Ad Hoc Determination of Geographic Regions for Concept@Location Queries

Andreas Henrich and Volker Lüdecke

Abstract Textual geographic queries to search engines usually consist of the desired concept and also of one or more terms describing a location, which is often the name of a city, which in turn can usually be grounded with the help of a gazetteer. On other occasions, though, the location refers to a (vague) geographic region and may also be a vernacular expression for that region, so that this location specification cannot be found in a gazetteer.

In this chapter we describe an approach to determine the boundaries for such locations and how to integrate this approach into the query process. The key features of our approach are that a geographic search engine is able to handle any textual description of a geographic region at query time and that this computation can be done completely automatically. In our approach we derive a representation for a region from the toponyms found in the top web documents resulting from a query using the terms describing the location.

In addition to that, we introduce two other uses of this approach: first, this method can be used for answering where-is queries (where only a query location, but no query concept is given), and second, we can determine geographic representations for arbitrary terms that are not genuine geographic regions. In that case, the geographic representation provides a visual impression of the geographic correlation of those terms.

1 Introduction

In Geographic Information Retrieval, textual user queries in the format concept@location are quite common. While the concept-part usually refers to a real-world object, the location-part mostly consists of a certain place or a region, which defines a geographic constraint of the query. Examples are *hotels in Bamberg* or *jobs in*

A. Henrich (✉)
University of Bamberg, Germany
e-mail: andreas.henrich@uni-bamberg.de

southern Germany. A geographic search engine processing such a query therefore has to be able to know the position or the boundary of the given location, which is usually done by looking it up in a gazetteer-like database [12]. Places not contained in there thus cannot be found that way.

Gazetteers usually contain information about cities, locations or regions with distinct boundaries, like administrative regions. In everyday language you can find many more region names, many of which are only vaguely or ambiguously defined and do not have strict borders.

In this chapter we elaborate an approach, which was first published by us in a preliminary version [10][1] that uses knowledge from the World Wide Web to automatically determine any location that is not yet in the database of a search engine at query time, so that any user query in the above format could be answered. While the location-part of a query is usually considered to be geographic in nature, it does not necessarily have to be, which is the second proposition we make in this chapter: a geographic reference in a query must not be restricted to a place or a (vernacular) region, but may be any concept, for which the search engine then has to find out an appropriate corresponding geographic extension. Examples of such queries are *camping ground near theme park* or maybe *cycle path near brewery*.

The chapter is structured as follows: Our approach is explained in Sect. 2, while the usage of arbitrary terms as location-identifier is covered separately in Sect. 3.1. Section 3.2 deals with the application of this approach for answering where-is queries. In Sect. 4 we give an evaluation of the quality and performance of the system, which also shows the influences of some of the parameters involved in the process. Section 5 covers related work. Finally, Sect. 6 concludes the chapter and discusses future work.

2 Our Approach

The situation we are concerned with can be described as follows: We are given a set of terms T_{loc} describing the location part of a concept@location-query.[2] We want to derive a geographical footprint of query region R_{query} (or more precisely $R_{query}(T_{loc})$) representing that location.

In this section we outline a system architecture and its components for a search engine that is able to automatically compute geographic representations for arbitrary terms at query time. Since we want to focus on delimiting geographic regions while processing a geographic query and not on standard text search engines, we assume the following whenever we refer to a geographic search engine:

[1] This chapter extends the preliminary version considering additional potential application scenarios (where-is queries) and extending the experimental results (e. g. considering additional parameter settings and quality measures).

[2] With the query *camping ground near golf course* we get T_{loc} = *golf course* and with the query *hiking in Franconia* we get T_{loc} = *Franconia*.

- This search engine has indexed a good-sized part of the world wide web, as every major search engine has, since we need the WWW-knowledge for determining geographic regions.
- This search engine is able to handle standard textual queries and to provide relevance rankings.
- Users can enter textual queries in the format concept@location. Whether this is done by two separate textfields or by a more sophisticated analysis of the queries does not matter for our purposes.

We will provide the necessary details on how we actually handle this for our experiments in the corresponding passages.

2.1 Aims

Since we want to delimit the query-region R_{query} at query time, the system must be able to do that very fast. So we aimed at a response time of less than one second (using a standard desktop PC). Secondly, we want to provide a geographic representation for any set of terms entered by the user and not just for terms describing a geographic region. Finally, the whole process must work completely automatically without user interaction.

Our prototype system is restricted to text in German language and to locations in Germany. While there are certain differences concerning phrases and patterns for example, the methods are applicable to any language and location data.

2.2 System Architecture

Figure 1 shows the relevant system components of our proposed geographic search engine and how they are related to each other.

Gazetteer A gazetteer provides the location data of known places (and regions) to the search engine.

Region-Engine The Region-Engine comprises all specific methods for delimiting vague or unknown regions. If a location from a concept@location-query is not contained in the gazetteer, the Region-Engine uses location information from the gazetteer for determining the boundaries of the unknown location. We will cover it later in this section in detail.

Geographic Search Engine The ranking process of the geographic search engine has to use distances of geographic coordinates and regions rather than ontological connections between places for the ranking process. The geographical similarity is thus computed by degree of overlap or a distance measure between the query region and the geographic footprint of the document.

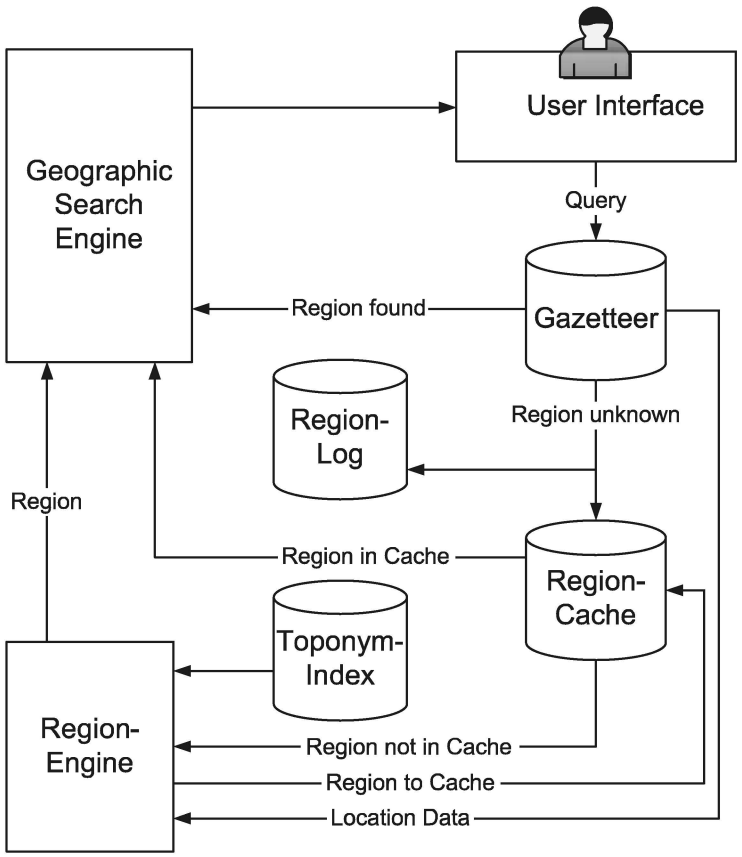

Fig. 1 Proposed system architecture

Region-Cache Caching will improve the performance of the system, but has no other specific function.

Toponym-Index This contains information about the toponyms contained in a document. It is used for improving the performance of our approach.

Region-Log It certainly makes sense to log the location terms entered by users of a live system. Frequently used regions should probably be manually revised and added to the database / gazetteer of the system.

User-Interface In addition to any existing user interface, we recommend giving the user feedback as to what was actually considered his query-location. Ways of providing some kind of relevance feedback are not discussed here, though.

2.3 Region-Engine

The Region-Engine is the main focus of our work. It will take one or more terms T_{loc} and try to determine a corresponding geographic region, whenever that region is used in a query and cannot be found in the gazetteer used by the system or in the cache of previously delimited regions.

The first step is to retrieve documents $D_{retrieved}$ relevant to T_{loc}, describing the query-region R_{Query}. Since our prototype system does not have indexed enough web documents yet, we used the Google-API to get the first 500 results (html-pages

only) for each region description T_{loc} and archived them locally (T_{loc} was used as Google-query without further modifications). We used the first k of these documents as $D_{retrieved}$, while k is a design parameter. That way we can better evaluate the total performance of our approach, since a working geographic search engine would simply use its own (local) data.

The subsequent steps are to extract toponyms contained in $D_{retrieved}$ and to determine their geographic coordinates (which can be done at index time). Finally, we compute a two-dimensional (region) representation based on all coordinates found.

2.3.1 Toponyms

A time consuming process step is to detect and disambiguate toponyms in the documents $D_{retrieved}$. We used the GeoNames gazetteer as well as the OpenGeoDB, but found the former to lead to slightly better results, since it is more comprehensive. We then parsed a German dictionary to remove all location names that were also common German words, but if a location had a population of more than 30.000 people, we did not remove that location name. These decisions are the result of manual experiments and may not be optimal for all kinds of queries or documents. That way we had a list of about 70.000 toponyms in Germany.

