RegionRank: a New Approach for Creating Web Search Services

Máté Láng, Levente Pál, Zoltán Sebesi, Károly Simon, Péter Szilágyi, Tamás Török-Vistai
Codespring LLC, Babes-Bolyai University
lang.mate@codespring.ro
pallevente13@gmail.com
sebesi.zoltan@codespring.ro
simon.karoly@codespring.ro, ksimon@cs.ubbcluj.ro
peterke@gmail.com
torok.tamas@codespring.ro

Abstract— **The aspects and ideas behind the RegionRank software are presented. The project aims to provide a whole new approach to searching geographical and related data on the Internet.**

RegionRank is a map service based web application. It generates results for queries containing multiple criteria and displays these results on a heatmap. In this way, the user can easily select the areas which are the best match for the given query. The user can choose from multiple aspects to build the query, and there is a possibility for setting the importance of these aspects by associating different weight values to each corresponding criterion.

A significant difference between RegionRank and other map-based web applications is that it extends the concept of POI (Point of Interest) and introduces the ROI (Region of Interest) concept. In this way not only points, but also regions can be defined and searched by the application.

Furthermore, in the framework of the application there is a possibility for using the classical searching approach and data representation. The ROIs can be visualized on the map and clustering algorithms will be used for view optimization.

RegionRank provides a unified API for handling data originating from different data providers, supports different formats, being easily extensible to integrate new data sources.

Keywords- Point of Interest, heatmap, clustering, web search

I. INTRODUCTION

The RegionRank project aims to implement a web search service based on a new approach.

For demonstrating the idea, an example can be considered. Foreign students visiting Cluj-Napoca, would like to plan their evening. They want to watch a movie, serve a dinner at a local restaurant, take a walk and go sight-seeing. They also want to minimize the time spent with unnecessary travelling. Using the currently available search services, they have to search separately for each mentioned criterion: first for cinemas, then for restaurants, then later for parks and finally for local attractions. After getting the results, they have to evaluate and make a proper decision. This method is rather unpleasant and time consuming.

The example can be generalized, and applied to other specific search situations. For example, consider a young couple, searching for a rent in a region which has decent public transportation, reasonable prices, low crime rates, good schools and medical centers in the area etc.

Compared to other currently available services, RegionRank provides a faster and more convenient way for searching. Based on the selected criteria, it generates a heatmap, displaying those regions in a city, which are the best-matches, and also those ones, which are the worst, in an easily analyzable way. According to their needs, users can select one or more criterion and they can also set the importance of each criterion separately.

Some examples of criteria which the application can search for are:

- Nearby POIs (schools, hospitals, theaters, museums, cinemas, restaurants, malls, recreational and sport centers, etc.)
- Transportation network (primary roads, public transportation nodes, bicycle roads, etc.)
- Environmental factors (air and noise pollution, green areas, cell phone coverage, public safety statistics, etc.)
- Residents of an area (political views, religious groups, health conditions, etc.)
- Time-dependent criteria (weather, local events, seasonal offers, etc.)

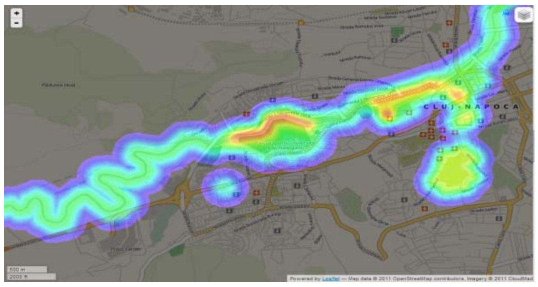


Figure 1. Result of a complex search displayed on a heatmap (based on a sample data-source containing restaurants and bars, green areas and rivers in Cluj-Napoca). The regions marked in red or warmer colors are more appealing to the user`s search criteria.

An important feature of the project is that it extends the concept of POI (Point of Interest) and defines the ROI (Region of Interest) concept. The application can search not only for objects characterized by a geographic coordinate (restaurants, cinemas etc.), but also for objects represented by abstract shapes (parks, roads, areas with different crime rates etc.). The software supports two result presentation strategies: the default heatmap presentation and the “classical” presentation. The classical presentation is optimized, using clustering algorithms.

