the development of our system) do not directly support parsing of non valid XML documents. Since regular HTML code is not valid

XML code, TagSoup [7], an external Java library capable of parsing typical HTML code, was used to clean up the content.

2.2 Geoparsing

In our testing system, we used a combination of cascaded regular grammar transducers and approximate String searching and matching algorithms (those algorithms will be discussed in detail later on this paper). Every regular grammar level is based on the results of the previous grammar level applied and traces information on a higher abstraction level. In our earliest lower level, our bare character input is converted into words, postal codes, spaces and candidate Placenames. In the highest grammar level, our postal codes and placenames (created from previous grammar levels) are converted into valid addresses. The cascaded regular grammar level parsing implementation was realized using JFlex [19] (the free lexical analyzer generator for Java). The approximate String searching and matching algorithms were used for comparing words residing in the test Web pages, with those inside our database index entries.

The traced geographic information is then normalized and standardized. For example the raw address: Agiou Sosti 3, Chalandri, 23135 will be normalized as: Street: Agiou Sosti, Number: 3, Postal Code: 23135, Area: Chalandri. This is a necessary process for the geocoding phase that follows.

Although previous research approaches tried to exploit the hyperlink information of the parsed Web pages, we decided after extensive testing, that the geographic information that was retrieved through hyperlinks utilization, was rather minimal compared to the effort required to obtain it.

2.2.1 Brill matching for place names

In order to identify all potential Placenames inside a Web page, we used a concept proposed by Brill [5], and its implementation for Greek texts [25], in which, if a group of words is a potential placename match, each of its subgroup of words will also be considered as a potential match. Consequently, if we have a Greek phrase such as "Ατρόμητος η ποδοσφαιρική ομάδα του Βόλου, Μαγνησίας" (meaning "Atromitos the football team of Volos, Magnisia"), we will have to search for matches of the following words and phrases: "Βόλου", "Μαγνησίας", "Βόλου, Μαγνησίας", which are all valid placenames and even search for the word "Ατρόμητος", for which we will not find a match, since it is not a placename.

This extra matching process does not slow down our implementation significantly, since the average Web page usually includes a limited number of (usually) scattered Placenames.

2.3 Geocoding

The geocoding module takes the text portions identified by the geoparser and relates them to geographic data and coordinate information, typically stored in a data repository. The geographic data consists of place names, addresses, telephone numbers, etc., and links these to a respective geographic location by means of coordinates.

Relating phrases to geographic data required the development of approximate string matching and searching algorithms. As we will see in Section 2.4 and 2.5, these algorithms will return with each approximate match a confidence factor indicating if (and to what degree) the phrase represents in fact a geographic location. The geocoding will be in the typical case a simple coordinate pair, indicating a point location, but can also be a Minimum Bounding Rectangle (MBR) of the geographic limits the phrase in question describes.

With respect to the Greek language aspect in our work, one of the most important aspects was the development and integration of approximate string searching and matching algorithms. Approximate String searching and matching is used, both, during the geoparsing and geocoding phases, since focusing on finding an exact match between the phrases found on a Web page and the geo data would only produce a subset of geocoded phrases. In addition, these techniques have been used to improve the overall geocoding speed (filter step in candidate phrase retrieval).

The geo data that is used in the process is stored by means of a DBMS. As all data is retrieved based on alphanumeric queries, and there was no issue with respect to optimizing spatial queries, the popular MySQL DBMS is utilized. However, our algorithms work with any other DBMS, since it does not rely on any of MySQL's specific features.

The following sections introduce first approximate string matching and search and subsequently survey the geo data that was used in the process.

2.4 Approximate String Matching

One of the problems we had to address was how to effectively compare and match two Greek phrases.

2.4.1 Pre-processing of phrases

Before the actual comparison of the two Strings, both Strings phrases should be preprocessed, standardized and parsed. By using external lexicons, relative to the domain, we can rectify minor spelling mistakes and ensure that special keywords with high frequency of appearance are converted into their normalized form prior to the comparison. For example, the Greek word "οδός" means "street". Any other grammatical form or abbreviation of the specific word, like "οδ.", "οδού" will all be converted to the normalized form "οδός". We also use an alternative form for each of the String phrases we are about to compare that does not include any of those high frequency words. This is a necessary step that greatly improves the number of matches, since those words do not add significant information to the Strings in question. Pre-processing of the two Strings prior to comparison is a method that is recommended and considered effective by other researchers as well (cf. [21] and [27]).