All potential toponyms were extracted from the retrieved documents $D_{retrieved}$. If a toponym had more than one corresponding location, we used a simple disambiguation mechanism that chose those toponyms that resulted in the smallest bounding region over all toponyms in the document.

For a better performance at query time, we built a toponym index that contains a list of toponyms contained in $D_{retrieved}$ with their document-weights for each document. Since the toponym extraction should be made at index time, the time spent on retrieving the toponyms of a document at query time is reduced to an index lookup. For a geographic search engine to work, toponym extraction is usually done anyway for determining the geographic footprint of a document. For this step, performance is certainly an issue, but out of the focus of this paper. Obviously, toponym extraction at index time is not possible if $D_{retrieved}$ is derived via the GoogleAPI at query time, but it would be possible and straightforward if $D_{retrieved}$ is taken from the geographic search engine, so for further considerations we assume that our methods are to be used in a geographic search engine that has an index of its own and that does toponym extraction at index time.

2.3.2 Density Surfaces

Having determined all toponyms contained in $D_{retrieved}$, we create a two-dimensional representation based on the aggregation of the geographic coordinates of these toponyms. For this, we used density surfaces, as did Purves et al. [22], using a kernel density estimation with a standard Gaussian function as kernel. That means that we determine the location(s) for each toponym found and weight them by their document and term frequency. As a consequence, each location has an *area of*

influence with a peak at the exact location and a decreasing influence according to the kernel function. Then we sum up all areas of influence. For a faster computation, we discretize this step. That way, the whole target area (which is Germany in our case) is divided into tiles, and we determine the total value V_i for the center of each tile i. We chose a tile size of about one square kilometer per tile, but the ideal tile size depends on the intended application. For a Germany-wide search engine, a maximum resolution of 1 km seems about right, since most regions are larger than a few kilometers and all methods involved are afflicted with a considerable level of uncertainty.

The density surface may be used in two different ways for a relevance ranking by location or distance respectively.

First, all tiles with a value greater than a threshold value T_{min} may be considered to be part of the region, while all others are not. The resulting 2D-area may then be stored in a quadtree or any other representation the search engine uses for storing geographic footprints of documents or query footprints. The geographic similarity between the query footprint and each document footprint can then be computed by region overlap or any other distance measure, which is not the focus of this chapter.

Secondly, depending on the similarity measure used, the density surface representation provides additional information about the area. The higher the values of some tiles, the more likely it is that these tiles lie near the core of the query-region R_{Query}: although in the first instance these high values result from a number of locations that are often mentioned in documents relevant to the query-region and have no immediate causal connection to the core region, experiments show that more often than not the core of the target region indeed lies near the maximum of the density function.

While the kernel density estimation is usually computed on a cell-by-cell basis, this is very time consuming, since there are usually a lot of tiles with values only slightly above zero. We therefore took a different approach and iterated over all location names found in the documents. For each location we computed the corresponding fraction of the total density function for only a number of tiles, as long as the contributing value was above a (low) threshold and added it to the result matrix. If, for example, *Berlin* was the only toponym found in all $D_{retrieved}$, the resulting density surface would have a single peak value in the tile where Berlin is located, falling off to all sides. We would compute the value of this peak as well as the values of a number of neighbouring tiles, leaving out the rest of Germany. When multiple toponyms are found, we compute a fraction for each of them and the resulting sum over all tiles is the result. This is more than 100 times faster than a computation cell-by-cell, as you will see in the evaluation section.

In order to get a satisfying result area A_{result}, it is necessary to determine a threshold value T_{min} for the density surface, with all tiles with a smaller value not being part of A_{result}. Experiments showed that the best threshold-value heavily depends on the query-region R_{query}. In the following we describe a way how to determine a good threshold-value automatically.

2.3.3 Training Data

For evaluating the quality of an automatically computed region representation (A_{result}) we need a correct representation for comparison. For that reason, two persons diligently researched various sources of information to find out the commonly accepted borders of about 120 regions R_{user} of various sizes in Germany. When both persons agreed to a certain border of a region, a polygon shape was drawn in GoogleMaps and later imported by GoogleEarth to get the coordinates of the polygon. Since the borders of most of the regions are vague, the resulting polygons must be seen as an approximation, but are well suited for our purposes, because all regions were created by using the same criteria and judgements. An exemplary region can be seen in Fig. 2, showing the *Havelland* in its geographic representation, while Havelland is also an administrative unit, which has a slightly different border.

Even with a given 'correct' region, it is not easy to define the best threshold-value T_{min}, as there is a trade-off between coverage of the targeted region and a too large,

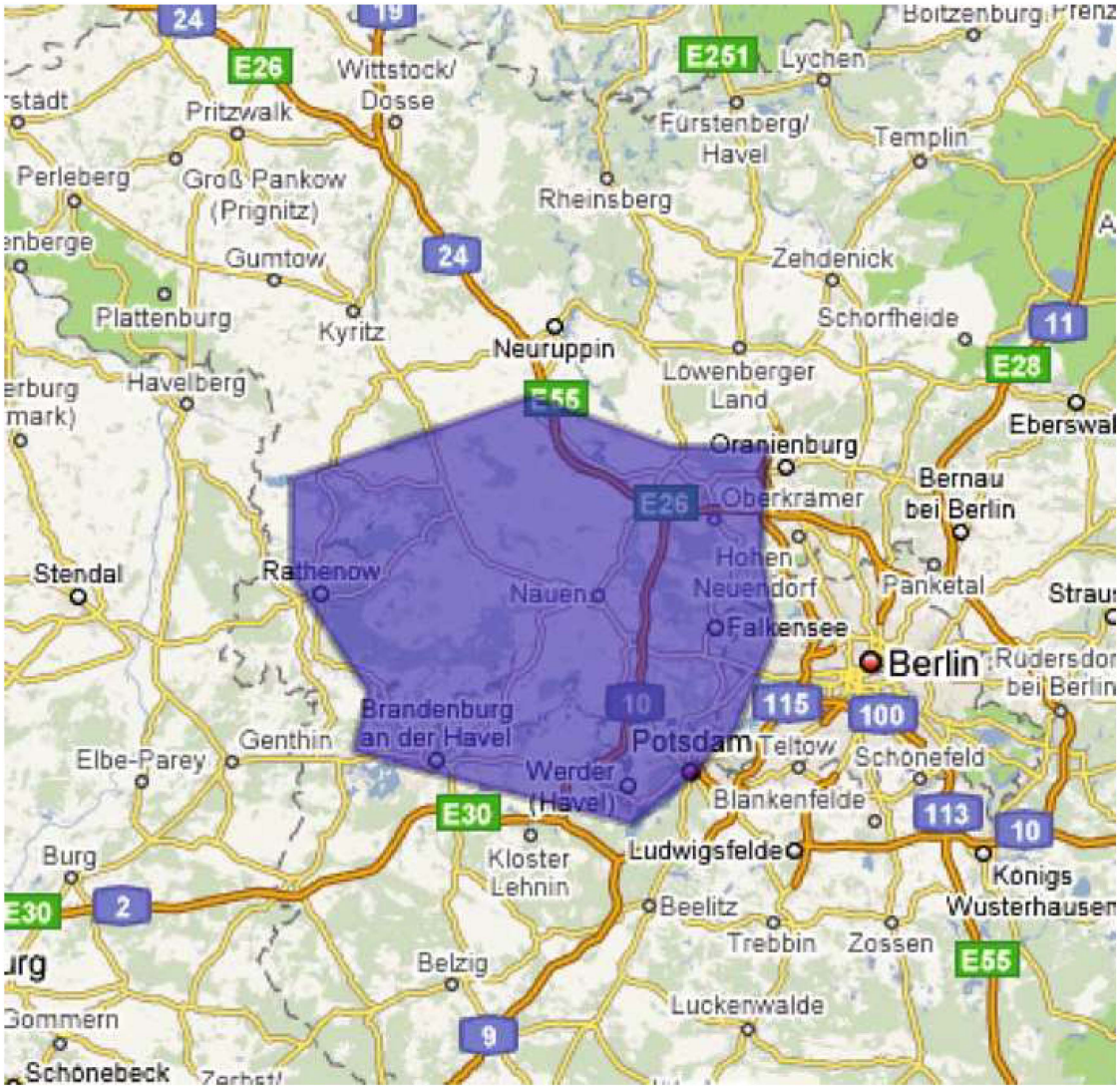

Fig. 2 Polygon representation of *Havelland* (Background ©2008 Google – Map data ©2008 PPWK, Tele Atlas)

faulty shape. Therefore, we implemented two simple measures to automatically find the best threshold-values for 39 regions. One measure takes the absolute values of the density function / the tiles into account, while the other simply counts the number of tiles. Which measure makes more sense thus depends on whether or not the actual values are used in the later ranking process or not. Let *sim* be the similarity between a given user-created region R_{user} and the density surface with a given threshold value T, n the number of tiles being part of R_{user}, m the number of tiles outside the region R_{user} and V_x the value of tile x (n and m depend on the threshold-value, of course.) The measures are:

$$sim_{bin} = 2 * n - m \tag{1}$$

$$sim_{val} = 2 * \sum_{i=1}^{n} V_i - \sum_{j=1}^{m} V_j \tag{2}$$

For determining the optimal threshold-value T_{min} for each region, we simply maximized *sim* by iterating over all (sensible) threshold-values. More sophisticated measures taking distance into account are possible, of course, but probably would not result in very different values. Further experiments in this chapter refer to the usage of sim_{val}.

