An Automatic Approach for Discovering and Geocoding Locations in Domain-Specific Web Data

Chris A. Mattmann[1,2] and Madhav Sharan[1]

[1]University of Southern California, Los Angeles, CA 90089 USA
Email: {msharan,mattmann}@usc.edu

[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA
Email: chris.a.mattmann@jpl.nasa.gov

Abstract

We present an automatic approach for discovering location names in WWW data culled from diverse domains. Our approach builds upon the Apache Tika, Apache OpenNLP, and Apache Lucene frameworks. Tika is used to extract text and metadata from any file. The text and metadata are provided to Apache OpenNLP and its location entity extraction model. The discovered location entities are then delivered to a gazetteer indexed in Apache Lucene derived from the Geonames.org dataset. This paper describes the overall approach and then explains in detail the challenges we faced, and the methodology that we employed to overcome them. We describe the evolution of our geo gazetteer process and algorithm and demonstrate the approach's accuracy in data collected in the DARPA MEMEX and NSF Polar Cyber Infrastructure efforts.

1. Introduction

The world-wide-web (WWW) is home to a rich diversity of and large volumes of data across many domains and disciplines. Broadly speaking, the WWW is home to exabytes of information [18], by some estimates[1] 10,000-50,000 file types, and according to [19], 259 thousand hours of video, 2 million new websites, 252, 000 new domains registered, and 6 billion likes on Facebook per day.

Throughout the data variety, volume and velocity referenced above there are many instances of textual representations of geo-locations. As an example, consider the DARPA MEMEX [11] program. MEMEX is a large program with 17 performers working to bring about new types of domain specific search engines and is specifically targeted at the deep and dark web. In MEMEX we have collected web page and textual data from many domains including Materials Research, and Autonomous Robots and other domains. Many of the research questions being asked of the data involve two key elements - *geospatial location* and *temporal constraints* - and center around textual and domain insights derived from data that requires location information. MEMEX and its challenges are comparable to our other work in the NSF Polar CyberInfrastructure program [3, 4, 10] where we have collected over 1.7 million URLs from three Polar oriented websites - the National Science Foundation (NSF) Advanced Cooperative Arctic Data and Information System (ACADIS); the National Aeronautics and Space Administration (NASA) Antarctic Master Directory (AMD); and the National Snow and Ice Data Center (NSIDC)'s Arctic Data Explorer (ADE). In that work we are also interested in location extraction from a variety of domain specific textual and other data on the web.

A straightforward way to answer questions about the collected web data in MEMEX and the NSF Polar CyberInfrastructure project related to geospatial location and time would be to use scientific data such as Hierarchical Data Format (HDF) [5] and/or NetCDF [15] data or shape files, etc., that have this information readily available and accessible. However, that is not the case in our MEMEX project as the location mentions are in the form of author affiliations (e.g., `University of Southern California, Los Angeles`); place name mentions in text (`Beijing, China`), and place name mentions in file metadata (`California`). Given

[1]http://fileext.com

that these location data are textual, must be disambiguated, may contain duplicate and/or unnecessary information; and do not have geospatial coordinates, it is difficult to directly answer any of the questions that MEMEX and the NSF PolarCyberInfrastructure and collaborating researchers have about the data. As an example the question `what is the density of data collected that mentions e.g., Central America?` involves a multi-step process: (1) collecting the mentions of Central America in text wherein which location names may be convolved with text, punctuation and other information; (2) finding synonyms including `Latin America`; `Brazil`, etc.; (3) finding locations within some acceptable range Central America (within e.g., 10 miles, 50 miles, 100 miles); (4) finding the latitude and longitude of those locations; and (5) plotting location and density on a map.