After initial pre-processing, both Greek String phrases are converted into their intermediate phonetic equivalents based on a custom phonetic alphabet for the Greek language. In that sense the Greek word "Βενιζέλου" will be converted to its phonetic equivalent "venizelu". This was a realistic hypothesis after observing that (in most cases) spelling mistakes for a specific word do not significantly alter the word's pronunciation. There also very few Greek words (especially in the Geographic domain) with different meanings and yet same pronunciation. The phonetic conversion was based on Greek pronunciation rules along with some custom heuristics.

The Greek phonetic alphabet we used, is comprised of 23 phonetic symbols (cf. [13] and [28]). We define "phonetic

distances" between the various phonetic symbols, based on their phonetic pronunciation grouping. To this phonetic alphabet we also added the character "/", which means the presence of at least one more character, e.g., the Greek phrase "Αιτ/νία" will be matched with the phrase "Αιτωλοκαρνανία". The addition of the character "/" proved to be quite effective, since it allowed shortened alternative forms for a specific word commonly used in Greek geographic names.

2.4.2 Modified Levenshtein distance

After the initial preprocessing of both Greek String phrases and their conversion according to the Greek phonetic alphabet, we compare their phonetic equivalents according to their Levenshtein distance. Originally, Damerau and Levenshtein proposed that the distance between two words is defined as the number of characters we must insert, delete, or modify in order to get an exact match. We modified this classic algorithm for calculating the Levenshtein distance [15], in order to exploit Greek pronunciation similarities (e.g., the Greek word "Μάνου" is similar to "Νάνου") and the addition of the "/" character. Experimental results showed that those minor additions do not modify the algorithm's complexity.

2.4.3 Threshold comparison

The calculated Levenshtein distance will be compared to a threshold distance that depends on the two phrases' length. The threshold distance was calculated using heuristics such as comparing significant number of words with similar pronunciation and best threshold distance based on the F-measure[2]. If the calculated Levenshtein distance is smaller than the threshold distance, then the two phrases are considered similar, with an approximation factor relative to their calculated Levenshtein distance.

2.5 Approximate String Searching

To geocode a phrase, an approximate string look-up algorithm is used to compare this phrase with our database entries; find a potential approximate match and then geocode the phrase, using the geographic location information.

The approach entails translating the phrase (we are looking for) into a simplified hash key, searching the database for entries that have the same hash key, and then performing an approximate String comparison between the initial phrase and the phrases found in our database.

Our hash key was generated using our custom version of Metaphone [20] based on the custom Greek phonetic alphabet described in the previous section. According to this method, each sequence of identical characters is replaced by only one character, the Greek character "ς" found only at words' end is dropped completely and so are all vowels (except the one found at the beginning of the word). Thus, the Greek word "Βενιζέλου" will be converted to its phonetic equivalent "venizelu" and the resulting hash key will be "vmzl". We also chose to preserve the "/" character, during the hash building phase. So, even when comparing hash keys, we may have "compatible" and yet not exact matches. For example, the hash key for "Αιτωλοκαρνανία" will be "edlgm" and the hash key for "Αιτ/νία" will be "ed/m". These two hash keys are not exact matches, but are still considered compatible.

We developed an Approximate String searching algorithm, which searches for potential matches between a phrase inside the Web page and the entries residing in the database based on ideas proposed in [14] and [17]. Searches in the database table are optimized by using phonetic indexes.

2.5.1 Phonetic Index

For the placenames stored in our database, each record is uniquely identified by a single numeric field placenameId (Primary key), which will be the result returned by our approximate String look up algorithm in a case of a potential match. For each such table, a new phonetic index table needs to be built (this process has to be done once per table).

The phonetic index is created from the original placenames according to the following procedure. Each placename phrase in the original table is divided into single words. Each of these words will be a separate record in the phonetic index table. Each record in the phonetic index table will also include the following attributes:

- The *PlacenameId* of the phrase that this word belongs to in the original placename table.
- The *Sequence index* of the word (is this the second or the third word in the original placename phrase?) inside the original placename phrase.
- The *phonetic hash key* of the word. The word is initially converted to its phonetic equivalent to create its phonetic hash key
- The actual *word*
- The *length of word* in characters
- An *isAbbreviation* Boolean field (that shows if the word includes the "/" character.

In order to accelerate searches in the phonetic index table, a B-tree index was used for the following attributes set, {isAbbreviation, Phonetic hash key, PlacenameId, Sequence index of word}.

2.5.2 Approximate String Searching Algorithm

When we search a phrase identified on the Web page against our database entries, we apply the following methodology:

First, split the original phrase found on the Web page into single words.

Second, in a first filtering step, create a new temporary table with all the records of the phonetic index table, where the phonetic hash key for each entry matches the phonetic hash key of at least one of the words traced in the original phrase.