To derive a function to predict the optimal threshold-value T_{min} for a query-region R_{query}, we computed the correlation between several indicators and the threshold-value on basis of learning data. Indicators we used were: (1) the number of toponyms per document from $D_{retrieved}$, (2) the size of the area, represented by the number of tiles with a value greater than the threshold-value T_{min}, (3) the maximum value V_{max} of all tile values V_i in the density surface, (4) the total sum of all tile values with a value greater than the threshold-value T_{min}, and (5) the total sum of values of all tiles.

Table 1 shows the resulting pairwise correlation values. Obviously, the threshold-value T_{min} correlates quite strongly with the maximum value V_{max} of the density function, which is often greater than one, because we did not normalize the function to the number of toponyms to get a better distinction. Figure 3 shows the individual optimal threshold-values T_{min} in dependency on the maximum values for each training region R_{user}. A simple linear regression lead to the following function for the threshold value:

$$T_{predicted} = max\,(0.2, 0.91 * V_{max} - 0.79) \tag{3}$$

Table 1 Pairwise correlation to T_{min}

Indicator	Correlation to T_{min}
(1) Toponyms per doc	0.604
(2) Area size	−0.369
(3) Maximum Value	0.923
(4) Sum above threshold	−0.177
(5) Total sum	−0.312

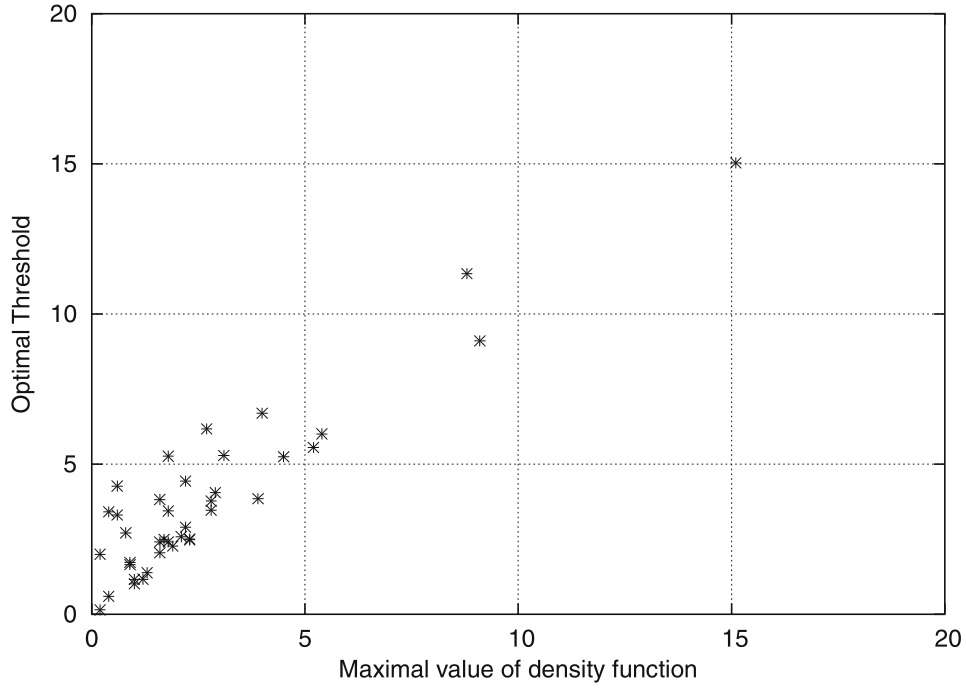

Fig. 3 Correlation between V_{max} and T_{min}

The number of documents k used for delimiting a geographic region influences the performance both in quality and computation speed, of course. A comparison of the influence of k on the results is done in the evaluation section.

3 Application Scenarios

3.1 Arbitrary Terms as Regions

There are certainly geographic information needs which result in queries that do not use toponyms as location reference. For example, a user might want to find a camping ground near a theme park or a cycle path near some breweries. In these cases, *theme park* and *brewery* would be used in much the same way as any toponym as location reference.

If a search engine was able to deal with arbitrary terms as regions using the same automated mechanism to delimit these *region-like references*, it could answer a lot more information needs. The prerequisite for that is that this approach is indeed applicable to any non-toponym terms and that the quality of the results is sufficiently high. With that, *where-is*-like queries can also be answered.

Another application of this approach is to find out whether certain terms have a significant geographic correlation. That may be region specific expressions for things (e.g. bread rolls have completely different names in several parts of Germany) or other things that are typical only for a certain region.

We applied all the mechanisms described in the preceding section to arbitrary terms.

3.2 Where-is-like Queries

Our approach can also be used to answer where-is queries, where no concept-part is given and where the answer to a query is either a geographic region or a location. The region engine as described above provides answers to the former by default, only that in this case the region representation does not have to be integrated further in the ranking process. An evaluation for that is given in the following section. For the latter, where basically a single point is requested, the approach has to be adapted slightly.

For a simple evaluation, we considered the (first) tile with the highest value of the density estimation V_{max} to be the answer. The resolution of that is therefore about 1 km^2. For the scope of a germany-wide tool, this seems to be reasonable enough, but of course this approach as is has an important drawback for this application: There is only one coordinate per city, even for cities which are larger than 1 km^2.

Fig. 4 Result of *where-is Lange Anna* (marked with *A*) and the actual position of it (wikipedia-link, marked by the *arrow*) (Background ©2008 Google – Map data ©2008 PPWK, Tele Atlas)

In [27] the authors show that the general method of this approach can also be used with other data sources, which can provide a finer resolution.

For this evaluation we used our system to determine the location of 27 randomly chosen showplaces in Germany and reviewed the results manually. We measured the distance from the center of the resulting tile to the correct position of the showplaces. Figure 4 shows the answer to *where-is Lange Anna* (which is a famous rock on the island of Helgoland, marked with the arrow). This also shows that other data sources can provide a more fine grained result, as indicates the geocoded wikipedia-article that is linked into the map by Google. Table 2 summarizes the quality of the where-is results. While the results are not bad, a distance of up to 5 km might not be good enough for some applications.

Table 2 Quality of results for where-is queries for single locations

	Up to 5 km	Up to 10 km	Up to 20 km	Completely wrong	Total
Number of results	18	7	1	1	27

4 Evaluation

In this section we want to evaluate three aspects of our approach: (1) Performance and applicability of our system for delimiting regions at query time, (2) Quality of representations for geographic regions and (3) Use of *geographizing* arbitrary terms.

4.1 Performance

As we already mentioned, the performance of the system is very important and mainly depends on the number of documents used for delimiting each region as well as the resolution of the density surface (the number of tiles used). The costs of looking up a region in a gazetteer or in the region cache thereafter are negligible, as is the cost of looking the toponyms up in the toponym index.

For performance evaluation, we computed 20 region representations and measured the time effort for each region. In Fig. 5 you can see the performance of building the density surface incrementally in comparison to computing it on a cell-by-cell basis, each once for $k = 20$ and once for $k = 100$ web documents $D_{retrieved}$ per query-region R_{query}. We used a tile resolution of about one square kilometer and a resolution of about four square kilometers per tile, both of which should be sufficient for the context of (not too small) regions within a Germany-wide search engine. It is clearly visible that the modification of computing the density surface improves the performance by a factor of more than 100. Both versions of the kernel density estimation show the same trend. The reason for that is that both computations depend on the absolute number of locations found, which is identical for both approaches.