In the aim of answering domain-specific web queries involving location, our group at the University of Southern California and Jet Propulsion Laboratory, California Institute of Technology began a project that uses modern Information Retrieval software including Apache Tika [13], Apache OpenNLP [8] and Apache Lucene [14] to automatically extract and identify location mentions in domain specific web data. Apache Tika extracts text from *any* of the files identified in the Internet Assigned Numbers Authority (IANA) MIME Taxonomy [6], allowing for locations to be extracted from visible text and from metadata. Apache OpenNLP takes the extracted text and automatically identifies location mentions. The location mentions are then provided to a specialized Apache Lucene index built from the `geonames.org` dataset that allows for geocoding and geolocation from location names. The result *best* location and a set of alternate location names, and latitude and longitude coordinates are returned from the entire process allowing for locations to be automatically identified from the volume, velocity and variety of data available on the WWW and in the domain specific web.

In this paper, we will describe our approach and implementation for automatically identifying locations in text extracted from any file discovered on the WWW. Section 2 describes how we perform geographic topic identifcation from text. Section 3 describes specifics of locating information using the Lucene Geo Gazetteer including the process for constructing the Lucene index from the `Geonames` data, and the process of improving location ranking and identification from textual data. Section 4 rounds out the paper and describes future work.

2. GeoTopic Identification

Our approach is informed by basic observations resultant from early studies in the NSF Polar Cyber Infrastructure TREC-DD-Polar dataset [10]. The dataset is 158Gb of data crawled from 1.7 million URLs across a wide diversity (98) of file types including plain text (ASCII), HTML, audio, video, and e-mail. We performed preliminary research in 2015 to explore the data for mentions of location information and our results are shown in Table 1. The table demonstrates the source website (AMD, and/or ACADIS, recall Section 1); the particular science topics; the location related field, and any other notes about location information we discerned.

As can be gleaned from the table, there was no single source of location information in the data. Sometimes it was present in textual mentions e.g., in ACADIS, text was present in the description field, and/or in the first few sentences of text, and/or the title. Other times, e.g., AMD, there were location keyword fields in the metadata. Even in the case of AMD, location spatial coordinates were also sometimes available, but broadly that was not the case.

Our preliminary work evaluating location information in the DARPA MEMEX project also bore similar fruit. For example, we studied 75,000 publications in the Materials Research science domain, performing in particular at full text extraction from PDFs and HTML abstracts. For the Autonomous Robots research domain, we examined 50,000 publications spanning a number of open journals and private literature. In both cases, we extracted location mentions from author affiliations; from locations present in the text, focusing on identifying locations with frequent mentions as an indication of their importance; and we also examined locations mentioned in the PDF and abstract (HTML) metadata.

As there was no single field or textual pattern for discerning location information in both the NSF and MEMEX use cases our approach required the ability to extract text and metadata from any file type, and then to be able to identify locations within the extracted text. We used the Apache Tika [13] framework to perform text and metadata extraction. Tika extracts text, metadata, and language information from over 1,400 file types defined in the IANA MIME taxonomy, the widely cited classification of files on the Internet. Though not as coarse grained as `fileext.com`, the IANA taxonomy includes rich parent/child information; MIME magic and file name hints that suggest how to best identify file types. Using this taxonomy, Tika integrates all of the necessary parsing libraries to extract text and metadata from *any* file discovered on the WWW. In the rare cases necessary, adding new files is as simple as updating Tika's copy of the IANA database.

2.1 Identifying locations

The next step in our approach uses Apache OpenNLP's `location-ner-en.bin` model to identify locations in text. OpenNLP provides a `NameFinder` class that takes a

Source	Science Topic	Location-related field	Notes
AMD	Agriculture, Atmosphere, Biological Classification, Biosphere, Climate Indicators, Cryosphere, Human Dimensions, Land Surface, Oceans, Paleoclimate, Solid Earth, Terrestrial Hydrosphere, Spectral Engineering, Sun-Earth Interactions	1. Location Keywords field in Metadata 2. Sometimes mentioned in the title and Summary fields 3. Spatial coordinates provided	1. Continent level and Geographic Region 2. Island, sometimes specific 3. Multi-values 4. Not missing in Location Keywords
ACADIS	Agriculture, Atmosphere, Biological Classification, Biosphere, Climate Indicators, Cryosphere, Human Dimensions, Land Surface, Oceans. Paleoclimate, Solid Earth, Terrestrial Hydrosphere	1. Location(s) field in Metadata 2. Location mentioned in Description field, often in the first few sentences or last few sentences. Sometimes missing. 3. Longitude and Latitude provided in the Metadata. 4. Sometimes mentioned in the title	1. Province/State level or Geographic Region 2. Multi-values 3. Not missing in Location

