To Learn or to Rule: Two Approaches for Extracting Geographical Information from Unstructured Text

Philipp Katz **Alexander Schill**

Dresden University of Technology
Faculty of Computer Science
Institute of Systems Architecture
01072 Dresden, Germany
Email: philipp.katz@tu-dresden.de

Abstract

Geographical data plays an important role on the Web: recent search engine statistics regularly confirm that a growing number of search queries have a locale context or contain terms referring to locations. Assessing geographical relevance for Web pages and text documents requires information extraction techniques for recognizing and disambiguating geographical entities from unstructured text. We present a new corpus for evaluation purposes, which we make publicly available for research, describe two approaches for extracting geographical entities from English text—one based on heuristics, the other relying on machine learning techniques—and perform an extensive discussion of those two approaches. Furthermore, we compare our approach to other publicly available location extraction services. Our results show, that the presented approaches outperform current state of the art systems.

Keywords: Toponym Resolution, Toponym Recognition, Toponym Disambiguation, Machine Learning, Feature Mining, Dataset

1 Introduction

According to recent statistics, between 30 and 40 % of the users' queries at Google are now related to physical places (Parsons 2012). Therefore, correctly recognizing and extracting geographic information from unstructured text can be considered a crucial step for offering more appropriate answers to users' information needs. Domains where geographic information plays an important role are, for example, the daily news; methods for searching and organizing news articles can greatly benefit from place information, as it is one of the fundamental parts of the *Five Ws* (Who, What, When, Where, Why) employed in journalism to describe events. As a further example, consider advertising: In *Geotargeting*, geographic information extracted from available data can be used to deliver more individual and context relevant content to the user.

The roots of geographical information extraction lie in Named Entity Recognition (NER) as it was defined by the Message Understanding Conference 6 in 1996 (Grishman & Sundheim 1996). However, general NER usually neglects spatial properties in favor of recognizing a broad range of different entity types (e. g. in MUC-6, the types "Organization", "Person", "Location", "Date", "Time", "Money", "Percent" were used). Later, dedicated approaches focussed explicitly on extracting geographic data from text and associating extracted location references with models of the real world by providing spatial and/or topological properties such as coordinates or administrative/part-of relations.

The task of extracting geographical information from text can be divided into the following three disciplines: *Toponym Recognition* (TR), *Toponym Disambiguation*[1] (TD) and *Scope Resolution* (SR). TR identifies and marks occurring toponyms in a text and is therefore closely related to NER. A central challenge lies in the so called "geo/non-geo ambiguity" (Amitay et al. 2004). Considering the sentence "Mary is in Turkmenistan.", it is unclear without further background information, whether "Mary" refers to a person or to the city which is located in the south east of the country. The process of TD associates identified toponyms with entries in a database serving as a so called gazetteer. Here, we face the problem of the "geo/geo ambiguity" (Amitay et al. 2004). Consider the sentence "San Antonio is a place in California.". While looking up the term "San Antonio" in a gazetteer would yield matches all over the world, using the determiner "California" as filter, the number of potential places can be greatly reduced, although there are still multiple places called "San Antonio" in California.

SR on the other hand identifies a spatial scope for a document as a whole, summarizing all geographical evidence to one appropriate abstraction. While the presented approach can be further used as foundation for SR, the focus of this work is TR and TD.

During our research, we found that existing datasets for geographic information extraction are usually not publicly available, limiting the possibilities to compare different approaches with each other. Thus, the first contribution of this work is a novel dataset with NER-style type annotations for locations and associated geo coordinates, which we make freely available for research purposes on the research platform Areca. The creation and properties of this dataset, called "TUD-Loc-2013" hereupon, will be described in detail in Section 3. As a second contribution, we present two new approaches for combined TR and TD: A heuristic approach relying on several rules which will be described in Section 4.2, and an approach using machine learning (ML) techniques, based on a plethora of extracted features, which will be discussed in detail in Section 4.3. To the current

[1] Others, such as Leidner (2006), Lieberman & Samet (2012) refer to this task as "Toponym Resolving" or "Resolution".

state of the art, none of the existing approaches as presented in Section 2 has applied ML with such an extensive feature set for this task. Section 5 presents our gazetteer, which is aggregated from multiple freely available location sources. In Section 6, we outline our experiments for fine-tuning the approaches. In particular, we review the features which serve as input for the machine learning approach and determine, which of them are actually valuable for the task. In Section 7 we compare our approaches to other services for extracting locations from unstructured text.

2 Related Work

This section describes related work for TR and TD. General all-purpose NER approaches which perform only TR, but do not associate recognized toponyms to geographic representations are not considered here, neither do we present approaches for SR such as Andogah (2010) or Wing & Baldridge (2011).

One of the first notable works in this area can be attributed to Smith & Crane (2001), who did TR and TD for a digital library with historical content. After a rule-based identification and filtering of toponym candidates, their approach calculates a centroid coordinate for all location candidates and continuously elmiminates those which are more than a specified distance away from this centroid.

Li et al. (2002, 2003) build a weighted graph of all location candidates in a text. The edge weights are determined using a set of rules which rely on topological properties between the represented locations beside some intrinsic text metrics. Using Kruskal's algorithm, a Maximum Weight Spanning Tree is calculated from which the final disambiguation is derived.

Rauch et al. (2003) describe a confidence-based mechanism which, on the one hand relies on intrinsic textual properties, such as the proximity between two toponyms within the text and the actual geographic proximities of their potential locations. On the other hand, for locations with same names, higher confidence values are assigned to those with larger population figures.

Smith & Mann (2003) employ a Naïve Bayes classifier for a simplified TD task, where the aim is to recover the correct U. S. state or country for a given text. The classifier is trained using phrases from texts, which disambiguate place names by giving explicit cues, such as in "[. . .] Nashville, Tennessee [. . .]", and relies on text features (unfortunately, the work gives no deeper information about the feature types they use, e. g. n-grams, tokens, etc.). However, their results yield in only minimal improvements over a baseline which simply assumes the most frequently occurring location.

Similar to Rauch et al. (2003), Amitay et al. (2004) assign confidence values to locations based on some heuristics. Additionally, they address the problem of geo/non-geo ambiguities with a large curated corpus of over 1 million Web pages. Location names from their gazetteer, which do not occur as proper nouns in the corpus frequently, or where the number of mentions in the corpus is strongly disproportional to the population of the place, are considered as non-locations, unless explicit evidence is given in the text.

Leidner (2007) presents an algorithm for toponym resolution using a spatial minimality assumption. Besides the heuristic to resolve each toponym to a country, if such exists, and exploiting explicitly given disambiguations in text, a cross product is computed of all locations for remaining unresolved toponyms. From all potential combinations, each containing one potential location for the toponyms given in the text, the combination which spans the smallest area is selected.