Third, in a second filtering step, drop from the temporary table all records that diverge significantly from the original phrase in terms of total number of words and length of individual words. Here, consider the following example. Let us assume that the Web page includes the phrase "Λεωφόρος Ελευθερίου Βενιζέλου". In the first step, the record "Πλατεία Βενιζέλου Σοφοκλή" contained in our placename table is a potential match, since they share the

[2] F-measure is a weighted precision and recall average.

word “Βενιζέλου”. In order to obtain a match, two words must be removed ("Πλατεία","Σοφοκλή") and, furthermore, two more words must be added ("Λεωφόρος","Βενιζέλου"). In the second filtering step, the algorithm will subsequently drop "Πλατεία Βενιζέλου Σοφοκλή" as a potential match and all records that correspond to its PlacenameId will be dropped from the temporary database table.

After those steps, the database returns a limited number of records that match the original phrase in terms of phonetic hash key similarity (all String comparisons up to now were done at the phonetic hash key level). These two filtering steps can be performed efficiently, since they are supported by the indexes and did not utilize the sequence index of words inside the original phrase.

Fourth, a refinement phase compares word for word, all words from the original phrase, with each word located inside each potential match. This comparison is performed in memory and uses the modified phonetic Levenshtein distance that was described earlier. If during this comparison, a potential match is found to differ significantly from the original phrase, it is dropped. After finishing the word to word comparison, then and only then do we take into account (with the use of an appropriate weight factor) the sequence of words in the original phrase. Although this exhaustive search process is significantly slower than the first two filtering phases, it is still quite fast, because after the first two phases we only have a limited number of potential matches (in most cases 1-5 and very rarely up to 15 potential matches).

Finally, the remaining PlacenameIds are returned to the user, in ascending order of the Levenshtein distance calculated from all previous phases.

2.6 Geo Data

Although in several countries there is a wealth of quality geographic information publicly available (for example in the USA, all addresses, streets, postal codes are fully geocoded and publicly available), in Greece there is no central government agency to provide such data. Some organizations, such as the Greek Postal services or the Greek telecommunications agency offer their information for free on the Web, but only for limited Web requests and not as a single downloadable file.

Another problem with Greek geographic data is its quality. Since Greek geographic data originates from various sources, there is no central organization for organizing and filtering duplicate or erroneous data. Additionally, most of the providers of such data are located outside of Greece. Therefore, most of this information is depicted in Latin characters creating the issue of converting this information to the Greek alphabet. There are also inconsistencies with respect to the projection and reference system used for the geo data, e.g., some providers use WGS84, others use the Greek EGSA87 projection system.

The geographic data acquired from the various sources, had to be filtered before it could be integrated. Such a process was automated as much as possible to minimize costly human intervention and allow for a general extensibility of our system.

The following Greek geographic data was stored in our database:

- Table *attiki*: This dataset includes about 6100 streets and about 500 squares geocoded with 300m precision.
- Table *gns*: includes many (about 44,000) Greek placenames with many alternative ways of expressing them and including their approximate geographic location. This dataset was derived from the GEOnet Names Server (GNS) [24] that provides access to the National Geospatial-Intelligence Agency's (NGA) and the U.S. Board on Geographic Names' (US BGN) database of foreign geographic feature names. This database is the official repository of foreign place-name decisions approved by the US BGN. This dataset was modified extensively, mainly during the conversion of Greeklish names to Greek [6], including alternative spelling. In this dataset, each placename has a Unique Name Identifier, along with a field containing information about the type of each placename (city, state capital, village, river, etc.).
- Table *mapdecode*: contains the geographic location of Greek placenames (6000) and various streets (66000) [11]. Some of the geographic data contained in this dataset is not entirely correct, but its significant size was a good starting point for the data collection process. The Greeklish issue had to be addressed as well.
- Table *telephone*: contains all the telephone prefixes (~240) for each Greek city and many (~1,800,000) telephone numbers in Attiki, Greece, fully geocoded. This dataset required extensive processing, in order to geocode the telephone numbers, using from the previous datasets (phone number → address → coordinate information).
- Table *postals*: includes all postal codes for Greece, fully geocoded. This dataset was geocoded using the previously mentioned datasets.

2.7 Performance Assessment

The performance of the geoparsing/geocoding toolkit was evaluated by (i) comparing it to an alternative imeplementation of the geoparser and by (ii) comparing the overall geocoding performance to tools available in the market. What follows is a brief overview of the obtained results.

As an alternative to the JFlex implementation, the Text Mining toolkit GATE [8] was used for parsing. Using JFlex for the implementation of the cascaded regular grammar transducers, combined with optimized Java code, resulted in faster and more effective parsing of Web pages when compared to the results that GATE was able to produce. In our GATE implementation, parsing a single page required an average time between 2 and 2.5 seconds. Using our new optimized system, the average time required to parse the same Web page dropped to below 1 second. For the tests to be comparable, the same hardware platform was used.