Table 1. Locations mentioned in the NSF TREC-DD-Polar dataset. [10]

token stream and trained location model and that identifies locations in the stream, taking into account preprocessing steps including tokenization, stop-word removal, and stemming. A snippet showing the ease of use of the location name finder is shown below:

```
InputStream modelIn =
new FileInputStream("en-ner-location.bin");
TokenNameFinderModel model =
new TokenNameFinderModel(modelIn);
NameFinderME nameFinder=new NameFinderME(model);
Span nameSpans[] = nameFinder.find(tokens);
```

As shown in Listing 2.1, we can use the `NameFinder` implementation to identify locations given a set of `tokens`. In our `GeoTopicParser` implementation in Apache Tika [2], we leverage the above code snippet as part of our overall extraction algorithm. The algorithm discerns the most frequently occurring location named entity, and selects it as the *best match*. The remaining identified named entities are provided and returned as *alternate locations* from the given textual data and metadata via Tika.

Given a best match location and alternate location set, our approach turns it attention to automatically discerning latitude and longitude coordinates for our locations and we will describe this process in the next section.

3. The Lucene Geo Gazetteer

We used the Geonames.org dataset (referred to as `Geonames` in the rest of the paper) to match location names with latitude and longitude coordinates. `Geonames` contains over 10,000,000 geographical names corresponding to over 7,500,000 unique features. Besides names of places in various languages, data stored include latitude, longitude, elevation, population, administrative subdivision and postal codes. All coordinates use the World Geodetic System 1984 (WGS84) for its coordinate reference system.

The `Geonames` dataset provides useful geographic features for discerning everything from the location in WGS84 coordinates of a particular location, to its location type (country, city, state), to its ISO country code. Geonames also provides a set of *alternate* names for geographic places that can be used to match location queries to those within the data set.

Though a full explanation of the `Geonames` schema[2] is beyond the scope of this paper, we highlight the following attributes as those used primarily in our algorithm.

[2]http://download.geonames.org/export/dump/readme.txt

3.1 Geonames Features

1. *Name* - Legal name of a location.

2. *Alternate names* - All other names by which a location is known. Alternate names provide a CSV of all pronunciations and synonyms for a location.

3. *Latitude* and *Longitude* - Geographical coordinates of a location in WGS84.

4. *Feature class* - A high level bucket for similar locations. This bucket distinguishes continent and countries from cities and villages.

5. *Feature code* - A more granular bucket for locations to identify it as a country, state, capital, city.

6. *Population* - Population of a location.

7. *Country code* - ISO-3166 2-letter country code of a country. e.g. US, CA for United States and Canada.

8. *Admin1 code* - First level of administrative code e.g. A state in USA - e.g., CA for California.