Da Graça Martins (2008) use a set of manually created contextual rules for TR. Furthermore, an exclusion list is used to remove common terms which are not usually locations. The TD also makes use of the context rules and further applies a set of heuristics which exploit the topological relations between pairs of potential locations.

Buscaldi & Rosso (2008) employ classical methods for word-sense disambiguation and rely on WordNet to recognize toponyms. They note the poor coverage of WordNet in regards to geographical information compared to classical gazetteers.

Lieberman & Samet (2012) are the first to evaluate a machine learning-based TD approach. They present seven features, two of which are extracted from a so called "adaptive context". The adaptive context is characterized by a window breath and depth. The first denotes a context of a specified number of toponyms before and after the currently considered toponym, while the depth limits the number of potential location candidates to consider for each toponym. These two figures are motivated by the requirement to allow for a fast TD computation. Using labeled data, a classifier is trained, which carries out a classification for each potential location assignment to a toponym to be either correct or incorrect.

While most of the aforementioned works rely on heuristics, only the more recent approach of Lieberman & Samet (2012) successfully takes up the idea of a feature-based machine learning technique. However, they rely on a comparatively small feature set and only employ them for the TD phase. In this work, in contrast, we are going present a comprehensive set of different features, which we will then evaluate for a machine learning-based, combined TR and TD approach. In addition, we will draw a comparison with our new heuristic-based method.

3 Datasets

Despite the growing attention of the research community during the last years in toponym recognition, the lack of publicly available datasets makes it difficult to compare different approaches to each other. In the following, we will shortly give an overview over datasets which have been employed in the past and outline our motivations to create the novel, freely available "TUD-Loc-2013" dataset.

3.1 Existing Datasets

"GeoSemCor", as presented by Buscaldi & Rosso (2008), is a freely available dataset, where Toponyms have been annotated with WordNet senses. It contains only annotations for the relatively popular toponyms which exist in WordNet and has no geographical referents. The same applies to "CLIR-WSD"[2].

"TR-ConLL", which was presented in Leidner (2006) is based on English news texts from the Reuters CoNLL corpus. It contains 946 documents with 6,980 toponym instances, of which 1,299 are unique. The dataset can be purchased from the author and costs 550 US$ for an academic license.

The "ACE 2005 English SpatialML Annotations"[3] consists of 428 documents from a broad range of different sources (news, blogs, newsgroups). The dataset is

[2] http://ixa2.si.ehu.es/clirwsd/

[3] http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2011T02

not publicly available; non-members of the Linguistic Data Consortium pay 500/1,000 US$.

The "Local-Global Lexicon" (LGL) corpus from Lieberman et al. (2010) was obtained from 78 newspapers with a mainly local focus, yielding in 588 documents. The sources were selected with an explicit focus on ambiguities (for example, containing the cities Paris in Texas, Tennesse, and Illinois). The dataset therefore seems well suited for evaluating location extraction in a local domain, but does obviously not represent realistic properties. LGL is not publically available, but the author kindly supplied us with the corpus. During a thorough inspection, however, we noticed several issues and inaccuracies concerning the annotations: Plenty of locations in the dataset have not been marked[4] or for many annotations, more specific locations exist[5]. Besides, demonyms and adjectives are marked as locations. We have seen, that these drawbacks favor extraction approaches with low recall and penalize approaches with very fine-grained toponym resolution. That is why we will not include LGL in our evaluation given in Section 7.

"TR-CLEF" and "TR-RNW" are manually anotated corpora used for evaluations in Andogah (2010). They are not published however and we never received a reply from the author to our inquiry.

3.2 TUD-Loc-2013

Motivated by our experiences as described in Section 3.1, we will describe the creation of "TUD-Loc-2013"; a novel dataset for evaluating TR and TD approaches. The dataset described here is published on the open research platform Areca[6] to increase transparency and to allow future research to be compared on this publicly available dataset.

The dataset consists of 152 English text documents retrieved from different URLs. An index file within the dataset package gives the original URLs from which the pages were obtained. We focused on content-oriented pages such as news and blog articles, but excluded start pages which combine multiple topics. The main text content was manually extracted, removing elements such as banners, navigation menus, comments, headers, and footers.

	Total		Unique	
Type	**#**	**%**	**#**	**%**
CONTINENT	72	1.89	6	0.43
COUNTRY	1,486	38.96	147	10.49
CITY	1,031	27.03	401	28.62
UNIT	242	6.35	131	9.35
REGION	139	3.64	83	5.92
LANDMARK	281	7.37	183	13.06
POI	454	11.90	355	25.34
STREET	55	1.44	45	3.21
STREETNR	37	0.97	33	2.36
ZIP	17	0.45	17	1.21
# All	3.814		1.401	

Table 1: Counts of annotations in TUD-Loc-2013

We initially defined ten location types for annotation as depicted in Table 1. We tried to make different location types clearly distinguishable and wanted to avoid a too broad type variety. While the first three of the given types should be self-explanatory, we want to stress the difference between `UNIT` and `REGION`; the first refers to administrative entities, such as federal states, counties or cities' districts (e. g. "California", "Bavaria", or "Manhattan"). The latter, `REGION`, on the other hand is used to designate areas without political or administrative meaning (e. g. "Midwest"). While locations annotated as `LANDMARK` refer to geographic entities, such as rivers, lakes, valleys, or mountains (e. g. "Rocky Mountains"), the type `POI`[7] indicates buildings (e. g. "Stanford University" or "Tahrir Square").

The annotation of the extracted texts was done manually in XML style. This means, that relevant parts of the text were surrounded by tags denoting the appropriate types. Additionally, we allowed the attribute `role="main"` once per document, indicating the document's geographic scope (relevant for SR tasks, see Section 1). The following paragraph shows an example snippet from the dataset:

```
Tiny <LANDMARK>Heir Island</LANDMARK> -- one of the
many isles that are scattered across County
<CITY>Cork</CITY>'s <LANDMARK>Roaring Water
Bay</LANDMARK> in <COUNTRY
role="main">Ireland</COUNTRY>'s southwest -- is one
of the country's go-to gourmet spots.  So you will
need to book months in advance to dine at <POI>Island
Cottage</POI>, a restaurant run by the
husband-and-wife team John Desmond and Ellmary Fenton.
[...]
```