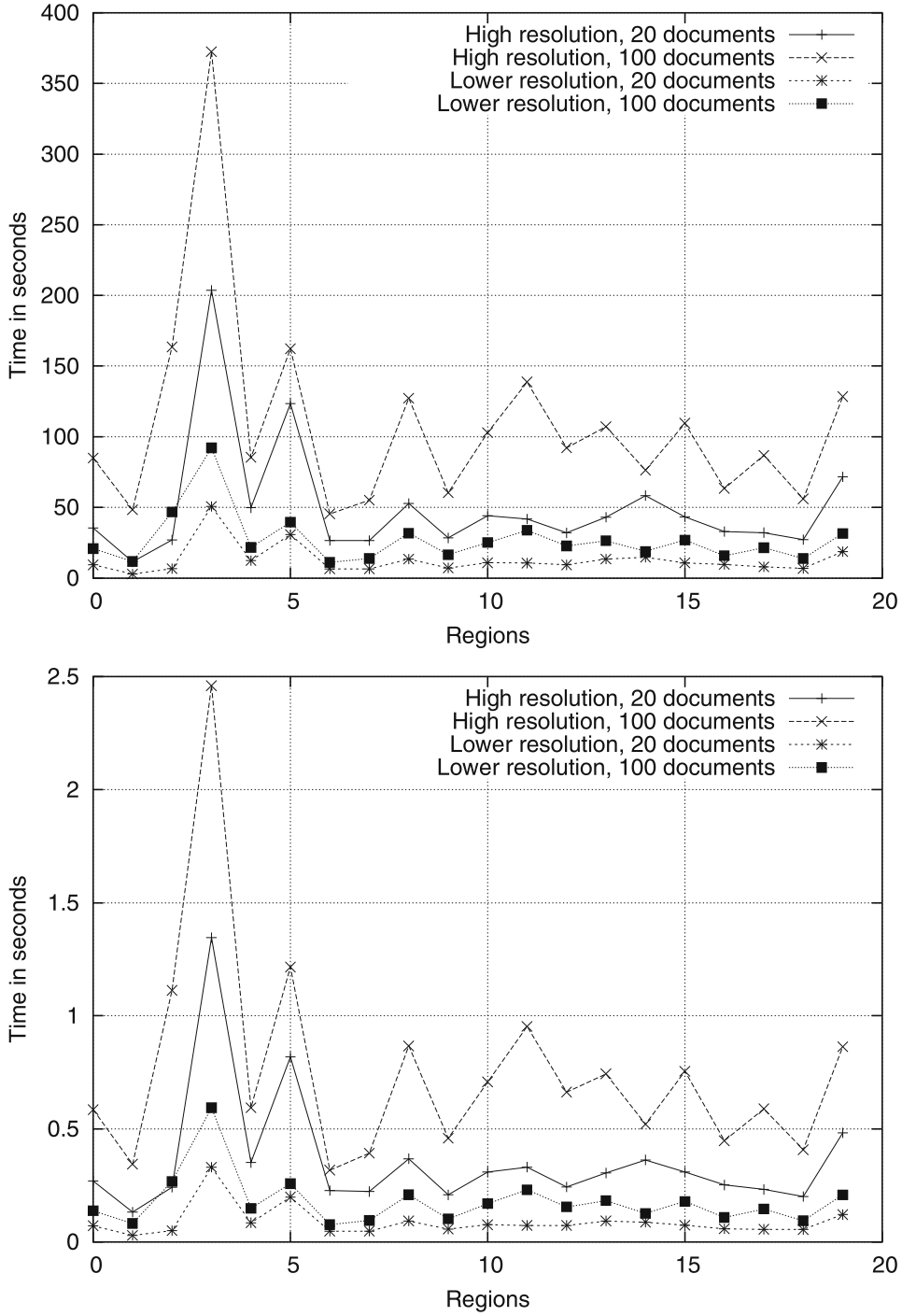

Fig. 5 Performance of standard kernel density estimation (*top*) vs. tuned kernel density estimation (*bottom*) with varying resolution and number of documents k

Overall, the performance of our system seems to make it possible to use our approach at query time, since it usually needs considerably less than one second per query representation (standard desktop PC).

4.2 Quality of Region Representations

There are two aspects to consider for evaluating the quality of the region representations: First, the computation of the predicted optimal threshold value $T_{predicted}$ for the

density surface in comparison to the optimal threshold value for each region T_{min}, which leads to a relative evaluation. Second, an absolute evaluation of the region representation by comparing it to the target region. We will evaluate both aspects in the following.

We tested formula (3) by comparing the predicted threshold values for 39 regions R_{user} not contained in the training set to their corresponding computed T_{min}, so we could see how good the regression function works for further regions. Figure 6 shows the results of this comparison. Obviously, the predicted values come close to the optimum.

Measuring the absolute quality of the representations is difficult for several reasons. Since most of the regions are no crisp regions, but have a vague border, which might also be seen differently by different persons, there is no exact ground truth. As we pointed out earlier, the polygons R_{user} seem to be a good approximation of a ground truth, though. Second, there are several possible points of view regarding similarity between regions. A common baseline approach measures regions of overlap between approximated and correct areas and regions of non-overlap between them. This is a simple point of view that assumes that there are only two states: either a point is within the correct region or not. Distances do not matter therefore, because each point outside the correct region is equally worthless, and each one inside is equally good. This is a very strict semantic, but in some scenarios it is important that an approximation of a geographic region is strictly within the borders of the original region. For example, a query might refer to a law that is only valid inside a region delimited by a border (like a US state). Even a small crossing of that border makes the geographic position worthless.

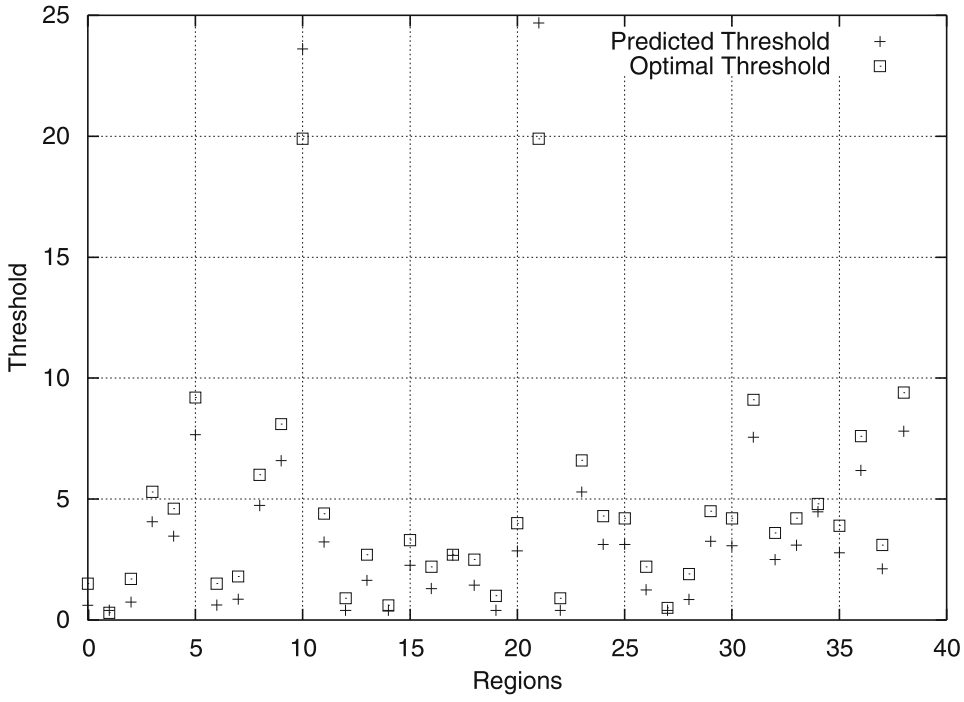

Fig. 6 Predicted threshold value in comparison to optimal threshold value T_{min}

The simplicity of these measures makes them suitable for a rough comparison of absolute result quality, though. One example of those measures can be found in [11, 19]. $A(R)$ is the surface area of a region R, R_{query} is the approximation of the query region, and R_{user} is the correct region:

$$sim_{prec} = \frac{2 * A\left(R_{query} \cap R_{user}\right)}{A\left(R_{query}\right) + A\left(R_{user}\right)} \tag{4}$$

We used this measure to calculate values for the absolute result quality for 75 geographic regions, none of them being part of the training data. Figure 7 shows the corresponding values for $k = 20$.

To get a better understanding of what the similarity numbers actually mean, we will give three examples for them. Figure 8 shows the best result, *Niederbayern*, with a similarity value of 0.84.

The worst result, *Spessart*, has a similarity value of 0, which is because *Spessart* refers to a geographic region as well as an administrative district, which seems to be slightly more popular in the first 20 results, so that the threshold value $T_{predicted}$ for that regions causes the geographic region *Spessart* to vanish. Although the administrative district can be seen as a geographic region, the corresponding R_{user} only refers to the genuine geographic region. The representation for *Ammerland* has a similarity score of 0.06. You can see in Fig. 9 that this region is very small and the formula for $T_{predicted}$ did not work very well here. Increasing the threshold value slightly improves the result a lot. It is probably possible to use heuristics and other

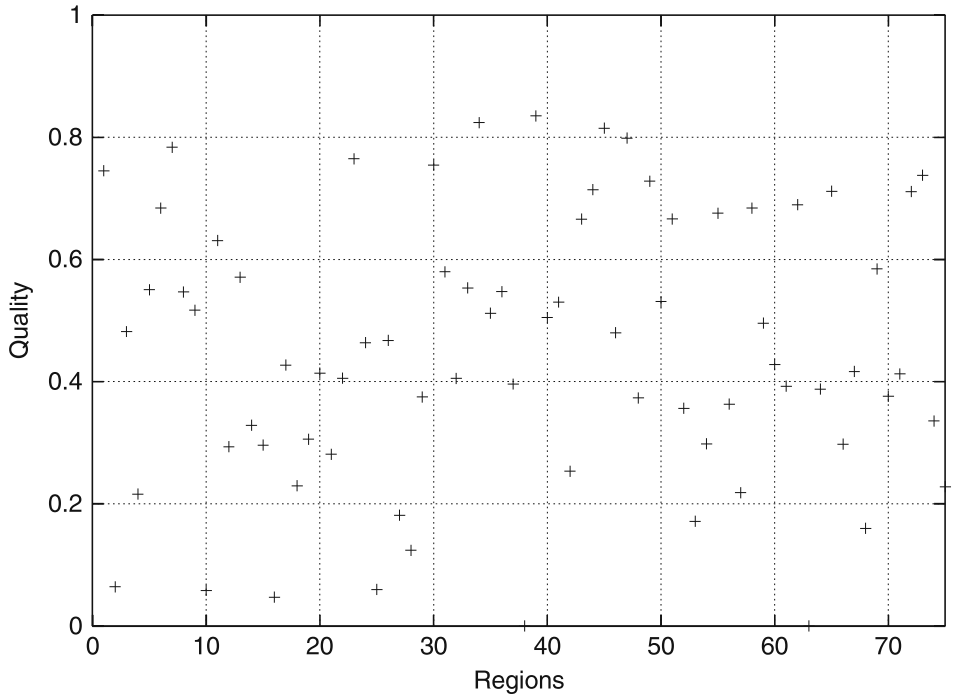

Fig. 7 Absolute quality of automatically determined region representations, $k = 20$

Fig. 8 Comparison of R_{user} and computed region representation for region *Niederbayern* with $sim_{prec} = 0{,}84$. The R_{user} is the polygonal area

more sophisticated methods than the pure linear function for $T_{predicted}$. The same holds true for the region *Sauerland*, which can be seen in Fig. 10.