To assess the performance of the proposed geocoding method, a catalog of 1800 actual delivery addresses in Attika, Greece that contain spelling errors, incomplete addresses, alternative place names was geocoded. The overall process took less than 2 seconds on a typical PC. In comparison, using the geocoding feature of the ESRI ARC software suite took one hour. When comparing the actual number of addresses that was geocoded, the result becomes even more impressive. Our geocoding tool

managed to geocode 95% of all addresses, while the ESRI tool only identified 82%.

2.8 Summary

Our proposed geoparsing/geocoding module provides *fast* and *effective automatic geocoding* of Web pages. However, there are certain issues that this fully automatic approach cannot entirely overcome. One such issue is the disambiguation of geographic entities. Overall, there are two types of ambiguity, related with geographic data:

- The geo/non-geo ambiguity occurs when the name of a geographic location shares a non-geographic meaning as well, such as a person's name (e.g., Washington) or a common word (Turkey). This type of ambiguity is very common in Greek place names as in many cases, famous person are used to name places, e.g., the Athens International Airport is also referred to as Eleftherios Venizelos airport, named after a famous Greek politician.
- The geo/geo ambiguity arises when distinct geographic locations, possibly of different scale, share the same name, as in London, England vs. London, Ontario or Ontario, Canada vs. Ontario, California, USA.

Another issue is the quality of the geocoding result, which depends heavily on the quality of the available geo data used. Although, a large body of data was collected, i.e., ranging from reverse phone directories to map data, certain errors in the automatic geocoding due to inaccuracies in the geo data are inevitable. The main errors encountered are as follows:

- Inaccurate coordinate information for a geographic entity.
- Text portions of a Web page
    - that do not contain geographic information are erroneously geocoded,
    - that contain geographic information are not recognized by our automatic geocoding system.
- Only a subset (e.g., "Korinthos") of a phrase (e.g., "Ancient Korinthos") that contains geographic information is recognized by our automatic geocoding system.

In order to overcome the inevitable limitations of our automatic geocoding system, we decided to improve its efficiency by allowing the user to manually intervene (add, delete and modify) the results returned by the automatic geocoding results. This semi-automatic geocoding process is described in the following section.

3. SEMI-AUTOMATIC GEOCODING

A highly intuitive way to support semi-automatic geocoding of Web pages, i.e., allow for human intervention in the process, is by means of Web browser functionality itself. We developed a *Web browser extension* that (i) allows the manual geocoding of text portions of a Web page and (ii) allows updating (including deletion) of automatically generated geocoding results. This proposed approach is especially helpful for persistent Web pages such as Wikipedia, i.e., pages that have a definite value to the community, are well maintained and are modified rather seldomly. In that sense, geocoding can become a regular part of Web page authoring.

Our semi-automatic geocoding application should be embedded in a web browser in order to be efficient and user friendly. We achieve that by developing a custom Web browser extension for Mozilla Firefox. We opted for Firefox since it is highly customizable and supports most popular Operating Systems. Moreover, the process of writing extensions for Firefox is well documented. Most Firefox extensions are written in JavaScript. JavaScript also contains libraries for modifying the layout of a web page as it is displayed on an individual user's browser. Our Firefox extension is the central node of our semi-automatic geocoding application, controlling all interaction.

The basic idea behind the overall functionality is to highlight geocoded content on the Web page itself, e.g., by means of highlighted text. Should the user click on any highlighted content, a map is displayed, showing the respective geographic location. Geocoding results are stored centrally accessible through the Web. The automatic geocoding tool of Section 2 is wrapped in a Java Servlet. It is only executed when the Web page has never been geocoded before or when its contents have changed. To visualize the geocoding on a map, the Google Maps API [16] was used. This API allows the embedding of Google Maps on web pages.

The basic design of our semi-automatic geocoding application is shown in Figure 2. The semi-automatic geocoding application is divided into several components: The Firefox extension which constitutes the system's client side, and the Java Servlet, the Google Maps web service and the central database, which collectively constitute the server side of our application.

The following sections give a detailed description of the various application components.

3.1.1 User Interface – Firefox Extension

The Firefox extension is the user interface of our application. One of the actions that can take place is the marking - highlighting of geographic entities located on a web page currently shown in the web browser. This action is invoked by the pop-up menu entry "View geo info" (cf. Figure 3). This pop-up menu becomes available only after our extension is properly installed on the user's Firefox browser. The highlighted text portions in Figure 3 originate from the results of both automatic and manual geocoding processes stored in our central database.