In a second step, annotations were associated with actual locations. Through a dedicated Web-based annotation app, we allowed to query GeoNames[8] and—as a fallback—the Google Geocoding API[9] with the annotations' values. All results were displayed as markers on a map, including additional properties such as type, population, etc. and were then manually selected. As not all values can be found directly, we also provided the possibility to modify the queries (e. g. the term "Atlantic" needs to be corrected to "Atlantic Ocean" to give any results). Locations, which could not be found could be marked as "non resolved" explicitly. The result is a separate CSV file with pointers to the annotated text files (filename, running index and character offset of annotation), coordinates and source-specific identifiers. The following lines give an excerpt of the CSV file:

```
docId;idx;offset;latitude;longitude;sourceId
text1.txt;0;0;53.00000;-8.00000;geonames:2963597
text1.txt;1;28;53.00000;-8.00000;geonames:2963597
text1.txt;2;399;51.49475;-9.43960;google
text1.txt;3;469;51.96667;-8.58333;geonames:2965139
text1.txt;4;476;;;
text1.txt;5;497;53.00000;-8.00000;geonames:2963597
text1.txt;6;619;;;
text1.txt;7;755;51.51077;-9.42505;google [...]
```

From the given 3,814 annotations, 3,452 were manually disambiguated (90.51 %). The remaining non-disambiguated locations are mostly of type `POI`; here, only 50,88 % of the annotations could be assigned with coordinates. This is due to the fact, that the dataset contains several little-known locations such as restaurants, etc. which could not be found in the considered databases. Figure 1 shows the distribution of the disambiguated locations on a map.

[4] In doc. 38765806 for example, 20 annotations are present in the dataset, but we counted 44 toponyms.

[5] In doc. 38543488 for example, in the phrase "[...] building permit for the new Woodstock General Hospital [...]", the term "Woodstock" is marked, but we feel that "Woodstock General Hospital" is the more accurate location.

[6] `http://areca.co/21/TUD-Loc-2013-location-extraction-and-toponym-disambiguation-dataset`

[7] Point of Interest

[8] `http://www.geonames.org`

[9] `https://developers.google.com/maps/documentation/geocoding/`

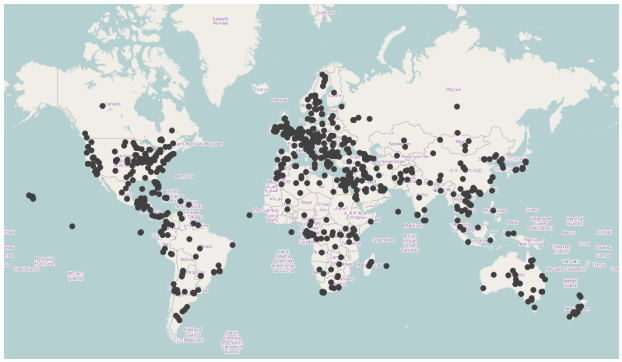

Figure 1: Coordinate distribution in TUD-Loc-2013

The final dataset is split in the three disjoint sets *training* (40 %), *validation* (20 %) and *test* (40 %).

4 Approaches

In the following Sections 4.2 and 4.3, our two strategies for TR and TD are described. The common processing steps to both of the strategies, which include candidate extraction, filtering and preprocessing, are described as follows.

4.1 Preprocessing

In contrast to various other approaches, which rely heavily on a full-fledged preprocessing pipeline, including PoS tagging, deep parsing or NER, we only use very basic mechanisms for TR as described below. We focus on correctly detecting entity candidates of all types (i. e. also potential non-locations) and only filter out those ones, where we can be sure that they do not represent a toponym. Our assumption is, that wrongly classified candidates can be better removed in the TD phase, where we can apply more knowledge such as the gazetteer and information about further candidates in the document. All kinds of lingustic processing on the other hand, especially NER, are strongly domain-specific and introduce additional chances for errors under suboptimal conditions.

Candidate extraction takes place using a rule-based tagger, which marks sequences of capitalized expressions in the text. Several exceptions are applied to correctly annotate term spans containing lowercased prepositions or special characters (such as in "United States of America", "Rue de Rivoli", or "Grand Tradition Estate & Gardens"). This approach guarantees high recall and was successfully applied by Urbansky (2012).

A problem which arises from the rule-based tagging is that tokens at the beginning of sentences are always considered as candidates because of their capitalization. Given the sentence "Tiny Heir Island is one of the country's go-to gourmet spots.", our tagger extracts the candidate "Tiny Heir Island", although "Tiny" represents an attribute for the actual location. A pre-generated case dictionary is used to remove or correct candidates at sentence beginnings. The idea has been described by Millan et al. (2008). The case dictionary[10] consists of a list of tokens with their occurrence frequencies as uppercase and lowercase variant. In case, a token occurs clearly more frequent in lowercase form, we remove it or correct the candidate's offset (so that in the given example, we would end up with the correct form "Heir Island").

Our experience shows, that many incorrectly extracted locations are actually person names. Therefore, we remove *sure*-negative candidates, using a well curated set of person-centric prefixes such as "Mr.", "Minister", "Officer", etc. in a first step. In a second pass, we classify *potentially* negative candidates using candidates' text contexts. Contexts are surrounding tokens before or after a candidate and give clues about its type. For example, in the sentence "Georgia attended the conference", we can conclude from the suffix "attended", that the entity is most likely a person, whereas in the case of "Georgia president concedes election defeat", the suffix "president" gives a strong clue, that the preceding candidate is a location. We have accumulated a massive amount of location and person specific texts building on the foundations as described in Urbansky (2012) to extract a corpus of characteristic contexts for both types. For a list of 800 manually compiled seed entities for each type "person" and "location", we queried Bing[11] and obtained at most 100 URLs per seed, yielding in 29,642 HTML pages for persons and 33,454 pages for locations. We tried to extract the main content block of each page using the Palladian toolkit (Urbansky et al. 2012) and filtered texts under 100 characters and short fragments within texts. The final context dataset consists fo 126,377 person entities and 184,841 location entities. Contexts for persons serve as negative, contexts for locations as positive indicators to build a context dictionary which we use for lookup during classification. To avoid misclassifications and thus decrease recall, the dictionary is created using a very conservative strategy: Only those contexts, where probability is over 90 % of being one of either type are incorporated into the dictionary[12]. We experimentally evaluated different context token sizes and a fuzzy matching (whether a term occurs within a window around the entity candidate), but found, that a fixed lookbehind and ahead of one token provides the most reliable classification results. We assume a "one sense per discourse" (Gale et al. 1992) and thus consolidate context classifications of identical entities within the document.

In contrast to the first filtering pass, we make no instant decision whether to remove the candidate here. The rationale behind this deferred commitment strategy is, that a distinction between location and person type is not perfect and complicated by common figures of speech such as metonymy. For example, for the phrase "U.S. says Rwanda aids Congo rebels", our context classifier would cleary label "U.S." as being a person. By postponing the decision whether to drop or to keep the candidate in question to the TD phase, we can make use of the gazetteer information to apply appropriate exceptions in case of prominent locations such as countries or capitals.