We found that the shape of a region did not change much between $k = 20$, 100 and 500 documents. Figure 11 shows these effects by the example of the region *Harz*. The main difference between the resulting density surfaces is that the peaks get softer. For arbitrary concepts, such as *mining*, instead of real geographic regions, the number of documents seemed a lot more important, since they are not well defined and as such, a more comprehensive coverage of documents makes more sense.

We used the measure sim_{prec} given in formula (4) to analyse the effect of k on the result quality. Therefore we calculated the quality for the same 75 geographic regions we used above and set k to 20, 50, 100 and 200. Figure 12 shows the resulting graphs. For this figure we sorted the data for each graph separately by

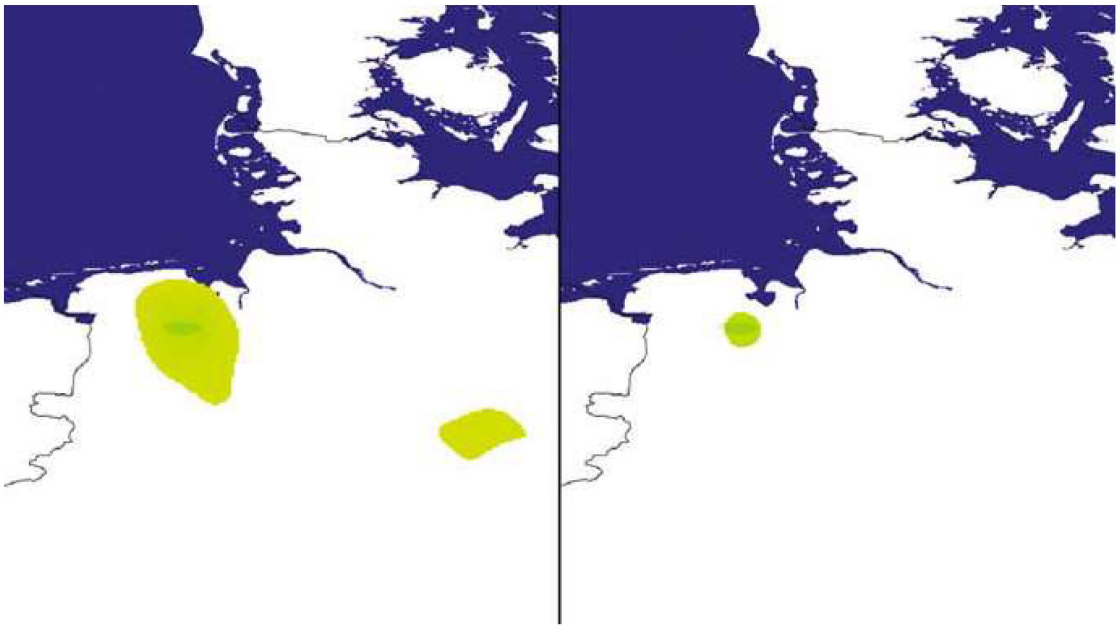

Fig. 9 Comparison of R_{user} and computed region representation for region *Ammerland* with $sim_{prec} = 0.06$ (**left**) and resulting region with a higher threshold T_{min} (**right**). The R_{user} is the polygonal area

result quality (descending) to better visualize the effects. Therefore, one point on the x-axis (e.g. 10) does not stand for a certain region, but for the 10th highest quality. In addition, we calculated the average quality as well as the number of regions where a certain k lead to the best result per region (e.g. for 17 of 75 regions, $k = 20$ lead to

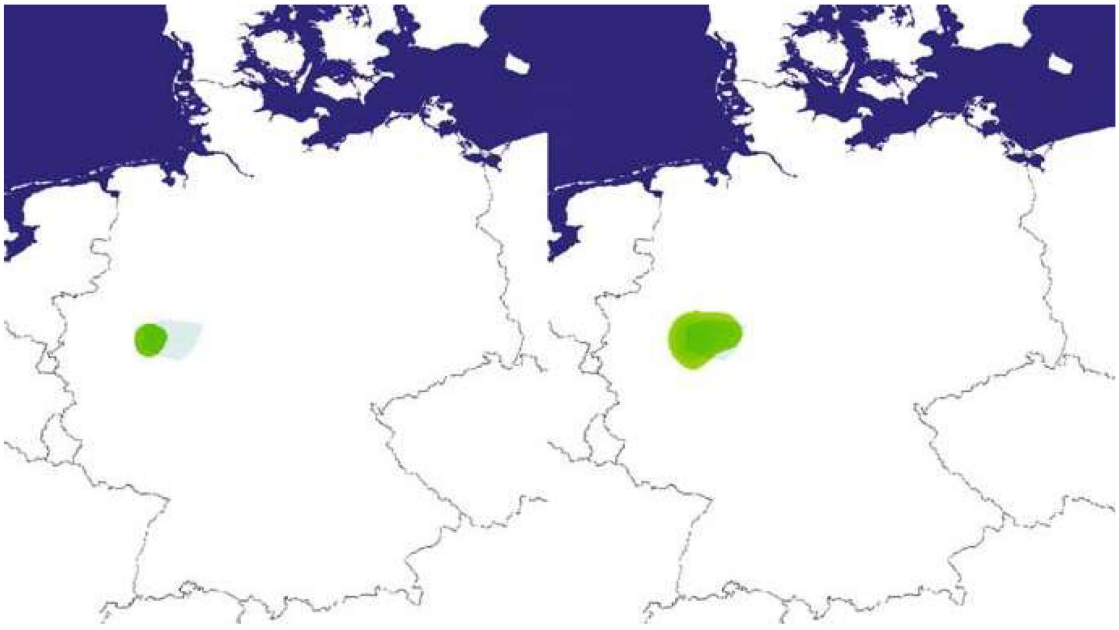

Fig. 10 Comparison of R_{user} and computed region representation for region *Sauerland* with $sim_{prec} = 0.43$ (*left*) and resulting region with a higher threshold T_{min} (*right*). The R_{user} is the solid area