Currently there are many initiatives that encourage organizations to make their own data transparent, to serve public interest. As examples the “European Public Sector Information Program” and the “American Open Government Initiative” can be mentioned. Countries have a tendency of joining such programs, making their gathered data publicly available.

The complexity of importing and aggregating these data is that they exist in various formats, which are not standard, and the accessing methods can also vary from case to case. RegionRank aims to find a solution to this problem, and to provide a unified API to handle data originating from different data sources, being easily extensible, so the application can adapt to format changes in an agile way.

The RegionRank application is a prototype that meets the previously mentioned requirements. The application itself is composed of three independent components: a server application (including a public API) and two web-based client applications. The server is responsible for storing and handling data, it implements the data access layer, contains the business logic layer, which offers the services used by the client applications (searching, clustering etc.). The RegionRank API is responsible for the communication between the server and the client applications.

The system provides an administration user interface for the maintenance personal, offering possibilities to manage persistent data (management of ROIs and external data sources, statistics, fine tuning the application, etc.).

For the common users (without administration rights), the system provides a map-based user interface. Using this interface they can select the criteria for their search queries, set the importance of these criteria separately and generate a heatmap accordingly. Clicking on a region of the heatmap shows additional information about that region. The client application also offers a way for visualizing data using a “classical” approach. The ROIs are represented on a map, and a clustering algorithm is used for a better presentation.

In the first section of this article the proposed methods and theoretical concepts are described. These are followed by a brief presentation of the used development tools, technologies, design patterns and development methods. The third section is the actual description of the project. Shortly describes the requirements, presents some use cases and the architecture of the application, the major milestones of the implementation, respectively. It also contains a short case study, which serves as a short user documentation. Finally, some conclusions and further development possibilities are presented.

II. METHODS AND THEORETICAL CONCEPTS

The business logic components of the application do a lot of background calculations and operations in order to provide the mentioned services. Some of the methods and algorithms are worth mentioning, such as the process of heatmap generation and the implemented clustering algorithms.

A. *Heatmap*

A heatmap is a graphic representation of a data set, where the differences between values are displayed using different colors. Heatmaps are often used for visualization, for example indicating geographical heights or for visualizing differences in statistical data.

In the framework of the RegionRank project heatmap is used for visualizing the aggregated effect of ROIs in a specified location. The ROIs can be defined by four parameters: type (geographic coordinate – e.g. representing a POI, polyline – e.g. representing a road, or a polygon – e.g. representing a park), position (the coordinates of the points defining the shape corresponding to the ROI), radius (the radius of the area in which the ROI has an effect) and value (the magnitude of the ROIs’ effect). Using these parameters, the application generates a heatmap, which displays the overall effect of the visible ROIs.

The color corresponding to a point on the heatmap is the sum of the ROIs’ effects on that geographical point. The effect of a ROI in a point depends on the distance between the point and the ROI itself. The sum of these effects can be mapped to a certain color. The shape of a ROI’s effect is influenced by the type of the ROI, as represented in figure 2.

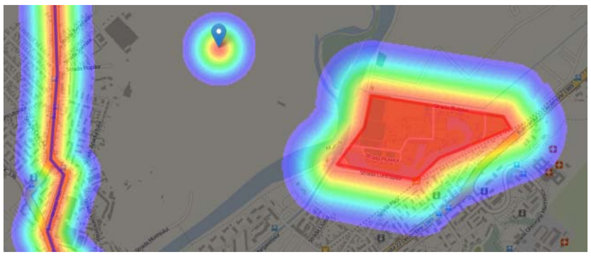


Figure 2. ROIs’ effect on the environment. From left to right: polyline`s effect (e.g. road or river), point`s effect (e.g. cinema or restaurant), polygon`s effect (e.g. park or region with specific crime rate).

RegionRank uses the Leaflet API to implement the map service, which gets image tiles mapped by special URLs from a Tile Service. The URL contains a set of parameters such as the position of the requested tile, the zoom level and other optional parameters which contain additional information about the search (e.g. selected categories/ criteria). Using the request parameters the Tile Service returns a certain part of the map in the form of a PNG image with a resolution of 256x256. This image is called a tile. The server stores tiles for different zoom levels (19 zoom levels), covering the whole world (1 tile for zoom level 0, 2n × 2n tiles for zoom level n, 68 719 476 736 tiles for zoom level 18).