The last step in the preprocessing phase is the lookup in our gazetteer (see Section 5). For each unique annotation value a_n, we query our database to retrieve a set L_n of potential location candidates. The TD strategies as described below take a list of all annotations $A = \{a_1, \ldots, a_n\}$ with their corresponding locations $AL = \{L_1, \ldots, L_n\}$. Then, per annotation a_n, either one location candidate $l \in L_n$ is selected, or the annotation is discarded as being a "non-location" by the TD.

[10] We created our case dictionary from English Wikipedia articles, consisting of approximately 91,000 tokens.

[11] `http://www.bing.com`

[12] The context dictionary used within this work consists of 318 prefix and suffix contexts for the type "person" or "location".

4.2 Heuristic TD

As a first step, the heuristic eliminates unlikely annotations by applying the following rules: Those annotations, which were marked as being likely of type "person" are removed, in case they do not have a location candidate which is of type `CONTINENT` or `COUNTRY`, or where the population is above a *unlikelyPopulationThreshold*.

Subsequently, the approach makes use of a concept which we call "anchor locations"[13]. The underlying idea is to first only extract those locations, where we can guarantee high precision and use them as reference points in a second extraction step. The mechanism for extracting anchor locations is outlined in Figure 2. First, we assume those locations as anchors, which are either of type `CONTINENT` or `COUNTRY` or those which exceed a high population count as specified with *anchorPopulationThreshold*. To extract further anchor locations, we employ the following criteria: We create groups of locations with equal names and determine the largest distance[14] between each pair in the group. In case, the largest distance is below *sameDistanceThreshold* (which is set to a two-digit value in kilometers), we suppose multiple location candidates to denote the same place[15]. Locations complying to this condition and having either a population of more than *lowerPopulationThreshold*, or a distinctive name consisting of more than one token (e. g. "Santa Catarina Federal University") as defined by *tokenThreshold* are added to the set of anchor locations. We have seen, that the probability for geo/non-geo ambiguities strongly decreases for terms with two or more tokens.

```
funct getAnchors(L) ≡
  Anchors := {}
  for l in L do
    if type(l) ∈ {CONTINENT, COUNTRY} ∨
      population(l) ≥ anchorPopulationThreshold
      then Anchors ← l; fi
  end
  for g in groupByName(L) do
    if largestDistance(g) ≤ sameDistanceThreshold
      then
        l := getBiggestLocation(g);
        p := population(l);
        t := |tokenize(name(l))|;
        if p ≥ lowerPopulationThreshold ∨
          t ≥ tokenThreshold
          then Anchors ← l; fi
    fi
  end
  return Anchors.
```

Figure 2: Algorithm for extracting anchor locations

In case we could not determine any anchor locations using the given strategy, we use a stepwise convergence approach comparable to a lasso, which is increasingly tightened. Figure 3 shows the progress, where we continuously remove the most outlying location from the center point of a given set. The idea is adopted from Smith & Crane (2001), but we stop the convergence, as soon as the maximum distance between any pair in the remaining set is below a specified *lassoDistanceThreshold*. We only take the locations remaining in the set as anchors, if it contains at least two differently named candidates. This way, we can avoid converges into the wrong direction. In case, still no anchor could be established, we simply select the location with the highest population from the candidate set as a fallback.

```
funct getLasso(L) ≡
  Lasso ← L
  while |Lasso| > 1 do
      maxDistance := 0;
      maxDistanceLocation := null;
      midpoint = midpoint(Lasso);
      for l in Lasso do
          distance := distance(midpoint, l);
          if distance > maxDistance
            then
                maxDistance := distance;
                maxDistanceLocation := l; fi
      end
      if maxDistance < lassoDistanceThreshold
        then break; fi
      Lasso := Lasso \ maxDistanceLocation;
  end
  if |groupByNames(Lasso)| ≤ 1
    then return ∅;
    else return Lasso; fi.
```

Figure 3: Algorithm for extracting lasso locations

The identified anchor locations are used as reference points in the final disambiguation phase. For all remaining location candidates, which are not in the set of anchor locations, we check the spatial distance between the candidate and all anchors. Locations either falling below a *anchorDistanceThreshold* or being child of a given anchor location and exceeding a *lowerPopulationThreshold* are added to the final result set. In case multiple location candidates with the same name fulfill the given criteria, we select the one with the biggest population, or—in case the locations are in a hierarchy—the deepest, i. e. the most specific one.

4.3 Machine Learning TD

The second approach is based on the findings of the heuristic. We have experienced, that adding more rules to further improve the approach described in Section 4.2 becomes increasingly complicated and bears the risk of overfitting the algorithm. We therefore present a more flexible approach in this section using machine learning mechanisms. It relies on a number of features and a classifier which is trained using manually annotated and disambiguated training documents. As initally outlined, the classifier is used to perform a binary classification for each location candidate for an annotation. The probability value assigned by the classifier is used to rank the location candidates. In case the probability exceeds a specified *probabilityThreshold*, we assign the highest ranked candidate location to the annotation, elsewise we discard all candidates, i. e. we discard the annotation as "non-location". Obviously, the probability threshold allows to adjust the results of the approach into a more precision- or recall-oriented direction.

Table 2 gives an overview over the multitude of features we extract for the classification. We give a general motivation for those features in the following and describe selected features in more detail. The

[13] Other approaches such as Rauch et al. (2003) or Lieberman et al. (2010) also use the term "anchor", their definitions are different, however.

[14] All distance calculations in this work use the haversine function (Sinnott 1984), which denotes the shortest distance between two points on an idealized sphere with a radius of $r = 6,371$ km.

[15] We have, for example, multiple entries for the location "Armstrong Atlantic State University" in our gazetteer, which is due to the fact, that multiple facilities exist. Still, they lie close together, so we treat the locations as one.

Feature Name	Type	Description
Annotation Features		
`numCharacters`	num.	Number of characters in name
`numTokens`	num.	Number of tokens in name
`acronym`	bin.	Name is acronym, e. g. "USA", "U.A.E.", etc.
`stopword`	bin.	Name is on stopword list, e. g. "Or"
`caseSignature`	nom.	Upper/lowercase signature, e. g. "Aa Aa" for "New York"
`containsMarker(M)`	bin.	Name contains marker token `M`, such as "city", "mountain", etc.
Text Features		
`count`	num.	Occurrence count in text
`frequency`	num.	Count, normalized by highest occurence count
Corpus Features		
`unlikelyCandidate`	bin.	Annotation was classified as being unlikely a location during preprocessing
Gazetteer Features		
`locationType`	nom.	Type of location (see Table 1)
`population`	num.	Population count of location
`populationMagnitude`	num.	Order of magnitude of population
`populationNorm`	num.	Population, normalized by highest population of location candidates
`hierarchyDepth`	num.	Depth of location in topologic hierarchy
`nameAmbiguity`	num.	Occurrence count of name in gazetteer, calculated as $1 / \mid sameNameLocations \mid$
`leaf`	bin.	Location has no child with same name
`nameDiversity`	num.	Diversity of alternative names for location, calculated as $1 / \mid namesForLocation \mid$
`geoDiversity`	num.	Spatial distribution of locations with given name
Text and Gazetteer Features		
`contains(p|a|s|c|d)`	bin.	Text contains parent/ancestor/sibling/child/descendant location as candidate
`num(a|s|c|d)`	num.	Number of ancestor/sibling/child/descendant location candidates in text
`numLocIn(R)`	num.	Number of location candidates in text within distance `R`
`distLoc(P)`	num.	Minimum distance to other locations with minimum population `P`
`populationIn(R)`	num.	Sum of population count of other locations within distance `R`
`locSentence(R)`	bin.	Location with maximum distance `R` occurs in same sentence
`uniqueLocIn(R)`	bin.	Location has a uniquely named location in maximum distance `R`