The automatic geocoder is invoked either when the loaded page has never been geocoded before or when its content has changed since the last time the page was geocoded. In persistent Web pages like Wikipedia, where changes in content are rare and do not affect the entire text, our geocoder only geocodes the modified sections.

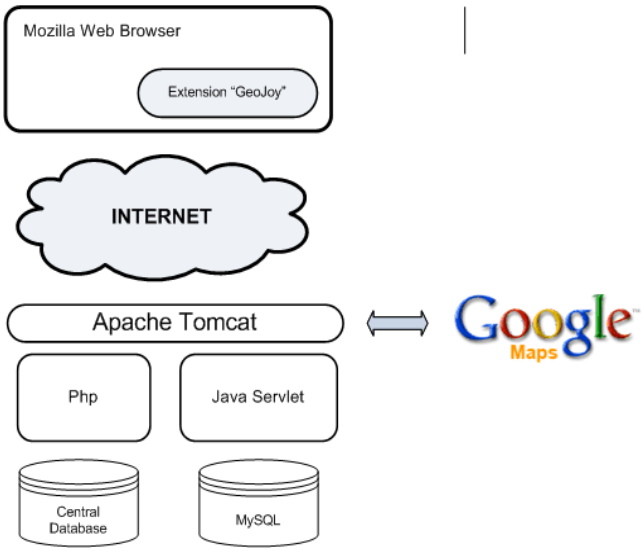


Figure 2: Basic semi-automatic geocoding system architecture

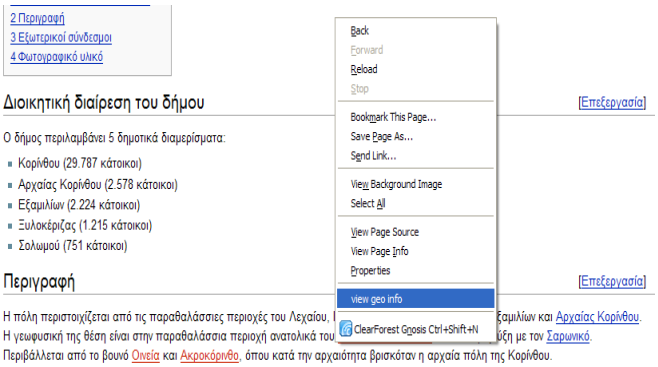


Figure 3: The Firefox pop-up menu, after installation of our extension

In our implementation, the central database containing the geo data and the geocoding results, and the automatic geocoder are hosted on a server running Apache Tomcat and PHP. The communication between the server and the Web browser (extension) is achieved using AJAX. This approach involves transmitting only a small amount of information (usually in XML format) to and from the server, in order to give the user the most responsive experience possible. A JavaScript function is called whenever information needs to be requested from the server. Instead of the traditional model of providing a link to another resource (such as another web page), each link makes a call to the AJAX engine, which schedules and executes the request. The request is done asynchronously, meaning that client-side code execution does not wait for a response before continuing.

Interaction with the central database is handled through PHP, a server-side scripting language. All information exchanged between the central database or the Java Servlet and the browser extension is encoded in XML.

The geocoding of a Web page is highlighted as they are retrieved from the central database (cf. Figure 4). Whenever the mouse pointer is placed above a highlighted text portion, a hover-over menu appears (cf. Figure 5) with the following three options:

- Locate the geocoding (using Google Maps).
- Locate the geocoding with the option to edit it (by dragging the marker to the right location).
- Delete the geocoding

