Automated Processing of Postal Addresses

Konstantin Clemens
Deutsche Telekom Laboratories, TU Berlin
Service-centric Networking
konstantin.clemens@campus.tu-berlin.com

***Abstract*—Postal addresses are involved whenever post mail is to be delivered to an addressee, or to describe the location of a person or organization. The world-wide adoption of postal addresses and the lack of a centralized control entity led to very heterogeneous postal address formats. Even within a single country various postal address formats may be used. In this paper the main tools for processing postal addresses in an automated fashion are compared: Multidimensional indices, address patterns and gazetteers. Special focus is put upon supporting as many heterogeneous postal address formats as feasible with the various tools.**

***Keywords*-postal automated processing; postal addresses; address patterns; gazetteer; spatial indices**

I. INTRODUCTION

Postal Addresses identify addressees so that no room for ambiguity is left. There was no need for that before modern postal systems began offering their service to the broad public. Until every citizen was eligible to receive post office mail, many regions lacked house numbers or street names. That still is the case in many areas. E.g., in Japan often named houses are used over numbered houses on named streets. Extremely poor neighborhoods without official organization as the Brazilian favelas often lack street names and house numbers too.

To disambiguate the name of an addressee, postal addresses utilize the addressee's whereabouts. A typical postal address is composed from multiple administrative areas, a postal code, a street name, and a house number. As names are reused across different streets often, administrative areas are disambiguating street names. The street name and house number in turn identify a specific house where the post office mail can reach the addressee. Postal codes are the latest innovation in postal addresses. Being invented in the first half of the 20^{th} century, postal codes usually encode information redundant to administrative areas. Sometimes postal codes are as accurate as streets or blocks. Besides functioning as a verification to confirm or repair a postal address, postal codes are mainly used to identify a post mail distribution center easily. They thereby enable automated mail sorting. Postal codes are not used by every addressing system. Also the way location data is encoded in postal codes obeys local requirements and varies from country to country. Similar to country specific postal codes, street naming conventions vary by region. While in European countries most streets are named by destinations, areas, landmarks, famous people or events, in USA and Canada many streets are incrementally numbered or labeled and carry directional information with regard to a reference point. House numbering schemes vary just as much as street naming conventions. House numbers may be assigned incrementally or by distance within a street or a block. In the latter case, usually blocks are numbered incrementally. These schemes can be combined with distributing even and odd numbers on different street sides. Some times a single house number is assigned to multiple buildings. A single house number is then not sufficient any more. Additional information identifying a block, a floor or a suite is then added to the postal address. Depending on how important an addressee is in a given region, the location specified in the address may be less specific. Parts of the location may be omitted still assembling a postal address identifying an important addressee unambiguously. E.g., children, addressing post mail to Santa Claus reasonably expect the addressee to be unambiguous, despite the location part of the address being not more specific then North Pole.

Usually computers are only utilized to make sense of location parts of an address. Reaching the addressee (and handing the post mail over) is left to human beings. Because of that there is no unified scheme applied for specifying addressees within large organizations. Arbitrary department hierarchies and case workers may be named as addressees therefore.

Nowadays addresses are present in a variety of sources. Postal addresses are hand-written or printed on letters, forms and packages. Addresses are stored in proprietary schemas in databases of users, customers, orders, etc. GIS databases store addressable entities too, usually with coordinates only not including their postal addresses. Often address parts are also mentioned in free text. From that we can name the following use cases:

Computer systems can be used to

1) derive complete and valid postal addresses from numerical location representations.
2) parse postal addresses from post mail or forms.
3) identify postal address parts in free text documents.
4) construct full and partial valid postal addresses from entities stored in proprietary database schemas.
5) validate, enhance and fix given postal addresses.

 ISBN: 978-1-61208-251-6

Use case (1) is also referred to as *reverse geocoding*. Measured coordinates, for example, attached to photographs by GPS-enabled cameras, can be presented to users in a more comprehensible way deriving named address parts at the coordinate's location. Use cases (2) and (3) are the daily business at banks, hospitals, government agencies, and such. These organizations do receive filled out forms and informal documents that contain postal addresses to be parsed. Use case (4) is the reverse of (2). It is applied to generate and print an address suitable to post mail from separate address parts. Finally use case (5) is of value for every system gathering postal addresses, to sort out fake or erroneous input. Also having a sparse but unambiguous input address, it is use case (5) to enrich it with all address parts required for a fully specified address.