Legend: bin. = binary, nom. = nominal, num. = numeric feature

Table 2: Features for machine learning TD

types of features can be grouped in different categories describing the origin from which they were extracted. *Annotation Features* are directly extracted from the candidates' text values and describe simple string properties. The `caseSignature` describes the upper/lower case combination of a candidate. The motivation behind this feature is, that specific location types have characteristic upper/lowercase mixtures (such as "University of California", "Isle of Man" which case signature is "Aa a Aa"), whereas other capitalization variants such as "McDonald" or "InterCity" (case signature "AaAa") might indicate a non-location. `containsMarker(M)` can be determined for a predefined set of indicative tokens, which give a strong clue that the current candidate is a location, such as "city", "river", "mountain", "university" etc. *Text Features* are extracted by regarding the whole document's text. `count` and `frequency` denote, how often an annotation occurs within the text. The *Corpus Feature* `unlikelyCandidate` is determined using the contexts as described in Section 4.1. *Gazetteer Features* are retrieved from the location database. We incorporate the `locationType` so that the classifier can potentially adjust its decisions to different types of locations. The `nameAmbiguity` signifies the number of locations with the respective name in the database; the more locations exist, the bigger the chance for misclassifications. `geoDiversity` follows a similar motivation, but here we measure, how widespread locations with identical names are spatially; we consider the chance for a misclassification higher, in case potential candidates are scattered throughout the whole world. In contrast, the risk should be smaller, in case the potential candidates are close to each other. *Text and Gazetteer Features* combine properties from the text with properties from the database. The `contains` feature signifies, whether location candidates with a specific topologic relation such as "parent", "ancestor", etc. occur within the text. The `num` feature gives the counts of topologically related location candidates. Thus, we can indicate that a specific location is disambiguated through a superior instance within the text, such as "Houston, Texas", or we have an enumeration pattern with locations of similar types, such as in "Stuttgart, Frankfurt, Munich". The remaining features describe spatial proximities to other candidates and thus exploit the fact, that usually, spatially related locations occur together. For example, `numLocIn(R)` signifies the number of location candidates within a maximum distance `R` to the considered location candidate occurring in the text. Note, that some of the employed features can be parametrized and thus replicated and evaluated with different configurations, which we will discuss within Section 6.

To train our classifier, we depend on manually annotated and disambiguated training documents, such as the TUD-Loc-2013 as presented in Section 3. We also use the classifier to detect non-locations, this means, that all toponyms in the texts must be annotated, as the non-annotated entities are considered negative examples for training. Figure 4 shows the training process. First, the annotated training documents are split into their original text and a set of annotations. We then extract potential location candidates using the mechanisms as described in Section 4.1. In a first step, we extract annotation, text and corpus specific features. For each toponym candidate we perform a gazetteer lookup to retrieve potential location candidates and add gazetteer specific features. The combination of location candidate extracted from text, its features and the location information from the gazetteer forms an instance for the classifier. Each

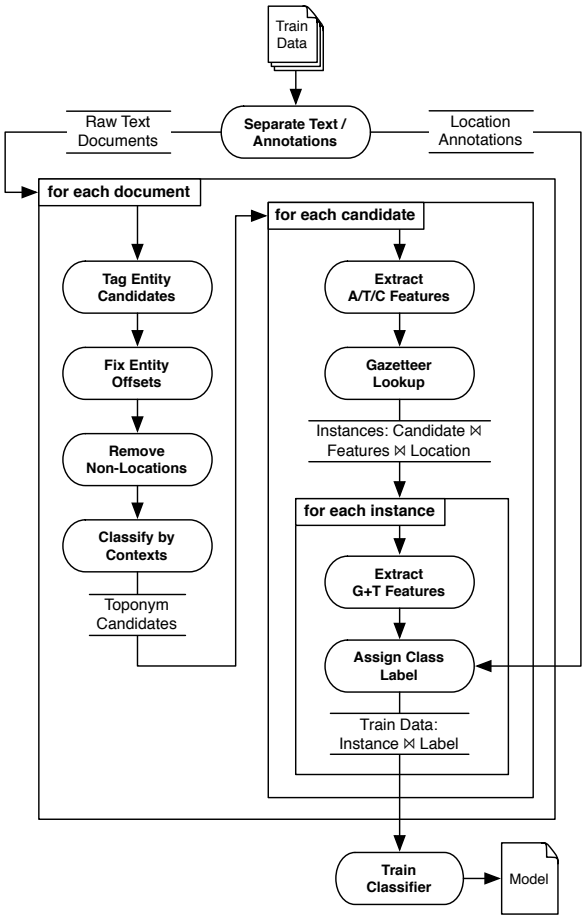

Figure 4: Training process of the machine learning-based approach

instance is checked against the manually assigned location annotations from the training data. We mark those instances as positive samples, which have a corresponding location of same type, same name and a small distance. The remaining instances are marked as negative samples for training.

We make no limitations on the actual classification algorithm, as long as it supports numeric and nominal input features and provides a probability value with its classification output. Our implementation relies on a Bagging classifier (Breiman 1996) which creates multiple decision trees using bootstrap sampling of the training data. The classification is carried out by letting all decision trees vote and taking the portion of each result class as its probability. Bagging decision trees avoids the problem of overfitting and improves classification accuracy compared to a single tree.

The learned model can then be used for classification. The preprocessing and feature extraction steps are identical to those employed during the training phase. All instances are classified, resulting in probability value of being the correct location. Annotations, where all candidates' probability values are below a *probabilityThreshold* are discarded as "non-locations", for annotations where the threshold is exceeded, the candidate with the highest probability is taken as result.