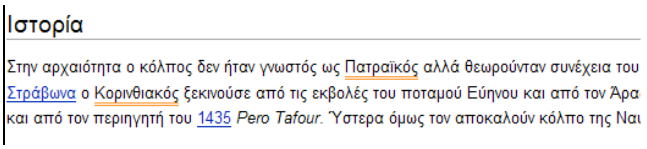


Figure 4: Marking – Highlighting of geocoding results

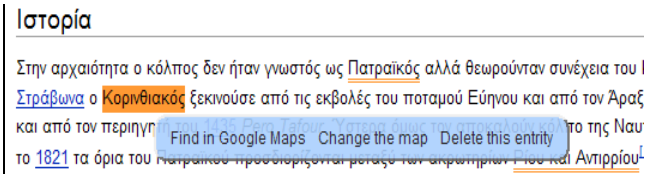


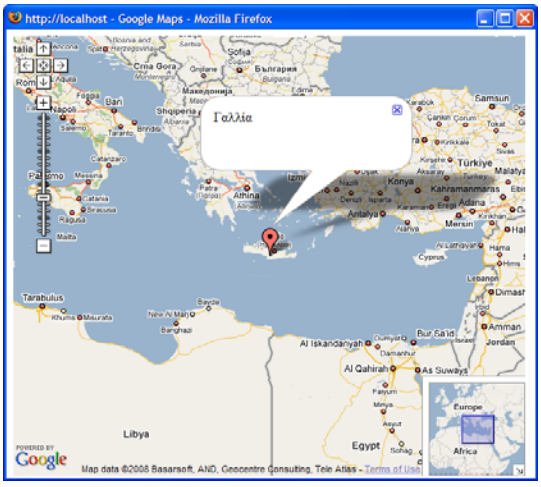
Figure 5: Hover–over Menu

The user can invoke any of the above three actions by clicking on the appropriate submenu. The first submenu "Find in Google Maps" opens a Google Map (in a new window) with the geographic entity's location marked. The second submenu, "Change the map", opens the same Google Maps window as before, where the marker representing the entity can be dragged to modify the geocoding. Upon modifying the geographic location, the user need only to confirm the modification for the update to be stored.

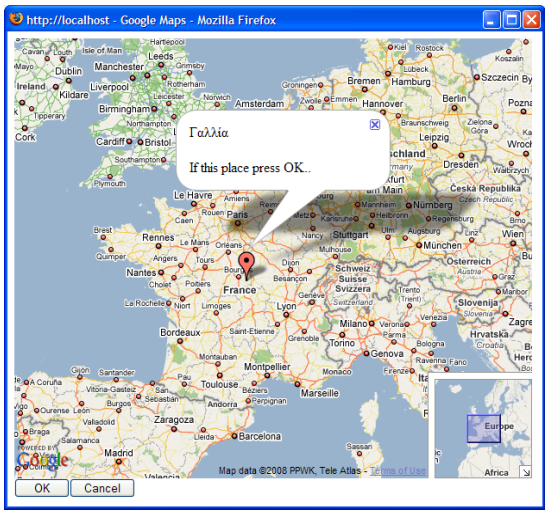
This functionality is demonstrated in Figure 6(a), where France was erroneously linked to homonymous village in the Greek island of Crete. This error can be rectified simply by dragging the red marker to the actual location of France (cf. Figure 6(b)).

Finally, the last submenu can be used to delete entities that were erroneously assumed to have a spatial extent by the geocoder.

Besides enabling the user to manually improve the geocoder's accuracy, the browser extension also allows her to add new geocodings. The user simply has to mark the respective text portion and select "geotag" (cf. Figure 7(a)) from the Firefox pop-up menu. A new Google Map window appears, where she may select the exact location ("geotag this point") to geotag the text portion (cf. Figure 7(b)). The geocoding is then stored in the central database.



(a) original entry – "France" linked to Crete



(b) updated entry

Figure 6: Updating geo-coding information

3.1.2 *Visualization – Google maps*

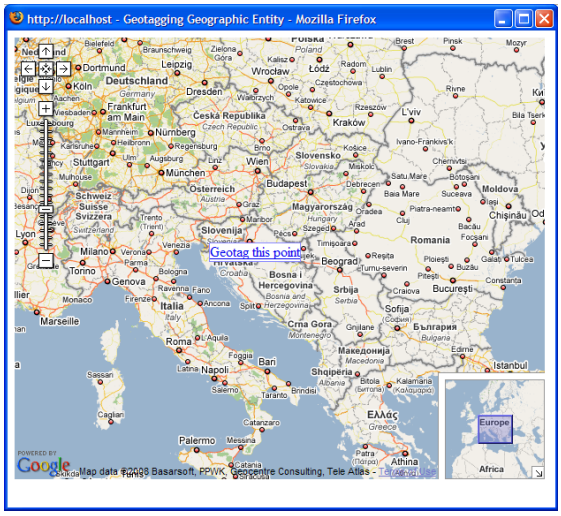
An important part of our application is the visualization of geocoding. For this reason, we used the Google Maps web service, which was essential for:

- visualizing the geographic extent of entities;
- rectifying said extent; and
- defining the geographic extent of newly added geographic entities

3.1.3 *Persistent Storage –Central Database*

A central database is used for storing location information for each web page. For every page, identified by its URL, and for every geocoded text portion, identified by its position on the page, the repository maintains (i) the respective geographic co-ordinates; (ii) a timestamp, indicating the version of the Web page that was geocoded; (iii) whether the geocoding was manual, automatic, or manually modified after its creation; and (iv) the desired zoom level for the map display.

(a) context menu - GEOTAG



(b) map interface – Google Maps

Figure 7: Adding a geocoding tag

4. EVALUATION

This section presents the results of geocoding the example Wikipedia page of "Κόρινθος" (Korinthos), Greece to illustrate the impact of our semi-automatic geocoding approach.