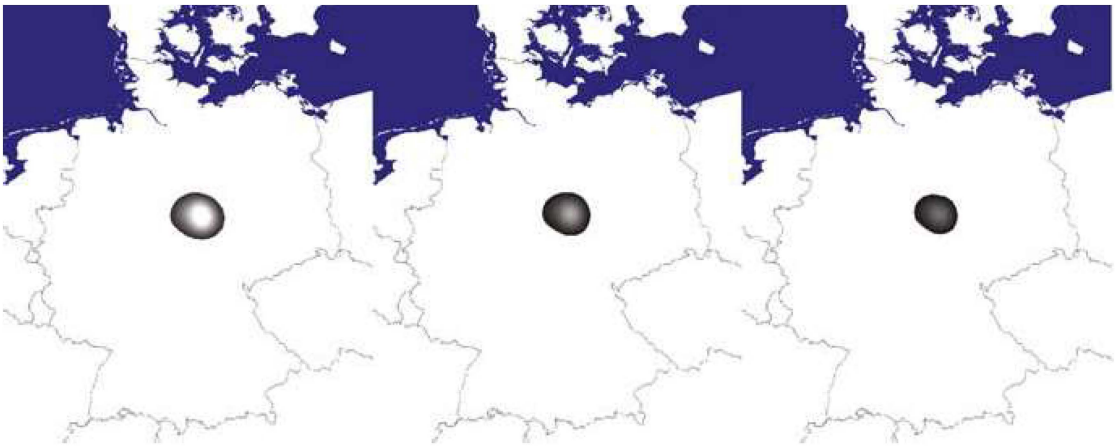

Fig. 11 Region *Harz* with 20, 100 and 500 documents used

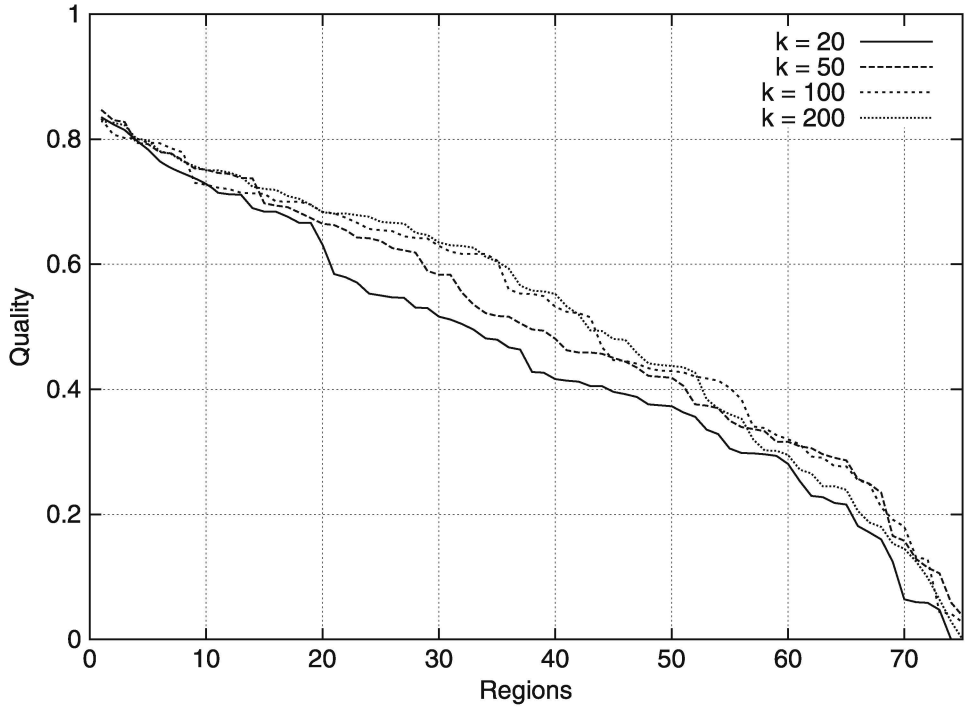

Fig. 12 Comparison of the effects of k, the number of documents $D_{retrieved}$ considered, on the result quality

Table 3 Effects of k on result quality

k	No. of regions with best qual.	Ø quality
20	17	0.45221
50	19	0.49902
100	11	0.51527
200	28	0.51278

the best quality). This data can be found in Table 3. It seems like the quality of the results improves by raising k to about 100, but this data is only an indication for that, since there are many parameters involved. Future work will be to further analyse the effects of k, to adapt the regression function used for determining $T_{predicted}$, and to use further heuristics to improve result quality.

Another important factor in this respect might be the (Google-)rank of a document. We therefore made a simple analysis of considering the rank of a document as a weight in the kernel density estimation. Obviously, the top ranked documents $D_{retrieved}$ should be more valuable than lower ranked documents. On the other hand, the ranks of documents (should) reflect the similarity between the query terms and the content of the documents. For this application, though, we are interested in the geographic correlation of the document's content to the geographic concept named by the query, which might not correlate to the result rank.

For this experiment, we compared four weighting functions for the toponym locations for the kernel density estimation, one of them being the none-weighted standard case used in the rest of this chapter ($w = 1.0$). Two of them were simple linear functions, one decreasing ($w = 1.0 - rank * 0.01$) and one increasing ($w = 1.0 + rank * 0.01$) (and as such weighting lower ranks higher than top ranks), and the fourth is a logarithmic function ($w = 1/rank$).

Figures 13 and 14 show the results for $k = 20$ and $k = 100$, while Table 4 summarizes these results, which indicate that there is no strong correlation between the rank of a result and its impact on result quality. With $k = 100$ the weights for the low ranked documents become very small for logarithmic and linear (decreasing) weighting, so that the regression function (3) does not work very well here. We therefore normalized the weights:

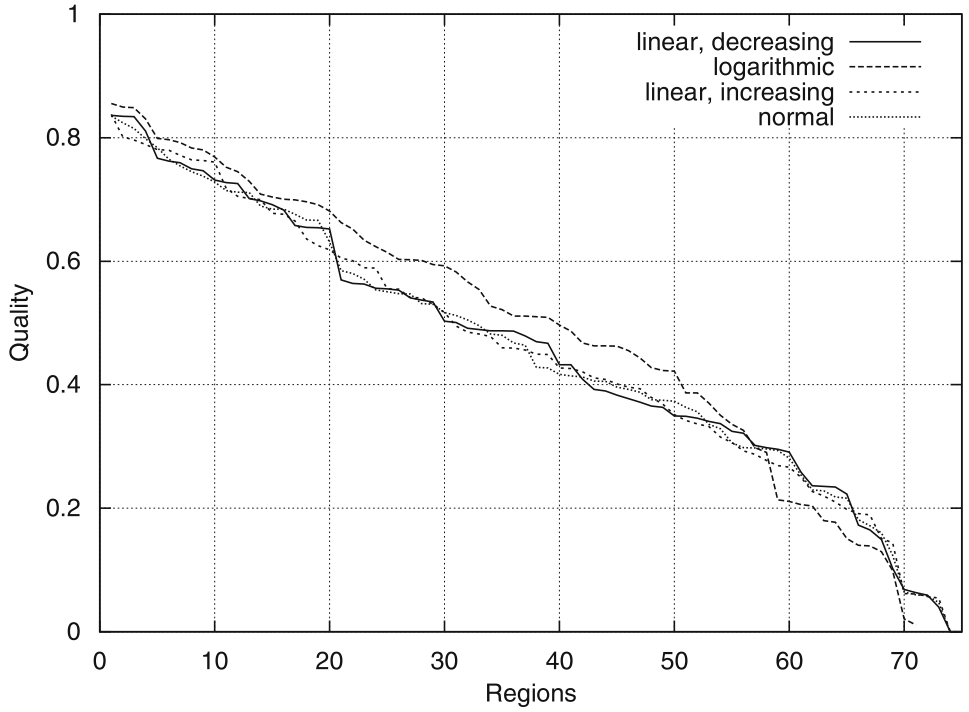

Fig. 13 Effects of weighting the Google result rank with $k = 20$

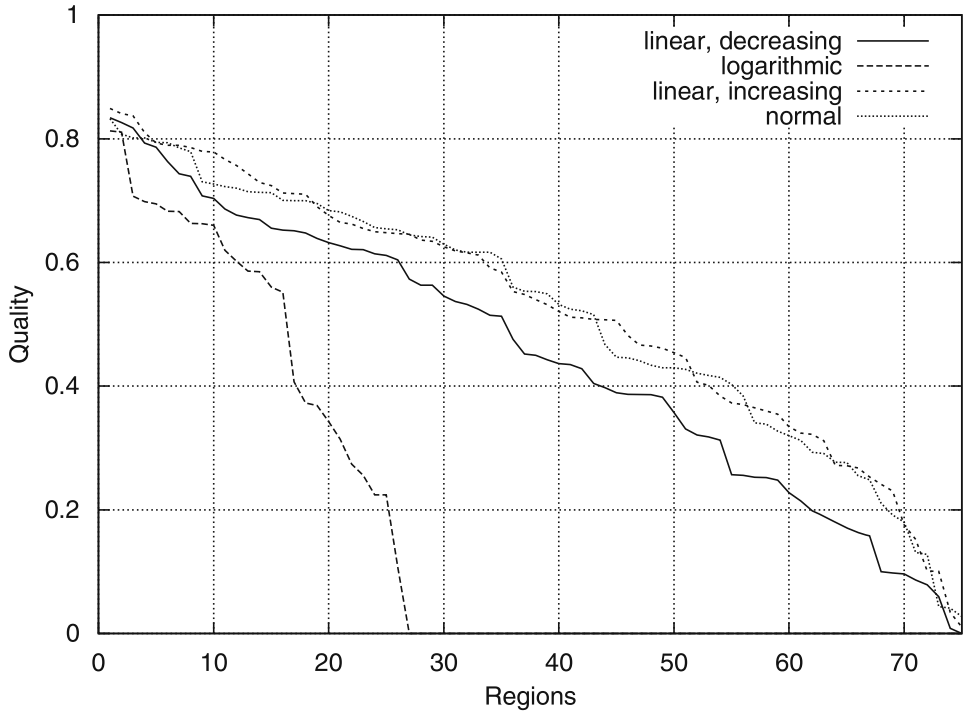

Fig. 14 Effects of weighting in the Google result rank with $k = 100$

$$w_{i,normalized} = w_i * \frac{k}{\sum_{j=1}^{k} w_j} \tag{5}$$

The results achieved with these normalized weights for $k = 100$ are given in the last columns of Table 4.

The presented results suggest that a weighting of the rank of a document does not significantly improve the quality of the geographic representations. The reason seems to be that the first 100 documents $D_{retrieved}$ (with often more than two million result documents overall) are more or less equally good. Future work will be to consider additional parameters, since these results might change slightly with adapted regression functions for T_{min}.