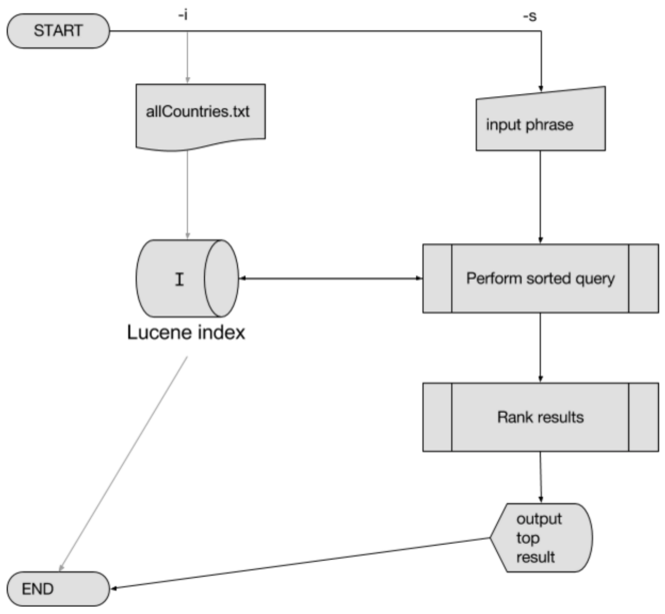
9. *Admin2 code* - Second level of administrative code e.g. A county of US - e.g., 037 for Los Angeles County.

To expose `Geonames` in our automatic location identification process, we developed an approach to make it searchable, by indexing the geo features and information in the Apache Lucene library - the resultant system is called `Lucene Geo Gazetteer`. Lucene provides a full-text inverted index and search implementation and allows for efficient access and retrieval in a local environment. This was a key feature we built upon as we did not want to introduce any service dependencies into our location identification process. We built a command line tool for indexing `Geonames` in Lucene, and then we exposed gazetteer functionality from the command line and from a REST web service for searching after the index is built. The command line tool and REST web service takes an input text phrase and returns locations and their associated latitude and longitude.

The overall approach for the Lucene Geo Gazetteer is shown graphically in Figure 1 and enumerated below.

Indexing We first index all of the entries from geonames.org and its `allCountries.txt` downloadable database to a lucene index through a specialized Java program that is part of our tool. The advantage of allCountries.txt is its coverage of geographical locations from all over the world. This is shown in the upper left of Figure 1.

Figure 1. Overall architecture for the `Lucene Geo Gazetteer`



Searching Our tool provides a search functionality based on provided input phrases that searches the resultant Lucene index and that returns a "best match" (described in this section) for locations found in the index. This process is shown in the middle-left and middle-right of Figure 1.

Ranking We developed a specialized ranking algorithm to sort returned results from `Geonames` based on data such as Feature Class, Feature Code, and other information already described. The ranking provides a mechanism to return a *topN* results from the `Geonames` Lucene index that our approach builds. This is shown in the bottom right of Figure 1.

In the next sub-section we describe how we evolved our ranking algorithm to handle location disambiguation, multi-location matches, and strength of match to ensure accurate location results.

3.2. Evolution of Ranking Algorithm

We went through several iterations to improve the relevancy of the results returned from `Lucene Geo Gazetteer`. Throughout the experiments we came across multiple use cases and we enhanced our process to accommodate those.

The first naive implementation of our algorithm used Edit-Distance [9] as an approach to find matching names,

Input phrase	Location retrieved	Location name in data set
India	India, Vigrestad, Norway	India
Japan	Japan, Montenegro	Japan

Table 2. Sample results with naive algorithm for strings "India" and "Japan"

and alternate name features from `Geonames`. This version depended entirely on string-matching and ignored the other feature elements of a location provided by the `Geonames` data. The fields used in this approach included *Name*, *Alternate Name*, *Latitude* and *Longitude*.

The results for this version of our approach were very low quality, and we tested it on a set of world countries generalized from the MEMEX and NSF Polar CyberInfrastructure dataset. One example illustrating why this approach was ineffective is illustrated in **Table 2**. This happened because there are 74,797 references to the string “India” and 304 references to the string “Japan” in `Geonames`. The country India is stored by it’s official name “Republic of India”. On the other hand, the country Japan was not returned properly due to multiple alternate names stored “An Iapan” ,“An tSeapain” ,“An tSeapaiin” that fell within the edit distance threshold originally employed by our approach. This is illustrated in Table 2.