B. *Cluster analysis and POI/ROI clustering*

During the clustering process [1] the application identifies data groups based on some similarity metrics. A group is generally represented by a prototype vector, the cluster center. This process is often required when dealing with large amount of data, so the number of clustering methods and algorithms is high. Beyond classical approaches, there are also some new methods available (e.g. evolutionary algorithms).

Clustering algorithms can be categorized by several aspects. There are hierarchical and non-hierarchical methods (searching for a certain partition or a complete set of partition hierarchy), dynamic methods (the number of clusters is an output of the method), fuzzy methods (an element can belong to more than one cluster based on values indicating the membership level).

Clustering is often applied for visualizing POIs on a map. Displaying a large amount of POI on a map would make evaluation of the presented data impossible. With the help of clustering algorithms the POIs can be grouped into clusters, optimizing the presentation and overall usability of the map.

RegionRank also uses clustering algorithms, but instead of clustering only geographical points, it also groups ROIs represented by abstract shapes. The comparison is based on the geographical distance between these ROIs.

RegionRank`s clustering component is developed using the well-known Strategy design pattern, so it can provide different clustering algorithms, making it easily extensible with new clustering methods. Currently the application provides an implementation for the *k*-means algorithm [1] and a grid-based clustering method [2]. Upcoming versions of the application are planned to implement an improved version of a hierarchical *k*-means-type method and a dynamic evolutionary clustering method.

The main advantage of the standard *k*-means algorithm is its simplicity and speed, even so RegionRank only uses it as a secondary clustering algorithm, because the need to initially specify the number of clusters. In the case of RegionRank, given the changing nature of the data and real-time calculations this can add up to an unexpected behavior and the need to recalculate the clusters.

In the case of map services for displaying POIs, grid-based clustering methods are widely used, providing a different approach. The search space is divided into boxes. The points that need to be clustered belong to one of these boxes.

When clustering POIs, the initial clusters are cells represented by rectangles or squares, partitioning the visible part of the map. The markers are grouped in these cells, so there is no need to calculate and evaluate the distance between the points. For example a grid-based clustering algorithm is used by the Google Maps MarkerClusterer tool. These clustering algorithms have some disadvantages. For example, adjacent points situated very close to the edges of the cells may be associated to different cells/clusters. On the other hand, grid-based clustering algorithms are very effective, and optimal from the perspective of visualization. This approach gave the best result in the case of RegionRank, too.

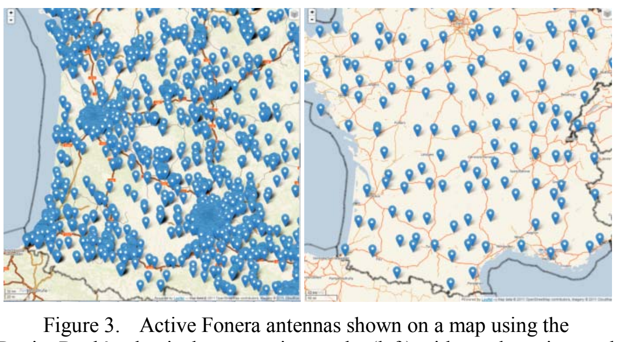


Figure 3. Active Fonera antennas shown on a map using the RegionRank`s classical presentation mode, (left) without clustering and (right) using grid-based clustering (used datasource: http://poiplaza.com)

III. Technologies

RegionRank is developed using Java technologies.

The project is based on the Spring enterprise framework [3]. Thanks to the IoC (Inversion of Control) container, using dependency injection, the components of the application are loosely-coupled.

The persistence is provided by MongoDB [4], a document-oriented NoSQL database, providing the possibility of creating 2d indexes and the option of using multiple, shared databases. MongoDB fulfills the needs of the application, it can handle large data sets in a fast and effective way and it is also flexible, being able to adapt to model changes.

The web-based user interface is created using Vaadin [5], an open-source Java-based web-application framework, providing easy component handling.

The map service uses the Leaflet JavaScript API, which is well documented and offers a unified API for reaching and using different map providers. Thanks to its architecture, Leaflet can be easily extended with third-party plugins or components.

Google`s Guava EventBus is a publish-subscribe framework that offers easy communication between processes and components, and is reliable for notifying different subsystems of the application about events.

Logging is implemented using SLF4J API, a facade to many logger implementations, for example log4j, which is used by RegionRank.