Table 4 Effects of weighting in the Google result rank on result quality

		Without normalization		With normalization	
k	Weighting	No. of regions with best qual.	Ø qual.	No. of regions with best qual.	Ø qual.
20	None	0	0.45221	–	–
20	Linear, decreasing	15	0.45312	–	–
20	Logarithmic	41	0.47757	–	–
20	Linear, increasing	19	0.45091	–	–
100	None	18	0.51527	18	0.51527
100	Linear, decreasing	15	0.44874	17	0.49913
100	Logarithmic	8	0.17963	26	0.43884
100	Linear, increasing	34	0.52264	14	0.49859

4.3 Geographizing Arbitrary Concepts

Evaluating the geographic representations for arbitrary terms is much more difficult, since there is usually no correct result to compare it to. We will discuss the applicability of our approach on the basis of some examples.

Weisswurst (Bavarian veal sausage) is especially popular in the south of Germany. It is often said that the border for this is the Main River, which is therefore often called the *Weisswurstaequator*, the equator of veal sausage. Geographic references like *south of the Weisswurstaequator* are commonly used in German language.

Fig. 15 Visualization of the concept *Weisswurstaequator*, a common word for the Main River (highlighted for comparison)

Figure 15 shows the result of representing it by means of our approach. It is a bit wide, but otherwise represents the term quite correctly. Since this is more or less a geographic region, it is one of the easier examples.

We found a map on the cultivation of sugar beets in Germany (see Fig. 16)[3]. We used the heading of that map as query-region. The resulting area of that query can be seen in Fig. 17, which is by no means exact, but the general idea is correct, anyway.

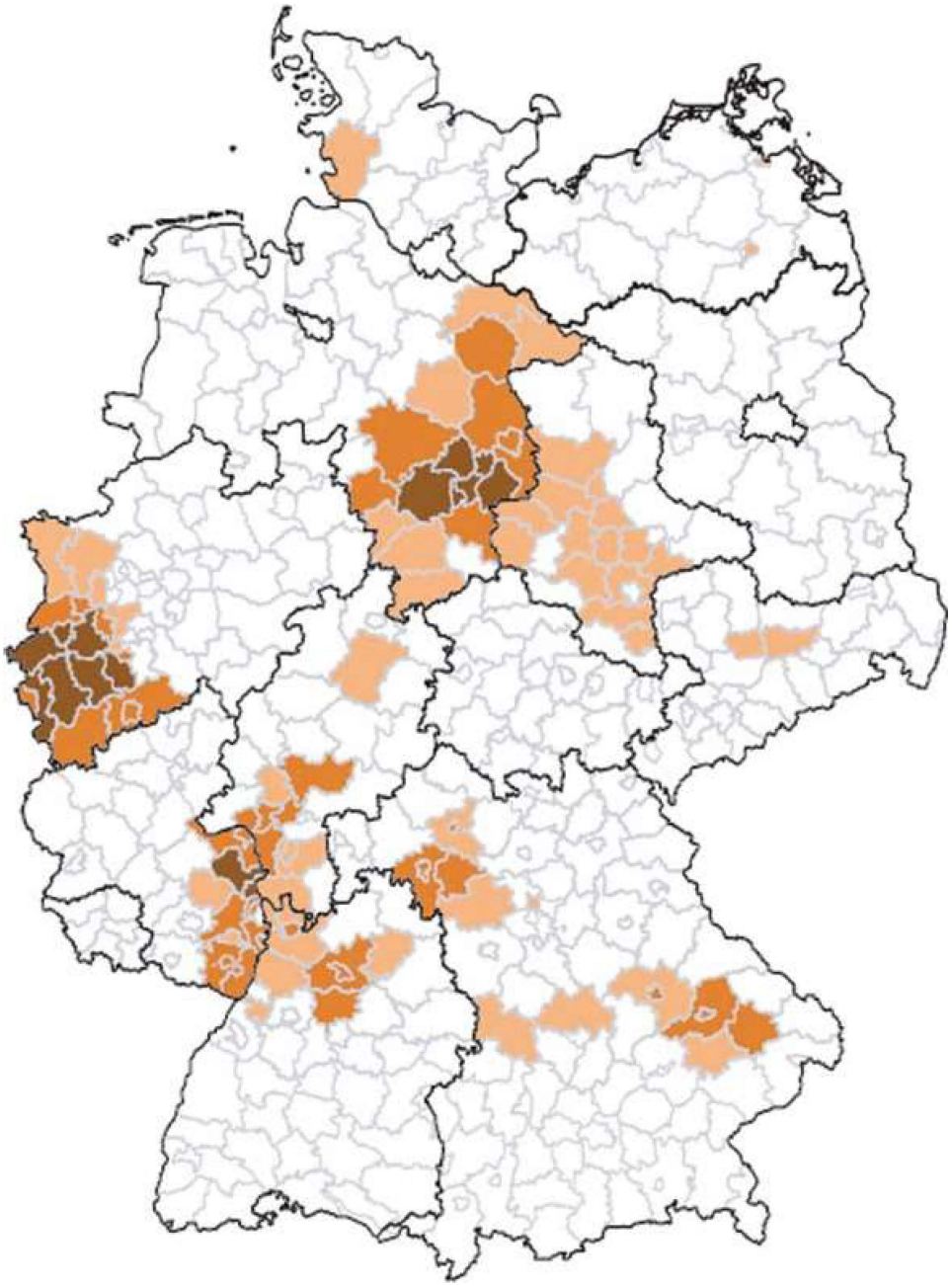

Fig. 16 Cultivation of sugar beets in Germany, source: i.m.a – information.medien.agrar e.V

[3] http://www.ima-agrar.de/_redesign/Dateien/Zuckerrueben anbau_.pdf, February 1, 2008

Fig. 17 Cultivation of sugar beets in Germany as result area A_{result}

We discovered a project run by linguists, who manually created maps about the region-specific use of language [7]. Figure 18 (right) shows the regional distribution of various terms used for *bumper cars*. The circle highlights the region where the very uncommon word *Knuppautos* is used for that. Our automatically created representation for *Knuppautos* can be seen in Fig. 18 (left).

While there were a lot of interesting and promising results, there were of course also many terms, for which we could not find a suitable geographic representation, while suitable refers to what we *expected*, since there does not seem to be a correct result to compare against, so we do not provide a numerical evaluation of the overall quality of this kind of representation. Nevertheless, we think that applying this approach to arbitrary concepts will be of some use for geographic search engines as well as standalone applications.

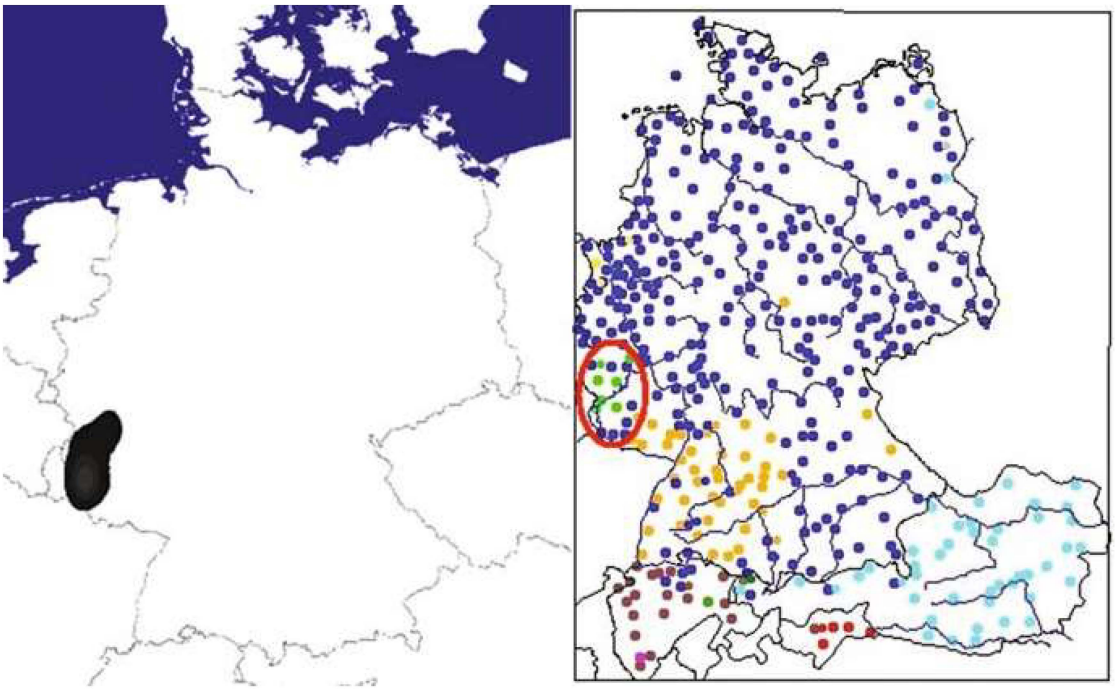

Fig. 18 Geographic representation for *Knuppauto* (*left*) and geographic extension of the language use of *Knuppauto* (*right*)

5 Related Work

Gazetteers or geo-ontologies form the geographical database of most geographic search engines [15]. They usually contain a place-name, the type of place, and a geographic footprint representing its location or its geographical extent [12]. This data is used for query disambiguation, query term expansion, relevance ranking or simply toponym detection in text [14]. A review of these functions is given by Jones et al. [13].