Based on this experience, we made the following observations about the `Lucene Geo Gazetteer` and its initial approach. First, exact/nearest string matching does not provide accurate relevancy or even a correct interpretation of location results. Second, there are many locations with similar names. For example, contextually though one may believe there is one, or even two (depending on if you live in Southern California and/or Texas) “Pasadena” locations, in fact “Pasadena” has approximately 250 matches in the `Geonames` data. Alternatively “Portland“ another location that, for those on the West Coast of the United States may believe has a single location and meaning in the state of Oregon, would be surprised to hear that “Portland” has 897 location matches within the `Geonames` dataset. Both of these relationships are shown in Figure 2.

Our first attempt to resolve the naive approach expanded our use of `Geonames` features to include *Name*, *Alternate Names*, *Feature Code*[3], *Feature Class* and *Population* as described in Section 3.1. We first split the *Alternate Names* feature by commas, to allow for more precise keyword oriented matches. We followed by developing a sorting approach that sorted by *Country*, then *State* feature classes, then by *City*, then *Village*, then *Spot*, *Building*, and *Farm*, then *Mountain*, *Hill* and *Rock*. We followed sorting by

[3]http://www.geonames.org/export/codes.html

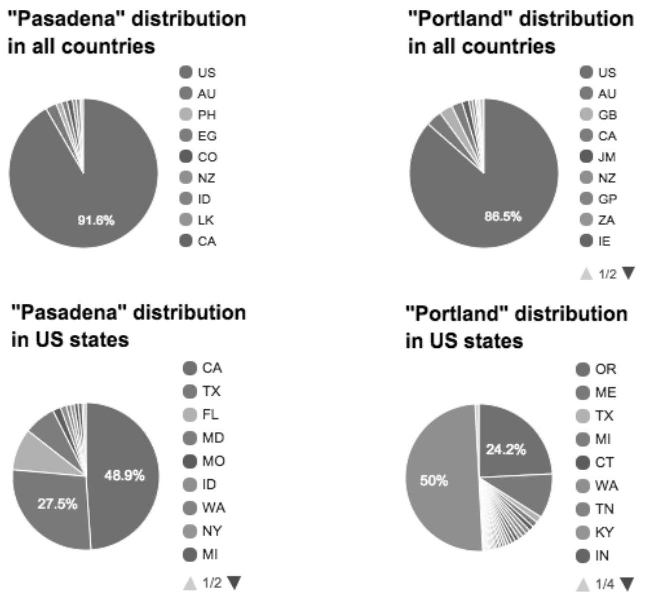


Figure 2. Word "Pasadena" / "Portland" distribution in data set

Parks and *Areas*, then *Stream* then *Lake* feature codes. The preceding sort corresponds to `APSTLH` feature classes. The remaining sorts were `RVU` or *Roads*, *Forests* and by *Undersea*. We also further sorted on feature codes that mapped to the feature classes referenced earlier.

Additional improvements to the algorithm constituted sorting by population in descending order especially those with the same feature class and code. These combined search criteria allowed for a more precise and relevant *topN* result retrieval from the dataset, and we finished up by only performing an edit-distance comparison between the identified geo-locations (recall Section 2.1) and those *topN* sorted results. Based on these improvements, the quality of results were significantly improved, though there were a few lingering issues that decided to address in our project and we will enumerate them below.

Improving disambiguation Edit-distance provided a sound string comparison technique, but as the results became more precise we realized we needed better disambiguation. For example, the Edit distance between “Los Cabos” and “Los Angeles” is 6, however, it is nearly the same value (7) for two very similar locations, e.g., “Los Angeles County” and “Los Angeles”.

Population *Population* is a key field in determining popularity of a place. Results sorted only by population are very relevant and query relevancy using only this feature can be dramatically improved.

Geo Name and alternate name - Overall, *Name* has a different meaning than *Alternate name* in a string matching sense. For example *Name* in `Geonames` is a legal name for a location, e.g., "China", or "People's Republic of China", however *Alternate Name* may contain e.g., "Cayina, Chaina, China, Chine, ... ".