For dealing with location data contained in postal addresses in an automated way, the common tools are *patterns, gazetteers* and *multidimensional indices*. Patterns allow splitting an address into its address parts and assembling it back according to a predefined format. Gazetteers allow looking up named entities and deriving attached information. The information retrieved from gazetteers may explain hierarchical dependencies between the entities or contain other metadata useful for processing addresses. Finally spatial indices support the reverse look up of geographical entities. Given a point or a polygon, such indices allow deriving address entities that enclose, overlap or touch the specified area.

This paper compares the named tools for processing postal addresses in an automated fashion. Special focus is given to the variety and complexity of postal address formats in use. The goal of this paper is to summarize the merits and demerits of the three tools, thereby identifying gaps for further tooling.

In the following section postal address formats in general are discussed. Next, these tools and their selected variants are analyzed with regards to the defined use cases. Note that it is outside of the scope of this paper to analyze how address texts get digitalized. Hybrid approaches for extending OCR with gazetteers to reach better address recognition while scanning printed or hand written addresses are available [1] but are not included in this analysis.

II. Postal Addresses

As discussed, there is a variety of postal address formats. As most postal services operate within country borders, most postal address formats differ from country to country. In some countries though, addressing differs within the country depending on the addressee's location being a city, a rural area or a specific region. There are also special address formats for special cases, as the military, where the location of the addressee alters quickly, or shall not be known to the public. Table I illustrates addresses in various formats valid in USA, France, Ireland, and Japan. All addresses

Table I
Example addresses in differing formats within a country.

JOHN DOE ABC COMPANY 1401 MAIN ST FALLS CHURCH VA 22042-1441 UNITED STATES OF AMERICA	SSGT KEVIN BEASLEY UNIT 2050 BOX 4190 APO AP 96522-1215 UNITED STATES OF AMERICA
Monsieur Jean DELHOURME Chez Mireille COPEAU Apartment 2 Entrée A Bâtiment Jonquille 25 RUE DE L EGLISE CAUDOS 33380 MIOS FRANCE	Madame Isabelle RICHARD LE VILLAGE 82500 AUTERIVE FRANCE
Ms M Sullivan 12 Morehampton Road DUBLIN 4 IRELAND	Mr J Murphy ABC Company Limited 1 Dublin Road Portlaoise CO LAOIS IRELAND
Mr Taro Tanaka 2-17-10, Aicicho Naka-ku, YOKOHAMA 231-0012 JAPAN	Mr Taro Tanaka 2338 Shiokawa Maruko-machi Chisagata-gun, NAGANO 386-401 JAPAN

were taken from the address examples provided by the Universal Postal Union UPU[2]. In the first row on the left hand side an typical US address containing *an addressee, house number, street name, town, state, zip-code* (the US-version of a postal code) and *country* is shown. The other address in the same row is addressing military personal, and contains no location information. Only numbers are used that unambiguously identify the addressee but only bear meaning to the Army Post Office of USA. In the next row two French addresses are displayed. While the address on the left refers to an addressee in the city Mios, the address in the right column points to the village Auterive in a rural area. The urban address format also contains the addressee's apartment number and entrance as extended location information. Its parts are *addressee, apartment, entrance, house number, street name, district, postal code, town* and *country*. The rural address format includes *addressee, district, postal code, town* and *country* but names neither a street name nor a house number. In the third row two Irish addresses are confronted with each other. In this case the address formats differ because of regional diversity. Postal codes in Ireland have only been introduced in Dublin. Thus, while the Dublin address contains *addressee, house number, street name, postal code* and *country*, the address with the location information outside of Dublin names the county instead. It consists of *addressee, house number, street name, town, county* and *country*. In the last row two Japanese addresses are shown. The left address points to an urban area. It contains *zone, block, house number, district, town, prefecture, postal code* and *country*, as in Japan street names are not common. Instead houses numbered within blocks which in turn are numbered within zones that assemble a district. If required, a fourth number could be added to that address to specify an apartment. Addressing rural

 ISBN: 978-1-61208-251-6

areas in Japan uses a different format. The address on the right contains *addressee, house number, district, town, area, prefecture, postal code* and *country*. The area hereby is a part of the prefecture and may contain multiple small towns.