4.4 Postprocessing

The postprocessing phase, which is carried out for the heuristic and the machine learning method as well, extracts location types which we do not cover through our gazetteer (yet): `STREET`, `STREETNR`, and `ZIP`. Our current approach is very rudimentary and involves great potential for future improvements. We start by extracting street names using a set of prefix and suffix rules (such as "*street", "*road", "rue *", etc.). We then try to match street numbers occurring before or after those street names. Similarly, we proceed for ZIP codes, which are searched right before or after location entities marked as `CITY`. A disambiguation of entities of the three types is not carried out currently.

5 Gazetteer

Our gazetteer has been aggregated from different sources and currently consists of circa 9.2 million locations. While other approaches employ only comparatively small gazetteer databases, our aim is to achieve a high recall from our gazetteer and guarantee precision through our algorithms. For each entry, we keep the following information: unique identifier, primary name, alternative names (a list of alternative names for the location, optionally including a language), a type (see Figure 1), latitude and longitude coordinates, population count if applicable and the topologic hierarchy (expressing the list of parent locations within the database, such as "Federal Republic of Germany → Europe → Earth").

While the major portion (8.5 million entries) of our data comes from GeoNames, we have further enriched our database from the following sources: The dataset from HotelsBase[16] provides about 500,000 hotels, protectedplanet.net[17] contributes about 200,000 protected areas.

To further complement our database, we extract locations from the English Wikipedia[18]. Using the Palladian toolkit's MediaWiki parser (Urbansky et al. 2012), we are able to extract 500,000 entries. Similar to universal information extraction approaches such as DBPedia[19], we therefore rely on so called infoboxes (see Figure 5), table-like templates which are used to describe entities of different types in a standardized manner. We use a manually created mapping between infobox types and our own location types to filter relevant pages (for example, the infobox in Figure 5 is of type `protected area`, which is mapped to the type `POI` in our schema) and add those pages to our database, which provide geographic coordinates.

As a further step, we tried to exploit Wikipedia-internal redirects to obtain alternative names for the extracted locations (for example, when trying to access the Wikipedia article "Alcatraz", one is redirected to "Alcatraz Island"). However, we came to the conclusion, that often very obscure redirects exist, which are not in general language use and therefore degrade the quality of our database. As an alternative for future improvement, we suggest to only extract those alternative names, which are explicitly mentioned and highlighted in an article's introduction.

As we perform no explicit deduplication, we have no exact figures on how many previously unknown locations we actually retrieve through the additional sources to GeoNames, but in our experiments we achieved a noticable recognition improvement. Our

[16] `http://www.hotelsbase.org`
[17] `http://protectedplanet.net`
[18] `http://en.wikipedia.org/wiki/Main_Page`
[19] `http://dbpedia.org/About`

Figure 5: Infobox on the English Wikipedia article "Alcatraz Island" providing geographic coordinates

extraction mechanisms can cope with semantically duplicated location data, so eliminating them is not important for us, but could be achieved using record linkage strategies.

6 Experiments

In this section, we will present our findings during the optimization of both presented approaches. The heuristic approach (see Section 4.2) can be fine-tuned using the presented threshold values. For the machine learning approach (see Section 4.3), we presented a set of features which can be used for the classifier. However, not all of those features might be necessary or useful, therefore we are presenting a backward feature elimination to narrow down our full feature set to a reduced necessary subset. We use QuickDT[20] with its Bagging implementation for the machine learning-based approach.

We use precisio, recall, and F1 measure as harmonic mean in the evaluation. First, we evaluate the TR and classification results following the "MUC" evaluation scheme (two dimensional evaluation with separate scoring for correct type and correct boundaries, giving one point for each, and two points in case both were identified correctly). For the TD, we evaluate the spatial distance between the location given in the dataset and the location given by the extractor and assume a correct disambiguation in case the distance is below 100 km. In the following, we also give precision, recall and F1, denoted as "Geo". We do not use a scoring based on actual distances' values such as RMSE[21], as this would pose a disadvantage to other approaches using different gazetteer data which will be compared in Section 7. The TD is only evaluated for locations of the types `CITY` and `POI`, as the shapes of other locations are generally to broad to perform a point-based matching.

6.1 Parameter Optimization for Heuristic TD

The presented heuristic was developed using the training set of TUD-Loc-2013 (see Section 3). The heuristic is based on seven threshold values which were selected intuitively at first. Consecutively, we will examine the impact on the extraction results when varying each of those threshold values, while keeping the rest of the values to the initial default value as given in Table 3. The analysis is performed on the validation set.

[20] `https://github.com/sanity/quickdt`
[21] Root-mean-square error

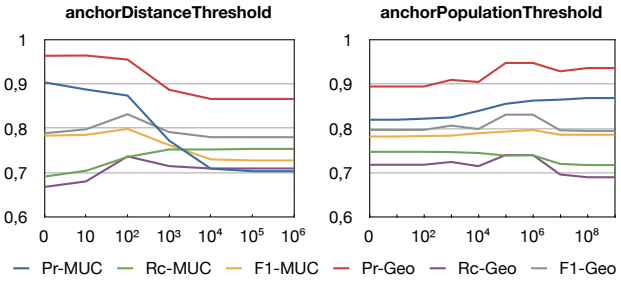

Figure 6: Influence of selected threshold settings on extraction performance of the heuristic approach

The results of our analysis show, that only the variation of *anchorDistanceThreshold* and *anchorPopulationThreshold* have a noticable influence on the extaction results. The value of 2 for the *tokenThresholds* gives best results, values above/below yield a lower F1 measures. A variation of the remaining parameters has no significant influence, which indicates on the other side, that our approach does not overfit to the given data and generalizes well. While we had initially planned to use more sophisticated optimization mechanisms, such as genetic algorithms, for parameter tuning, the results clearly indicate, that this is not necessary.

As Figure 6 shows, the given default values already provide good results in terms of F1 measure, while the *anchorDistanceThreshold* allows for a further adjustment towards more precision or recall. Intuitively, the lower the distance threshold between anchor locations and potential further candidates, the higher the precision. Increasing this threshold allows the recall to rise, however, this results in a comparatively strong decrease of precision.

6.2 Feature Elimination for Machine Learning TD

In Table 2, we describe various features which we use for classification. In total, we extracted 70 features. We used 24 tokens for extracting the binary `containsMarker(M)` feature such as "city", "river", "county", etc. For the features `numLocIn(R)`, `populationIn(R)`, `locSentence(R)`, `uniqueLocIn(R)`, which are to be parametrized with a distance `R`, we used values of 10, 50, 100, and 250 km for each. The feature `distLoc(P)` was extracted for values of 1,000, 10,000, 100,000, and 1,000,000 for `P`. Figure 7 shows the results of the backward feature elimination. The process is as follows: We start with the complete feature set. In each iteration, we remove each of the remaining features once and train the classifier using the training set and test the classifier using the validation set (see Section 3). This means, we ran $n(n+1)/2 = 2485$ train/test cycles. We evaluate the classification results using the F1 measure (note,

Threshold	Value	
unlikelyPopulationThreshold	100,000	
anchorPopulationThreshold	1,000,000	
sameDistanceThreshold	50	km
lowerPopulationThreshold	5,000	
tokenThreshold	2	
lassoDistanceThreshold	100	km
anchorDistanceThreshold	100	km

Table 3: Default threshold values for heuristic TD

that we only evaluated the binary classification performance, and did not employ the MUC or Geo F1 measure). After each iteration, we finally remove the feature, where elimination achieved the best results in F1. This means, that with each step to the right on the x-axis in Figure 7, the mentioned feature was removed in addition to the features on the left.