RegionRank uses Apache Shiro, a powerful security framework that handles user rights, authentication and sessions. Shiro can handle different types of data sources (e.g. LDAP, Active Directory, JDBC) providing information about the users, and can be extended with adapters to work with special, non-standard data sources.

Data stored in the database is first validated according to JSR-303 standard, using Java Bean Validation framework and annotations.

Public services are offered through a REST API, using a JAX-RS (Java API for RESTful Services) implementation, the Jersey framework. Some components that use Jersey are responsible for providing the necessary tiles for the Leaflet-based map component and others offer the possibility of importing and exporting data. The import and export functionality is made possible using JAX-B API`s implementation, the Jackson framework, providing and accepting data in JSON/XML formats.

IV. Tools

RegionRank uses Mercurial as source control tool, which is an open-source, free distributed version control system, written in Python.

During the development of the project BitBucket served as issue- and bug tracking system, which is a web-based project management solution, and also provides version control services (supporting Mercurial and Git, too).

RegionRank`s compilation and build processes are described and executed using Apache Maven, a build management system, which grants excellent dependency management, plugin management and deployment automation capabilities, thus promoting the component oriented development.

For ensuring code quality the Sonar, a freely available, open-source code quality tool was used. Its responsibility is to search for duplicates, calculate test-coverage and complexity measures, check for common bad-practices and ensure code conventions.

As a web-application container RegionRank`s developers could choose freely between Apache Tomcat or Jetty.

V. REGIONRANK PROJECT

A. Requirements

The following part shortly describes some of RegionRank's functional requirements, does not cover non-functional requirements, and does not contain a detailed description for functional ones. The detailed specifications are part of the project`s full documentation, and are not included in this article.

The RegionRank server is responsible for providing services used by RegionRank Admin UI, RegionRank Client UI and RegionRank API, including the data access layer and implementing business logic. For example: saving and retrieving model objects to- and from database, parsing different incoming file formats (XML, CSV, etc.), ROI clustering, security.

The RegionRank Admin UI subsystem is responsible for providing an easy-to-use administration interface for the system`s maintenance personal, offering the possibility to manipulate system data, such as regions, categories, ROIs, data sources, etc.

The role of the RegionRank Client UI subsystem is to implement the interface for common users (users without administration rights), where they can easily build their search queries, set the importance of each criterion, and display the result as a heatmap according to the search request. There is also a possibility for simple ROI visualization, filtering and clustering.

The RegionRank Widgetset consists of special components, contains classes responsible for displaying the heatmap, and a Shape Editor for visually editing ROIs on a map.

The RegionRank API is responsible for offering publicly available services and serves as a communication interface. Its main responsibilities: returning heatmap data for certain requests to the Leaflet API, providing a unified API for importing and exporting data.

B. Architecture

Figure 4 describes RegionRank`s subsystems and the communication between these subsystems.

RegionRank stores persistent data in a document-oriented NoSQL database, offering a flexible approach to handling data. Formats can be easily extended or changed, without compromising the integrity of the application.

RegionRank Model contains the core entities of the project. These are used by every subsystem, so each one depends on them. The model classes do not implement business logic, besides data validation.

The responsibility of the RegionRank Repository Layer is to save, load and delete objects from the database. This layer is based on the Spring DATA-MONGODB component provided by the Spring framework.

RegionRank Service Layer offers services used by other subsystems. The implementations of these services are provided by the RegionRank Business Logic Layer. This subsystem directly communicates with the repository layer, which provides the necessary data for the operations. The results of these operations are returned through the service interfaces defined in the Service Layer. Thanks to the dependency injection design pattern and Spring framework, the actual implementations of these services can easily be replaced with new ones, without compromising the architecture of the application.