Postal addresses do contain redundant information to make sure that the addressee is identified unambiguously. This redundancy allows postal addresses to be inconsistent. An address may be contradicting containing a valid postal code that does not match the specified administrative areas, or a street name of a street that is located in a district other than the one specified. An address may be incomplete not specifying some vital address parts. Due non-distinct naming of address entities, incomplete addresses may be ambiguous, making post mails undeliverable. Because such addresses do exist, when referring to locations in a computer system, usually numeric location identifiers are used. Such identifiers can be latitude and longitude values of points, a hash value computed from latitude and longitude, or any reference on a Cartesian coordinate system. However, as soon as humans are involved interacting with computer systems about location information, numeric identifiers do no longer suffice. Humans prefer postal addresses mainly for two reasons: First, common adaptation of postal addresses makes them easy comprehensible. Similarly the used scale when referring to time is comprehensible for every human, although it is not based on the decimal system that is employed everywhere else. Second, if an address is not known to a human being, a vague understanding of the location is supported by the named entities of higher hierarchies. In the worst case the only known address part of a given address is the country. Even then, that data is directly derivable data for a human, more then looking at a numeric value.

III. Evaluation of Postal Address Processing Tools

In this section the tools suitable for processing postal address formats are analyzed with regards to their capability for serving the use cases defined in the introduction. Table II summarizes the outcome. As we will see, multidimensional indices are only capable to look up postal addresses for given coordinates. Patterns support parsing formatted addresses to elementize the addresses parts. Also the reverse procedure assembling an address from separate address parts is executable using patterns. Gazetteers support all the use cases that multidimensional indices and patterns support. That is due to gazetteers containing multidimensional indices and patterns as internal components. In addition, gazetteers may support parsing address parts from unstructured text. As gazetteer contain only valid address parts, they implicitly validate the addresses given.

Table II
Overview of tools and the use cases they serve

use case	Indices	Patterns	Gazetteers
(1) derive from coordinates	X		X
(2) parse structured		X	X
(3) parse unstructured			X
(4) construct from parts		X	X
(5) validate			X

A. Multidimensional Indices

The algorithmic task for efficiently resolving coordinates on a surface into geographical objects covering the coordinates has been solved in a variety of ways. R-Trees [3], Quadtrees [4] and many specialized variants of these [5][6][7][6] make resolving coordinates performant and scalable. A necessary requirement for building such an index is the knowledge about the geographical spread of the postal address parts. With the spatial index at hand it is a straight-forward task to resolve coordinates to the postal addresses that address the same location. Multidimensional indices are therefore a suitable tool for serving use case (1) defined in the introduction.

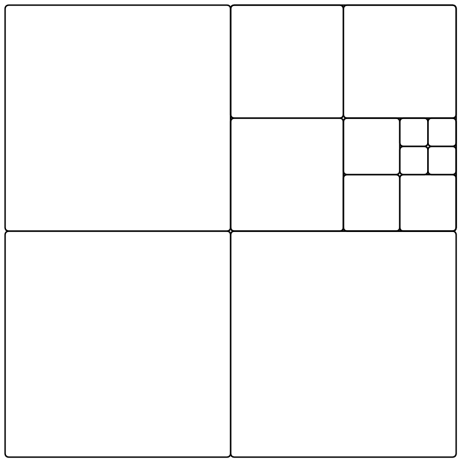

Figure 1. Organizational diagram of a two dimensional Quadtree. Each sector containing more than a specified amount of elements is split in four equally sized sectors.

Figure 1 illustrates the internals of a two dimensional Quadtree. Each square plane is split into four equally sized sectors, as soon as the maximum bound for entries in a single section is exceeded. Depending on the distribution of the entries, some parts of the area are split to smaller chunks, while others remain unsplit. The resulting rectangles are referenced as leaves in an unbalanced tree. The tree grows when on an insert into the Quadtree a sector needs to be split. It then becomes a tree node with four new children.

Often Quadrees are modeled using points with associated location data. The concept of points, e.g., coordinates with zero extent, does not reflect the real world. Houses, blocks, streets, etc. are all objects that have extent. Therefore some Quadtrees are modeled using polygons instead of points. Both points and polygons may be used as input parameters

 ISBN: 978-1-61208-251-6

to look up address data. Input coordinates may lie exactly between indexed points, or on the border of two polygons. Polygons may contain several indexed points or overlap with multiple indexed polygons to the same part. To resolve these ambiguities, Quadtrees can employ various strategies, i.e., always choose the indexed entity with the lowest coordinate values.