Unfortunately, not all geographic references used in queries can be found in those gazetteers, so that terms used for a geographic constraint cannot be matched to a certain region. The boundaries of these regions are also often only vaguely defined [4, 8]. Apparently, there is a need for identifying the boundaries of vague regions.

Several different formalisms have been proposed as an appropriate representation for vague regions, including supervaluation semantics [3, 4, 17], pairs of non-fuzzy sets [5, 6] and fuzzy footprints [9, 22, 25].

Alani et al. [1] approximate regions based on Voronoi diagrams which are constructed by a set of points known to be inside the target region and another one with points outside the region, but they do not deal with the source of the information about the sets. Vögele et al. [28] present a similar approach, where regions are represented by indeterminate boundaries with an upper and a lower approximation, which is also based on given sets with points and regions inside/outside the target region.

Methods for (semi-)automatically creating representations for vague regions usually mine the knowledge contained in the world wide web. The following approaches use documents retrieved from the www with certain queries or phrases containing the targeted region. Toponyms contained in these documents are extracted and disambiguated [18, 20] and provide the data for the algorithms.

Purves et al. [22] use kernel density surfaces for the representation of the imprecise regions. The authors elaborate their approach in Jones et al. [16], describing alternative web harvesting techniques for mining location information for geographic regions from the web. Reinbacher et al. [23] present two approaches to compute representations of regions, based on evidence of points that are likely to lie inside or outside this region. Arampatzis et al. [2] present methods to obtain locations inside the target region and locations outside from the www and use trigger phrases and patterns for the document retrieval to improve precision. They then approximate regions based on Voronoi diagrams. Schockaert et al. [25] also make use of regular expressions to acquire place names by a web search. They present a method for automatically determining vague footprints, represented as fuzzy sets.

A side aspect of Schöning et al. [26] deals with visualizations of geotagged Wikipedia articles.

Some approaches focus on small regions or domain specific regions. Schockaert et al. [24] present a technique to construct representations of the spatial extent of neighbourhoods. Pasley et al. [21] deal with the usefulness of different information resources depending on the size of the region.

While the basic principles of our approach are quite similar to some of the approaches above, we focus on integrating these mechanisms into the query process. Automation and performance are big issues therefore. While using phrases and additional concepts for retrieving $D_{retrieved}$ seems to lead to promising results for pure geographic regions like in Jones et al. [16] or Schockaert et al. [25], we try to evaluate the application of these methods to any given term. Thus, we want to be able to provide a representation or a map for arbitrary terms, which might be a geographic region or something completely different, like *mining* or *walking*, in which case the result should be a geographic representation of locations associated with that particular term.

6 Conclusion and Future Work

In this chapter we described an approach how to integrate a mechanism for delimiting geographic regions efficiently into a geographic search engine and showed that the performance of our system is good enough to provide such a feature at query time and that the quality of the representations were also appropriate for that application.

In addition to that we applied this approach to arbitrary terms instead of geographic regions only and showed that it produced reasonable results, leading to interesting further applications.

Future work will certainly be to further optimize the results for geographic regions as well as for arbitrary concepts. An interesting piece of work is how to best mine the information provided by the density surface for the ranking process. It may be also important to use other sources of data to get a finer resolution for small areas or single places.

References

1. H. Alani, C. B. Jones, and D. Tudhope. Voronoi-based region approximation for geographical information retrieval with gazetteers. *International Journal of Geographical Information Science*, 15(4):287–306, 2001.
2. A. Arampatzis, M. J. van Kreveld, I. Reinbacher, C. B. Jones, S. Vaid, P. Clough, H. Joho, and M. Sanderson. Web-based delineation of imprecise regions. *Computers, Environment and Urban Systems*, 30(4):436–459, 2006.
3. B. Bennett. Application of supervaluation semantics to vaguely defined spatial concepts. In *COSIT 2001*, pages 108–123, London, UK, 2001. Springer-Verlag.
4. B. Bennett. What is a forest? On the vagueness of certain geographic concepts. *Topoi*, 20(2):189–201, 2001.
5. T. Bittner and J. G. Stell. Vagueness and rough location. *Geoinformatica*, 6(2):99–121, 2002.
6. E. Clementini and P. D. Felice. Approximate topological relations. *International Journal of Approximate Reasoning*, 16(2):173–204, 1997.
7. S. Elspaß. Variation and change in colloquial (standard) german – The Atlas zur deutschen Alltagssprache (AdA) Project. *Standard, Variation und Sprachwandel in germanischen Sprachen/ Standard, Variation and Language Change in the Germanic Languages*, 41:201–216, 2007.
8. M. Erwig and M. Schneider. Vague regions. In *SSD '97: Proc. of the 5th Intl. Symposium on Advances in Spatial Databases*, pages 298–320, London, UK, 1997. Springer-Verlag.
9. M. F. Goodchild, D. R. Montello, P. Fohl, and J. Gottsegen. Fuzzy spatial queries in digital spatial data libraries. In *Fuzzy Systems Proc.*, volume 1, pages 205–210, Anchorage, AK, USA, 1998.
10. A. Henrich and V. Lüdecke. Determining geographic representations for arbitrary concepts at query time. In *LOCWEB '08: Proc. of the First Intl. Workshop on Location and the Web*, pages 17–24, New York, NY, USA, 2008. ACM.
11. L. L. Hill. *Access to Geographic Concepts in Online Bibliographic Files: Effectiveness of Current Practices and the Potential of a Graphic Interface*. PhD thesis, University of Pittsburgh, 1990.
12. L. L. Hill. Core elements of digital gazetteers: Placenames, categories, and footprints. In *ECDL '00*, pages 280–290, London, UK, 2000. Springer-Verlag.
13. C. B. Jones, A. I. Abdelmoty, and G. Fu. Maintaining ontologies for geographical information retrieval on the web. In *CoopIS/DOA/ODBASE*, pages 934–951, 2003.
14. C. B. Jones, H. Alani, and D. Tudhope. Geographical information retrieval with ontologies of place. In *COSIT 2001*, pages 322–335, London, UK, 2001. Springer-Verlag.
15. C. B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies an overview of the spirit project. In *SIGIR '02*, pages 387–388, New York, NY, USA, 2002. ACM.
16. C. B. Jones, R. S. Purves, P. D. Clough, and H. Joho. Modelling vague places with knowledge from the web. *International Journal of Geographical Information Science*, 22(10):1045–1065, 2008.
17. L. Kulik. A geometric theory of vague boundaries based on supervaluation. In *COSIT 2001*, pages 44–59, London, UK, 2001. Springer-Verlag.
18. R. Larson. Geographic information retrieval and spatial browsing. In L. Smith and M. Gluck, editors, *Geographic Information Systems and Libraries: Patronsand Mapsand and Spatial Information*, pages 81–124, 1996.

19. R. R. Larson and P. Frontiera. Spatial ranking methods for geographic information retrieval (gir) in digital libraries. In R. Heery and L. Lyon, editors, *Research and Advanced Technology for Digital Libraries: 8th European Conf., ECDL 2004*, LNCS 3232, pages 45–57. Springer-Verlag, 2004.
20. J. L. Leidner. Toponym resolution in text (abstract only): "which sheffield is it?". In *SIGIR '04*, pages 602–602, New York, NY, USA, 2004. ACM.
21. R. C. Pasley, P. D. Clough, and M. Sanderson. Geo-tagging for imprecise regions of different sizes. In *GIR '07: Proc. of the 4th ACM Workshop on Geographical Information Retrieval*, pages 77–82, New York, NY, USA, 2007. ACM.
22. R. Purves, P. Clough, and H. Joho. Identifying imprecise regions for geographic information retrieval using the web. In *Proc. of GIS RESEARCH UK 13th Annual Conf.*, pages 313–318, Glasgow, UK, 2005.
23. I. Reinbacher, M. Benkert, M. J. van Kreveld, J. S. B. Mitchell, and A. Wolff. Delineating boundaries for imprecise regions. In G. S. Brodal and S. Leonardi, editors, *ESA*, volume 3669 of *LNCS*, pages 143–154. Springer, 2005.
24. S. Schockaert and M. D. Cock. Neighborhood restrictions in geographic ir. In *SIGIR '07*, pages 167–174, New York, NY, USA, 2007. ACM.
25. S. Schockaert, M. D. Cock, and E. E. Kerre. Automatic acquisition of fuzzy footprints. In R. M. et al., editor, *OTM Workshops*, volume 3762 of *LNCS*, pages 1077–1086. Springer, 2005.
26. J. Schöning, B. Hecht, M. Raubal, A. Krüger, M. Marsh, and M. Rohs. Improving interaction with virtual globes through spatial thinking: Helping users ask "why?". In *Proc. of the Intl. Conf. on Intelligent User Interfaces (IUI)*, 2008.
27. F. Twaroch, C. Jones, and A. Abdelmoty. Acquisition of a vernacular gazetteer from web sources. In *LOCWEB '08: Proc. of the First Intl. Workshop on Location and the Web*, pages 61–64, New York, NY, USA, 2008. ACM.
28. T. J. Vögele, C. Schlieder, and U. Visser. Intuitive modelling of place name regions for spatial information retrieval. In *COSIT*, pages 239–252, 2003.