While the results of the backward feature elimination give no direct evidence of how strong or weak a feature is (for that, chi-squared or information gain tests should be employed), our results show, that we can remove a significant amount of features without harming the classification quality. In contrast—the values indicate, that F1 slightly improves, beginning in the last third of the elimination phase. The oscillation of the results is due to statistical properties and could be eliminated through cross validation, however, we wanted to stick to the predefined training/validation set for better reproducibility.

The results of the feature elimination show, that we can build a robust classifier for TD using a comparatively small feature set. For our following comparison, we rely on a set of the top 15 features, starting on the right of Figure 7. It is interesting to observe, that intuitive indicators, which we already employed in our heuristic, such as `populationNorm` or `nameAmbiguity` are among the leading features. Also, spatial proximity-based features such as `populationIn(R)`, `uniqueLocIn(R)`, `locSentence(R)`, as well as the features `num(c)` and `contains(c)`, based on topologic relations, are among the top 15.

A further question is, how to set the probability threshold, above which candidates are classified as locations. Figure 8 shows the evaluation measures with an increasing probability threshold. Intuitively, a lower threshold classifies more candidates as being locations, resulting in high recall, whereas higher threshold values achieve better precision. The best F1 measure is achieved for a probability threshold of 0.2, whereas higher thresholds lead to decreasing F1 values due to the high loss of recall.

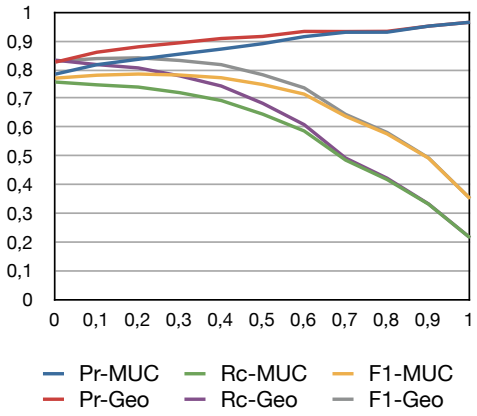

Figure 8: Results of the threshold analysis

7 Comparison

We evaluate our two approaches using TUD-Loc-2013 and the methods already described in Section 6 and compare them to publicly available state of the art approaches for location extraction. In particular, we consider the following Web-based APIs: Yahoo! BOSS GeoServices[22], Unlock Text[23], OpenCalais[24], AlchemyAPI[25], and Extractiv[26]. While all of the mentioned services perform a TR, only Yahoo and Unlock provide a full TD by returning geographical coordinates with each extracted location. OpenCalais, and Extractiv at least retured coordinates for some of the extracted toponyms, while AlchemyAPI does not deliver any coordinates at all.

Naturally, each of the service relies on its own set of location types. We therefore perform a mapping to the location types used in the dataset (see Table 1). The mapping was evaluated and optimized in advance using the training set to ensure a fair comparison.

Unlock is the only service which does not categorize the extracted locations, which is why we exclude it from the TR evaluation.

We compare the results to a baseline TD approach, using a "maximum population" heuristic, which either disambiguates candidates by taking the `CONTINENT` or `COUNTRY` locations, if such exist, or selects the location with the highest population count. The preprocessing and postprocessing phases are identical to the ones described in Section 4.1 and 4.4.

Figure 9 shows the comparison between the baseline, our approaches and state of the art services for the TR task. It is noteworthy, that the baseline already gives comparatively good results and even beats Yahoo in F1, which is due to the strong recall. Alchemy, OpenCalais and Extractiv perform considerably better, but are still outperformed, both by the heuristic and the machine learning TD method. Through machine learning, we can improve F1 to 77,09 % compared to the 76,02 % achieved via the heuristic approach. The runner-up Alchemy achieved 74,12 % F1.

While the previous comparison evaluated the accuracy of the TR, i. e. the correct identification and classification of location entities in text, the TD evaluation investigates, how well the different approaches identify the correct geographical locations (see Figure 10). In contrast to the comparison above, we do not consider Alchemy, OpenCalais and Extractiv here, as they do not provide coordinates for most extractions. On the other hand, we add Unlock to the comparison, which does not categorize extracted locations, but provides coordinates for all of them.

Even more than in the TR comparison, we can see that the baseline already performs considerably well and outperforms Yahoo and Unlock. Yahoo provides a high precision for the TR, but suffers from the comparatively poor results achieved during TD. In contrast

[22] http://developer.yahoo.com/boss/geo/
[23] http://unlock.edina.ac.uk/home/
[24] http://www.opencalais.com/documentation/calais-web-service-api/api-metadata/entity-index-and-definitions
[25] http://www.alchemyapi.com/api/entity/
[26] http://extractiv.com