B. Patterns

Schema patterns are by the simplest tool that can be used to process postal addresses. As patterns are stateless, applying patterns is parallelizable easy. Also there is no data used in a pattern that needs to be kept up to date.

It seems that patterns are a good fit in fulfilling use case (2) from Section I. If the addresses to be parsed contain a delimiter splitting the address parts, simple regular expressions suffice for that purpose. Table III illustrates a Ukrainian sample address with annotated address parts taken from the UPU address samples. The address contains natural delimiters surrounding address parts: Either a whole address part is on its own line, or, if multiple address parts are on the same line, they are separated by commas. Similarly the Japanese address in the fourth row of Table I is machine-readable using simple regular expressions.

That does not apply for every postal address format. The address formats for USA, France and Ireland on Table I use a single space to separate address parts. As some address parts contain spaces in their name, a simple regular expression cannot recognize the exact range of every address part. To approach this problem, Hidden Markov Models (HMM) [8] can be trained to parse the address parts. Conceptually HMMs are statistically learned patterns that pick the most probable path through a finite state automaton to determine the type of input tokens. It is common to use HMMs for elementising address parts as streets names and house numbers from single address lines [9][10].

Both regular expression and HMMs patterns are not flexible enough to support parsing of *all* address formats. Addresses in use contain errors and might have a required delimiter missing. Also, as discussed, not all addresses contain all address parts. Especially addresses of important addressees might omit address parts required by a pattern. Finally as multiple postal address formats are in use, it often is not unambiguous which patter to apply on which address. Therefore Patterns are only serving use case (2) well, if the input addresses to be parsed are of a single known format.

Table III
UKRAINIAN SAMPLE ADDRESS WITH ANNOTATED ADDRESS PARTS

Melnik Ivan Petrovitch	addressee identification
Vul. Lisova, 2, kv.3	street, house, apartment
s. Ivanovka, Semenivsky r-n,	locality, district
TCHERNIGIVSKA OBL.	province
15432	postcode
UKRAINE	country

Table IV
JOINABLE AND NON-JOINABLE ADDRESS FORMATS

Denmark	addressee street house	floor admin2 postcode admin1	country
Germany	addressee street house suite	postcode admin1	country
Denmark joined with Germany	addressee street house suite	floor admin2 postcode admin1	country
Great Britain joined with USA	addressee house street	admin2 admin1 postarea postcode	country
Great Britain	addressee house street	admin1 postarea postcode	country
USA	addressee house street	admin2 admin1 postcode	country

Patterns can also be used to process addresses in the opposite direction. Use case (4) requires constructing of valid post addresses from separate postal address parts. Given that the schema of the source containing the address parts is known, applying a pattern rule to assemble these is a very easy step. However, that approach too is constrained by the variety of the postal address formats. If the source contains addresses that have to be assembled according to different postal address formats, the information which pattern to use on which address needs to be available as well.

Figure 2 shows how an address is assembled using a pattern. The separate address parts are retrieved from a proprietary database schema. The pattern instructs in which order to assemble the address parts and what separators to use. Note that not all the address parts retrieved from the database are used.

The UPU follows this approach with their product ”International Postal Address Components and Templates” (UPU S-42). It specifies rules for assembling the 35 identified address parts. UPU S-42 covers 246 countries but it needs to define regional postal address formats too. To estimate the amount of schemes in UPU S-42, for a reduced address part set of 17 address parts, address samples provided by the UPU have been annotated. The resulting 450 address formats have been joined as shown in Table IV: Two address formats may be joined only if the address parts present in both address sets are ordered equally. They are then combined into a joined address format that contains all address parts from both address formats, respecting the order of address parts in both original formats. In Table IV the two combined address formats may not be joined, because the orders of street name and house number and administrative level one and postcode are conflicting. The joinable address formats have been joined so that the amount of address formats was reduced to 41. The same effort has been undertaken

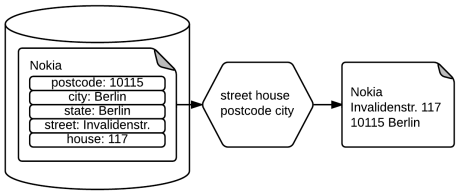

Figure 2. The address of the Nokia headquarters Berlin assembled by a pattern from a proprietary data base schema.

by the UPU. UPU S-42 is a ready alternative for composing valid postal addresses without the upfront analysis. However, it requires the data to be as fine-grained as the scheme used in UPU S-42 using the same 35 parts of a postal address. For denormalized data bases, and if not all address formats identified by UPU need to be constructed, a light-weight approach using patterns suffices, to construct postal addresses from their parts.