We describe our completed updates to the ranking algorithm based on our initial iterations in the next sub-section.

3.3. The Final Ranking Algorithm

Based on our observations using the `Lucene Geo Gazetteer` and its limitations, we made a series of updates to its ranking algorithm shown in the middle-right of Figure 1. In particular, we evolved our location input matching to perform exact phrase matching for the *Name* field, and to allow partial matches, but to flag them and treat them with less weight. We also only performed edit-distance matching on *Alternate Name* field, allowing such indirect comparisons for a field that matched those expectations in the `Geonames` dataset.

We updated our ranking to score results with more alternate names than others (meaning it has further ability to disambiguate between other locations with similar names). In turn, we also updated the mechanism in which we query the Lucene index we built of `Geonames` data by adding significant weight to *Population* which we have found to be an important discriminator of relevancy to a location query. In addition, we provided a sort feature based on *Feature Code* to handle smaller population data and locations as directed by the user.

The final version of our algorithm searches the Lucene index for a given location name from Apache Tika and Apache OpenNLP and retrieves the top *3N* results from the index. The results are sorted by *Feature Code*, and then those results are subset. The results are then re-scored in the following fashion. We weight the result higher (and thus more *relevant*) if the input location is a sub-phrase in a location *Name* e.g., "Los Angeles" in "Los Angeles County". We assign lower weight for partial matches. We compute the edit distance and find all matched *Alternate Name* locations within a given threshold and join those with the matched *Name* locations. Those matches with more *Alternate Name* locations are given higher weight. We return back the location results and the score to the invoker of the command line `Lucene Geo Gazetteer` or its REST equivalent.

We tested our improved algorithm on a diverse set of 50,000+ locations with countries, states, districts, cities, towns and villages from DARPA MEMEX and from the NSF Polar domains. Our ground-truth data comparisons resulted in a total overall accuracy of 94%.

name	**feature code**	**country code**	**admin1 code**	**popul--ation**
Pasadena	PPL	US	TX	149,043
Pasadena	PPL	US	CA	137,122

Table 3. Example for two optimal results

3.4 Discussion

While the early evaluation from our work is promising it is clear that there remains a mixed view of what defines an optimal result for every location query usually delineated by user preference. For example as indicated previously and now directly in **Table 3** if a user is in Los Angeles she is likely searching for Pasadena, California similarly if user is in Texas she is looking for the Texas equivalent.

Addressing this problem in an optimal fashion is quite difficult. One approach may be to provide N results using a *Count* parameter and then to allow a user to specifically sort through the results using e.g., *Country Code*, *Admin1 Code* and *Admin2 Code*. In addition to these features, our research has suggested that the following information would be useful additions to the `Geonames` dataset. For example, *Popular name of a location* could provide more widely known names e.g. Argentina for "Argentine republic", or Italy for "Repubblica Italiana". This would improve overall results as a user is more likely to query by popular name. In addition, the additional of further geospatial information such as area, and bounding box would provide more precise spatial comparisons not directly enabled currently by `Geonames`.

In addition, the `allCountries.txt` is a huge data set with locations ranging from parks to continents. We are investigating improving our `Lucene Geo Gazetteer` to operate only on bigger land masses by indexing selective feature class or feature codes. For example we may only want to create an index with continents, countries, state, counties, capitals and cities.

4. Conclusion and Future Work

We described an approach for automatically selecting location names from any file type found on the WWW. Text and metadata is first extracted by the Apache Tika framework, which in turn provides the extracted text and metadata to the Apache OpenNLP location entity recognition system. Extracted locations are then provided to a customized Geo Gazetteer built from the Geonames.org dataset, and from there, we have implemented sophisticated information extraction and retrieval techniques to sort and provide more relevant location results. When tested on ground truth data from the DARPA MEMEX and NSF Polar CyberInfrastructure programs, the approach performs with 94% accuracy in location identification.