Figure 8 shows the automatic geocoding results. The phrase that is geocoded is underlined. Phrases that were not recognized, phrases like "κοινά" (commons) and "wiki" that have been wrongly highlighted and other phrases ("Αρχαία Κόρινθος") that were not extracted in their entirety (only "Κόρινθος" was marked) are circled.

Figure 9(a) is a visualization of the raw geocoding results in Google Earth. Although the marked places should be around the area of Korinthos (gray rectangle at the center of the map), some of them are scattered all over Greece due to the above errors.

By manually updating the geocoding result, using the tool described in Section 3, missed or mislabeled entries of Figure 8 can be corrected. Figure 9(b) presents the respective Google Earth visualization of the results.

The differences between the two versions are obvious not only in terms of marked phrases on the web page but also in terms of the

Google Earth visualization, with the pins indicating the geocoding being centered at the Korinthos area.

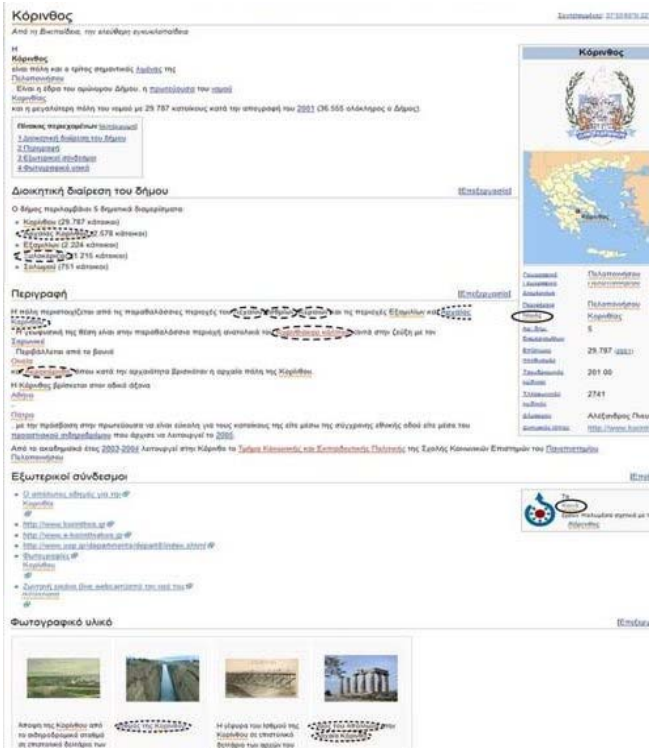


Figure 8: Automatic geocoding and missed information

5. CONCLUSIONS AND FUTURE WORK

The spatial aspect of information is becoming increasingly important as it can be used as an unambiguous, yet discriminative search criterion for information. This work presented (i) an efficient automatic geocoding framework specifically tailored to Greek content resources that was (ii) wrapped in a browser extension to facilitate (a) manual correction of the geocoding results, (b) public access to the tool and (c) the creation of a public repository for storing the geocoding of persistent Web pages. In that, the developed technologies advocate a community-based effort for the creation of spatial metadata for Web resources. The prototype implementation of the tool has been tested using Greek Wikipedia pages and the next step will be to make the software publicly available under an open-source licensing scheme. The collaboration model advocated by this work is community-based, similar to the one used by Wikipedia, Freebase and other public collaborative data repositories. Our approach, however, can be extended to less controlled scenarios, where multiple users annotate a single page, possibly in conflicting manners. We are investigating such extensions, by incorporating a voting scheme, where the edits of all users, optionally weighted by their "rating" in the system, are used to obtain a single, non-conflicting answer

The directions for future work are to improve the overall quality of the automatic geocoding result by adding additional geo data

(a) automatic geocoding

(b) updated geocoding

Figure 9: Visualization of geocoding results

resources. In addition, in cooperation with partners having the respective language processing know-how, the tool should be ported to other languages as well. With respect to geocoding Web pages, one has to consider the significance of information contained on a page with respect to the layout. Investigating the spatial arrangement of text on the page could provide significant insight and be used as a preprocessing step for the geocoding, i.e., weigh the geocoding with respect to where the content was found on the page (cf. [26]).

References

[1] E. Amitay, N. Har'EL, R. Sivan, A. Soffer. Web-a-Where: Geotagging Web Content. In Proc. of SIGIR, pages 273-280, 2004.

[2] A. E. Axelrod. On Building a High Performance Gazetteer Database. Technical Report, MetaCarta, electronically available at

http://www.metacarta.com/Collateral/Documents/English-US/Building-high-performance-gazetteer-Axelrod.pdf. Current as of June 2008.