C. Gazetteers

As multidimensional indices discussed in Section III-A, gazetteers index location parts entities. Unlike multidimensional indices gazetteers index names of entities instead of their position. This way gazetteers support querying for address parts by name. Retrieved entities usually have attached metadata describing relationships to other named entities. This spanned structure may be browsed.

Gazetteers can be used for detecting address parts in free text and in structured addresses. Those are use cases (2) and (3) from Section I. For both use cases, as no clear boundaries of a single address part is described. It is not obvious what part of an address or a free text to use when querying a gazetteer service. Some gazetteers may be able to retrieve *all* address parts it finds in a query, solving the splitting problem transparently to the user. Others require multiple queries of *n-grams*. Splitting a unstructured text or an address of an arbitrary format is complex in implementation and execution. Therefore most gazetteers are not independent of the input but define a fixed postal address format that they support.

With all address parts being indexed in gazetteers, only valid address parts are retrievable. Using the relationship data – if present in the gazetteer – the parsed addresses are implicitly validated. That serves use case (5) from the introduction. Often the relationships of location entities are represented via their numerical location and spread. E.g., a gazetteer could derive that New Jersey is a part of the USA knowing the geographical spread of both entities. That would be implemented using a multidimensional index as discussed in Section III-A, enabling gazetteers serving use case (1) as well.

Named location entities are the base units for gazetteers. As entity names alone are ambiguous, gazetteers have to qualify found entities using other address parts. Particularly, to distinguish between equally named location entities, gazetteers have to name all location entities enclosing the equally named ones. This happens according to a schema from which, in combination with a pattern, an address can be constructed. Many gazetteers implement that feature on their end, solving use case (4) from Section I.

A gazetteer protocol has been defined by the Open Geospatial Consortium (OGC) [11]. The OGC Best Practices Document defines a gazetteer as ”a database used to translate between different representations of geospatial references, such as place names and geographic coordinates” (Section 4, page 12). The OGC gazetteer does not define a specific address format for processing, but defines a hierarchical model of named location entities. These entities are associated to features, which are not necessarily spatial. This way a gazetteer that complies with the OGC protocol supports looking up location entities by name or coordinates. Queries to the gazetteer are allowed to specify a filter on non-spatial features, which reduce the result list. Optionally retrieving location entities by a gazetteer specific ID is supported too. The protocol allows browsing the relationship of entities directly via links between them, or indirectly by fetching all entities that are within a certain bounding box. A gazetteer implementing the OGC protocol and filled with location parts of addresses would support use cases (1) and (4) only if a set of patterns matching the gazetteers internal schema was provided. The gazetteer is incapable of detecting multiple named entities in a query. It therefore requires the client to slice the input according to the boundaries of address parts, to support use cases (2) and (3). As any gazetteer, it does fulfill use case (5) as only valid location address parts are identified. Browsing the links between location parts allows discovering contradicting addresses.

Another example of a gazetteer is the Google Geocoder [12]. Unlike the multi-purpose protocol of OGC, Google’s gazetteer focuses on translating addresses into latitude and longitude coordinates and vice versa. For that the gazetteer protocol accepts requests with addresses only. No specific address format is required. As the OGC gazetteer, Google’s gazetteer support filtering results by specifying selected address parts as required. The address parts *route, locality, administrative area, postal code* and *country* are supported for filtering. Apparently Google’s gazetteer internally defines the patterns required to construct an address, as an assembled address is always part of each result. Google’s gazetteer supports use cases (1), (2), (4) and (5) as defined in the introduction. As the OGC gazetteer, for detecting address parts in free text (use case (3)) it remains to the client to slice the text into chunks that contain single address parts.

Many other implementations of gazetteers exist. For

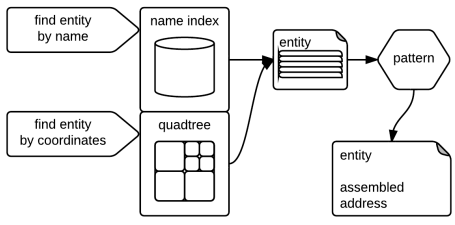

Figure 3. Gazetteer Data Flow. Depending on the request containing a name or a coordinate, either the name index, or the multidimensional index is queried. Addresses of retrieved entities are constructed through a pattern.