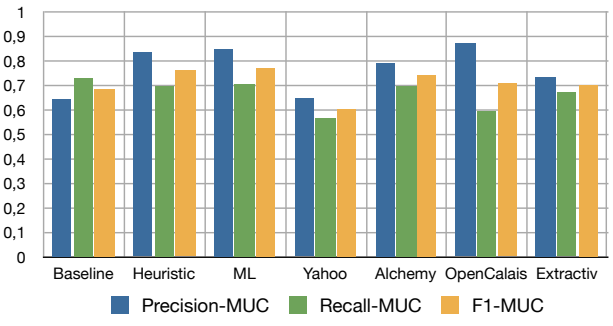

Figure 9: Comparison of TR on TUD-Loc-2013

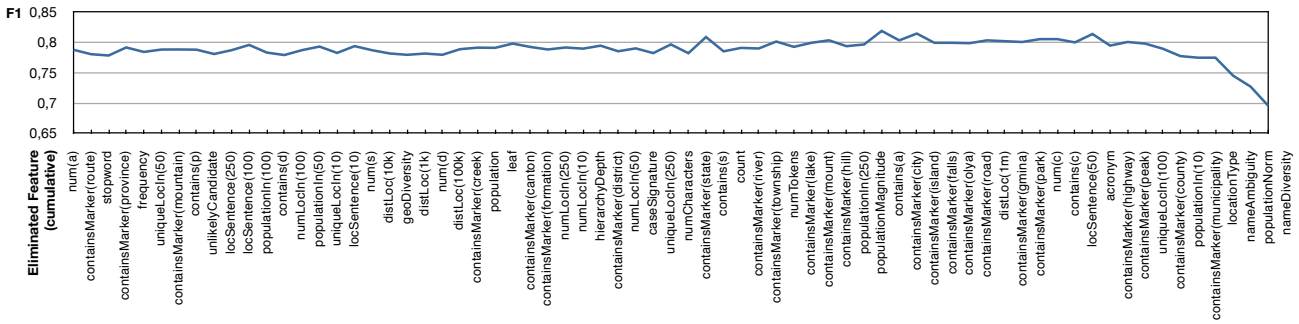

Figure 7: Results of the backward feature elimination

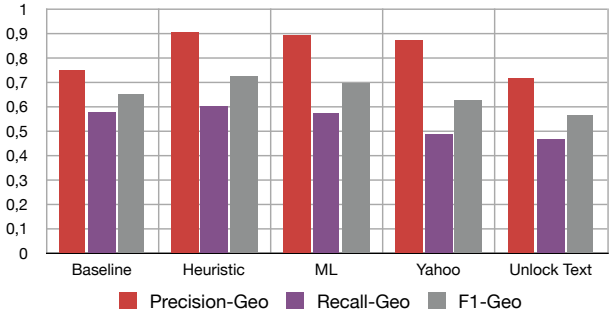

Figure 10: Comparison of TD on TUD-Loc-2013

to the results above, the machine learning-based approach falls short with 69,79 % F1 in comparison to the heuristic, which achieves an F1 value of 72,32 %.

8 Conclusions and Future Work

In this paper we have presented a new dataset for evaluating TR and TD approaches. TUD-Loc-2013 is publicly available for research purposes on the open research platform Areca and therefore facilitates the comparison of future approaches with the results presented here. We have presented two new methods for TR and TR in detail, one relying on a set of heuristics, the other using a classifier trained with machine learning. We have described a comprehensive set of features and seen, that a small subset of those features is sufficient for a well-performing classification-based method.

We have described an aggregated gazetteer database that improves extraction results. Incorporating further sources for obtaining street and ZIP information might further improve those results, but also possibly introduce inaccurate information. An alternative approach might try to exploit map APIs such as Bing or Google when necessary.

On the other hand, the recall achieved during TR can be further improved by extracting more location entities not found in the database. Currently, our approach only allows extraction of address-specific information not in the gazetteer, such as ZIP codes, street names and numbers. We have seen, that currently, the heuristic and machine learning approach deliver a neck-and-neck race, with each one winning either the TR and the TD competition.

In our comparison we have shown, that we can outperform other publicly available Web APIs for extracting location data, using both—heuristic and machine learning—approaches. Thus, we provide a strong foundation for improving current and future applications relying on location-specific data, such as search, geo-targeted advertising, data mining and anlysis, and many more.

Beside the presented dataset TUD-Loc-2013, the methods described within this paper are available as ready-to-use implementations in the Java-based information retrieval toolkit Palladian[27], which is freely available for non-commercial, scientific applications (Urbansky et al. 2012).

References

Amitay, E., Har'El, N., Sivan, R. & Soffer, A. (2004), Web-a-where: Geotagging web content, *in* 'Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 273–280.

Andogah, G. (2010), Geographically constrained information retrieval, Dissertation, Rijksuniversiteit Groningen.

Breiman, L. (1996), 'Bagging predictors', *Machine Learning* **24**(2), 123–140.

Buscaldi, D. & Rosso, P. (2008), 'A conceptual density-based approach for the disambiguation of toponyms', *International Journal of Geographical Information Science* **22**(3).

da Graça Martins, B. E. (2008), Geographically aware web text mining, Dissertation, Universidade de Lisboa, Faculdade de Ciências, Departamento de Informática.

Gale, W. A., Church, K. W. & Yarowsky, D. (1992), One sense per discourse, *in* 'Proceedings of the workshop on Speech and Natural Language', HLT '91, pp. 233–237.

Grishman, R. & Sundheim, B. (1996), Message understanding conference – 6: A brief history, *in* 'Proceedings of the 16th International Conference on Computational Linguistics', Vol. 1 of *COLING '96*, pp. 466–471.

Leidner, J. L. (2006), 'An evaluation dataset for the toponym resolution task', *Computers, Environment and Urban Systems* **30**(4), 400–417.

Leidner, J. L. (2007), Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names, Dissertation, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh.

Li, H., Srihari, R. K., Niu, C. & Li, W. (2002), Location normalization for information extraction, *in* 'Proceedings of the 19th international conference on Computational linguistics', Vol. 1 of *COLING '02*, pp. 1–7.

[27] `http://palladian.ws`

Li, H., Srihari, R. K., Niu, C. & Li, W. (2003), Infoxtract location normalization: a hybrid approach to geographic references in information extraction, *in* 'Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References', pp. 39–44.

Lieberman, M. D. & Samet, H. (2012), Adaptive context features for toponym resolution in streaming news, *in* 'Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR'12, pp. 731–740.

Lieberman, M. D., Samet, H. & Sankaranarayanan, J. (2010), Geotagging with local lexicons to build indexes for textually-specified spatial data, *in* 'Proceedings of the 26th International Conference on Data Engineering', ICDE 2010, pp. 201–212.

Millan, M., Sánchez, D. & Moreno, A. (2008), 'Unsupervised web-based automatic annotation', *Proceedings of the 4th STAIRS Conference: Starting AI Researchers' Symposium* .

Parsons, E. (2012), 'Ed Parsons at Google PinPoint London 2012', `https://plus.google.com/110553637244873297610/posts/8SYwR5Ze6nB`.

Rauch, E., Bukatin, M. & Baker, K. (2003), A confidence-based framework for disambiguating geographic terms, *in* 'Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references', Vol. 1, pp. 50–54.

Sinnott, R. W. (1984), 'Virtues of the haversine', *Sky and Telescope* **68**(2), 159.

Smith, D. A. & Crane, G. (2001), Disambiguating geographic names in a historical digital library, *in* 'Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries', ECDL '01, pp. 127–136.

Smith, D. A. & Mann, G. S. (2003), Bootstrapping toponym classifiers, *in* 'Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references', Vol. 1 of *HLT-NAACL-GEOREF '03*, pp. 45–49.

Urbansky, D. (2012), Automatic extraction and assessment of entities from the web, Dissertation, Technische Universität Dresden, Faculty of Computer Science.

Urbansky, D., Muthmann, K., Katz, P. & Reichert, S. (2012), 'TUD Palladian Overview'.

Wing, B. P. & Baldridge, J. (2011), Simple supervised document geolocation with geodesic grids, *in* 'Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies', Vol. 1 of *HLT '11*, pp. 955–964.