[3] M. Bacchin, N. Ferro, and M. Melucci. A probabilistic model for stemmer generation. Information Processing and Management, 41(1), pages 121-137, 2005.

[4] K. A. V. Borges, A. H. F. Laender, C. B. Medeiros, C. A. Davis. The Web as a Data Source for Spatial Databases. In Proc. 4th ACM Workshop on Geographical information retrieval, pages 31-36, 2003.

[5] E. Brill. A Simple Rule-based Part of Speech Tagger. In Proc. 3rd Conf. on Applied Natural Language Processing, 1992.

[6] A. Chalamandaris, A. Protopapas, P. Tsiakoulis, S. Raptis. All Greek to me! An Automatic Greeklish to Greek Transliteration System. In Proc. 5th Int'l Conf. on Language Resources and Evaluation (LREC), 2006.

[7] J. Cowan. TagSoup parser. http://home.ccil.org/~cowan/XML/tagsoup/. Web page, current as of June 2008.

[8] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In Proc. 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), 2002.

[9] P. DeRose, X. Chai, B. J. Gao, W. Shen, A. Doan, P. Bohannon, X. Zhu. Building Community Wikipedias: A Machine-Human Partnership Approach. In Proc. ICDE, pages 646-655, 2008.

[10] J. Ding, L. Gravano, N. Shivakumar. Computing Geographical Scopes of Web Resources. In Proc. VLDB, pages 545-556, 2000.

[11] R. Elsinga. www.elsinga.org. Web page, current as of June 2008.

[12] Explore Our Pla.Net. RSS to GeoRSS Converter. Web page http://exploreourpla.net/2006-06-08/georss- feed-reader-shows-podcasts.html, current as of June 2008.

[13] H. Foundalis. The Details of Modern Greek Phonetics and Phonology. Web page http://www.cogsci.indiana.edu/farg/harry/lan/grphdetl.htm, current as of June 2008.

[14] A. Fuxman, E. Fazli, R.J. Miller, ConQuer: Efficient Management of Inconsistent Databases., SIGMOD, pages 155-166, 2005

[15] M. Gilleland. Levenshtein Distance, in Three Flavors, http://www.merriampark.com/ld.htm, 2000.

[16] Google Inc. Google Maps API. http://code.google.com/apis/maps/. Web page, current as of June 2008.

[17] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Divesh Srivastava, Approximate String Joins in a Database (Almost) for Free. In Proc. VLDB, pages 491-500, 2001

[18] L. Gravano, V. Hatzivassiloglou, R. Lichtenstein. Categorizing web queries according to geographical locality. In Proc. of CIKM, pages 325-333, 2003.

[19] G. Klein, S. Rowe, and R. Décamps. JFlex - The Fast Scanner Generator for Java. http://jflex.de/. Web page, current as of June 2008.

[20] A.J. Lait, B.Randell. An Assessment of Name Matching Algorithms, Technical Report, Dept. of Comp. Sci., University of Newcastle upon Tyne , 1993

[21] M.L. Lee, T.W. Ling, W.L. Low, IntelliClean: A Knowledge-Based Intelligent Data Cleaner, In Proc. KDD, pages, 290-204, 2000

[22] MetaCarta Inc. Company homepage. http://www.metacarta.com/, Web page, current as of June 2008.

[23] K. McCurley. Geospatial mapping and navigation of the web. In Proc. 10th WWW conf., pages 221-229, 2001.

[24] NGA. GEOnet Names Server (GNS). http://earth-info.nga.mil/gns/html/index.html. Web page, current as of June 2008.

[25] G. Petasis, G. Paliouras, V. Karkaletsis, C.Spyropoulos, I. Androutsopoulos. Resolving Part-Of-Speech Ambiguity in the Greek Language Using Learning Techniques. In Proc. CoRR, 1999.

[26] S. Raghavan, H. Garcia-Molina. Crawling the Hidden Web. In Proc. VLDB, pages 129-138, 2001.

[27] E. Rahm, H.H. Do, Data Cleaning: Problems and Current Approaches, IEEE Bulletin on Data Engineering, vol 23(4), pages 3-13, 2000.

[28] K. Sgarbas, N.Fakotakis, G.Kokkinakis, A PC-KIMMO-Based Bi-directional Graphemic/Phonetic Converter for Modern Greek, Literary & Linguistic Computing, Oxford University Press, vol 13(2), pages 65-75, 1998.

[29] R. Waters. Way to go? Mapping looks to be the Web's next big thing. Financial Times, May 22, 2008.

[30] Yahoo Inc. Yahoo Yellow Pages. http://yp.yahoo.com/. Web page current as of June 2008.