 ISBN: 978-1-61208-251-6

example, Densham et al. [13] describe a system that is extending a gazetteer with capabilities to parse free text. Clough et al. [14] have built and measured a system that assigns coordinates to web documents.

Overall gazetteers are more complex than patterns or multidimensional indices, as most gazetteers contain both patterns and multidimensional indexes internally. That implies that gazetteers have to maintain a list of patterns to construct addresses according to various formats. Also as multidimensional indices, the data on location entities needs to be kept up to date. Finally indexing named entities is a non-trivial task too. In sum, that makes gazetteers a tool that is complex to manage, while it does support more use cases simultaneously.

In Figure 3 a basic gazetteer set up is visualized. It includes a text index for name queries, a multidimensional index for coordinate queries and a pattern to assemble address parts of results. The visualized gazetteer stores entities with their address parts in both indices. To save space, other gazetteer implementations might only store references in their indices that point to a common data source. The entities are stored in a schema that can be assembled to correct addresses using patterns.

IV. Conclusion

The paper evaluated *gazetteers, multidimensional indices* and *patterns* as tools for processing postal addresses. It showed that there is no general pattern that is suitable to support parsing or constructing addresses, because multiple address formats need to be supported. The option to use specialized patterns for individual address formats is not available for all address sources, depending on the extent of regions that are being covered. Multidimensional indices resolve numeric location references as coordinate pairs into location entities. These however still need to be assembled to valid addresses. Also multidimensional indices need to be kept up to date for resolving addresses accurately. Gazetteers are the most versatile tool for processing postal addresses. Depending on implementation and interface, gazetteers can support parsing structured addresses and free text, constructing and validating addresses and resolving addresses from numerical references. To achieve that, gazetteers contain patterns and multidimensional indices internally, in addition to the text index for looking up named entities. Because gazetteers are such a complex tool there are efforts joining gazetteers with other address related tools to achieve better performance.

There is a gap in complexity between gazetteers and patterns. While gazetteers serve more use cases they also are by magnitudes more complex compared to simple address format patterns. One option to close this gap is to split address patterns in logical units of multiple address parts. These sub-patterns could be used in gazetteers that support retrieving combined address parts if these belong together. Another method to combine these tools are smart patterns that rely on data to automatically determine how to assemble a complete and correct address, depending on country and region.

References

[1] S. Srihari, "Recognition of handwritten and machine-printed text for postal address interpretation," *Pattern recognition letters*, vol. 14, no. 4, pp. 291–302, 1993.

[2] "Universal Postal Union," http://www.upu.int, Jun. 2012.

[3] A. Guttman, "R-trees: a dynamic index structure for spatial searching," vol. 14, no. 2, 1984.

[4] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[5] R. Kothuri, S. Ravada, and D. Abugov, "Quadtree and R-tree indexes in oracle spatial: a comparison using gis data," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, 2002, pp. 546–557.

[6] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, *The R*-tree: an efficient and robust access method for points and rectangles*. ACM, 1990, vol. 19, no. 2.

[7] Y. Kim and J. Patel, "Performance comparison of the R*-tree and the quadtree for knn and distance join queries," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 7, pp. 1014–1027, 2010.

[8] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4–16, 1986.

[9] A. Kornai, "An experimental hmm-based postal ocr system," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 4. IEEE, 1997, pp. 3177–3180.

[10] V. Borkar, K. Deshmukh, and S. Sarawagi, "Automatically extracting structure from free text addresses," *IEEE Data Engineering Bulletin*, vol. 23, no. 4, pp. 27–32, 2000.

[11] J. Fitzke and R. Atkinson, "OGC best practices document: Gazetteer service-application profile of the web feature service implementation specification-0.9. 3," *Open Geospatial Consortium*, 2006.

[12] "The Google Geocoding API," https://developers.google.com/maps/documentation/geocoding, Oct. 2012.

[13] I. Densham and J. Reid, "A geo-coding service encompassing a geo-parsing tool and integrated digital gazetteer service," in *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*. Association for Computational Linguistics, 2003, pp. 79–80.

[14] P. Clough, "Extracting metadata for spatially-aware information retrieval on the internet," in *Proceedings of the 2005 workshop on Geographic information retrieval*. ACM, 2005, pp. 25–30.

 ISBN: 978-1-61208-251-6