While the early results from this effort are promising a number of avenues of future work remain unexplored. In particular, we plan on trying to automatically augment Geonames.org with crowd sourced popular place names and with geospatial coordinate data beyond points, e.g., bounding boxes, and other shapes, to allow for more meaningful and spatially directed queries, when combined with location textual data.

Acknowledgments

This work was supported by the DARPA XDATA/Memex program. In addition, the NSF Polar Cyberinfrastructure award numbers PLR-1348450 and PLR-144562 funded a portion of the work. Effort supported in part by JPL, managed by the California Institute of Technology on behalf of NASA.

The authors would like to thank Dr. Mattmann's DR student Yun Li for the initial implementation of Lucene Geo Gazetteer and GeoTopicParser in Tika. In addition, we would like to thank Thamme Gowda N for his implementation of the REST server for Lucene Geo Gazetteer. The authors finally would like to thank Maziyar Boustani and Lauren Wong for their work on MEMEX GeoParser that builds on this effort.

References

[1] GeoNames.org. `http://www.geonames.org/`.

[2] Geotopicparser - apache tika. `http://wiki.apache.org/tika/GeoTopicParser`, 2015.

[3] A. B. Burgess, C. Mattmann, G. Totaro, L. J. McGibbney, and P. Ramirez. Trec dynamic domain: Polar science. *Text Retrieval Conference*, 2015.

[4] A. B. Burgess and C. A. Mattmann. Automatically classifying and interpreting polar datasets with apache tika. In *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pages 863–867. IEEE, 2014.

[5] B. Fortner. Hdf: The hierarchical data format. *Dr. Dobb's journal*, 23(5), 1998.

[6] N. Freed and N. Borenstein. Multipurpose internet mail extensions (mime) part two: Media types. Technical report, rfc 2046, November, 1996.

[7] M. V. John Vogel and J. Pustejovsky. ATLIS: Identifying Locational Information in Text Automatically. `http://www.lrec-conf.org/proceedings/lrec2012/pdf/1022_Paper.pdf`.

[8] J. Kottmann, B. Margulies, G. Ingersoll, I. Drost, J. Kosin, J. Baldridge, T. Goetz, T. Morton, W. Silva, A. Autayeu, et al. Apache opennlp. *Online (May 2011), www. opennlp. apache. org*.

[9] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992.

[10] C. Mattmann. TREC Dynamic Domain Polar Dataset. `http://github.com/chrismattmann/trec-dd-polar/`.

[11] C. Mattmann. Search of the deep and dark web via darpa memex. In *2015 AGU Fall Meeting*. Agu, 2015.

[12] C. Mattmann, Y. Li, M. Sharan, and T. N. Gowda. Lucene geo-gazetteer. `https://github.com/chrismattmann/lucene-geo-gazetteer`, 2015.

[13] C. Mattmann and J. Zitting. *Tika in action*. Manning Publications Co., 2011.

[14] M. McCandless, E. Hatcher, and O. Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[15] R. Rew and G. Davis. Netcdf: an interface for scientific data access. *Computer Graphics and Applications, IEEE*, 10(4):76–82, 1990.

[16] M. Sharan. Visualization of final version. `https://goo.gl/P9XysK`, 2015. [Online; accessed 17-Mar-2016].

[17] M. Sharan. Visualization of naive version. `https://goo.gl/Mt3qgb`, 2015. [Online; accessed 17-Mar-2016].

[18] B. Vastag. Exabytes: Documenting the digital age and huge growth in computing capacity. *Washington Post*, 2011.

[19] V. Woollaston. Revealed, what happens in just one minute on the internet: 216,000 photos posted, 278,000 tweets and 1.8m facebook likes. *Daily Mail - UK*, July 2013.

[20] Z. X. Xueying Zhang, Guonian Lv and Y. Sun. EXTRACTION AND VISUALIZATION OF GEOGRAPHICAL NAMES IN TEXT. `http://icaci.org/files/documents/ICC_proceedings/ICC2009/html/nonref/12_8.pdf`.