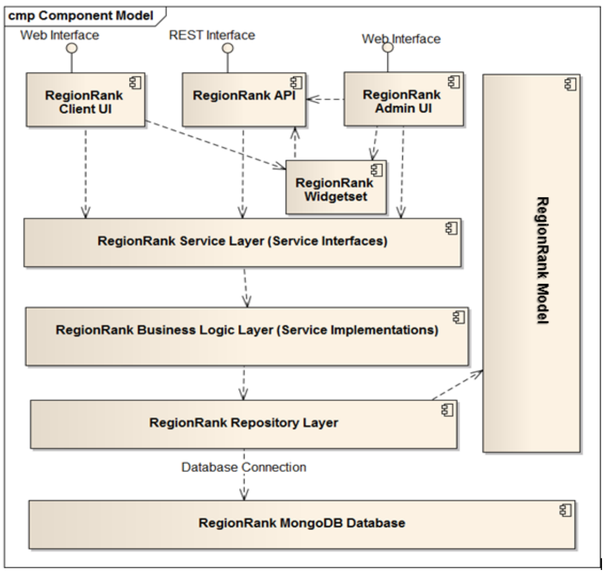


Figure 4. RegionRank`s architecture

The RegionRank Client UI subsystem is responsible for serving the needs of the users. It builds the graphical user interface, evaluates and sends user requests to the Service Layer, displays the returned results in an easily analyzable way. It communicates with the user through HTTP protocol.

RegionRank Admin UI is similar to the Client UI, only it brings additional functionality targeting system administrators, such as system maintenance tasks, application-wide configurations, data manipulation capability.

RegionRank API`s duty is to provide the possibility of communication with the software through a REST protocol. Currently the TileService and the DataService components are implemented. The first is responsible for serving Leaflet API with heatmap data, and the second one offers a possibility to import and export data. Both functionalities rely on JAX-RS standard and JAX-B standard, using Jersey and Jackson as implementations.

The RegionRank Widgetset module contains the special components that cannot be found in the Vaadin framework. For example, the ShapeEditor component designed for system administrators, which offers easy, visual handling of ROIs. Another part of the Widgetset is the map component that displays ROIs and heatmaps.

C. Heatmap generation

RegionRank generates heatmaps through a REST-based TileService. The request for a certain tile is made by accessing a special URL, which encapsulates the coordinates, zoom level and search criteria with their weights. Given the parameters, the tile service calculates the bounding box surrounding the coordinate. Given the boundaries of the box, the service calculates the aggregated effect of the ROIs participating in the search inside the tile.

The ROI`s effect to a given point is calculated considering the distance to that point. The component that calculates the effects scales the distances to an interval between [0,1] and maps a color to the given point according to this value, using a color map.

The module is implemented using the Strategy [6] design pattern, so the calculation method can be easily changed. There are different approaches to determine the value of an effect. RegionRank currently provides two implementations:

1) The distance is scaled with a linear function:

Distance = (radius – distanceOf(Point(i, j), roi)) / radius,

where radius is the ROIs radius, distanceOf(a, b) returns the distance between a and b, Point(i, j) returns the coordinates of the point (i, j).

2) The distance is scaled using a Gauss function:

$$\text{Distance} = e^{-\alpha*d*d},$$

$$d = (\text{radius} - \text{distanceOf}(\text{Point}(i, j), \text{roi})) / \text{radius},$$

where radius is the ROIs radius, distanceOf(a, b) returns the distance between a and b, Point(i, j) returns the coordinates of the point (i, j) and α is a scale parameter.

The business logic module calculates and summarizes the effects of each ROI which distance to the given point is less than the radius of the ROI. The calculation is made using a distance function, and the color is defined accordingly.

If the ROI is a geographical point, the distance from the point will be the geographical distance between the point and the coordinate. In the case of a polyline ROI, the distance is calculated using the minimum of the following distances:

- the distance of the point from the nearest node of the ROI;
- the distance of the point from a segment of the ROI, if the projection of the point is on that segment.

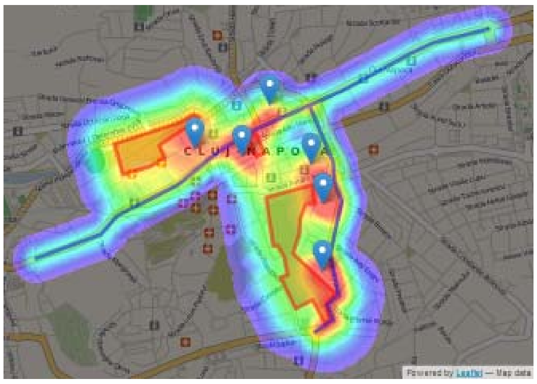


Figure 5. summarized effect of ROIs on a region (the data set contains ROIs of the following types: 6 points, 2 polygons and 2 polylines).

In the case of polygon ROIs, the algorithm checks if the point is inside the polygon. If it is so, the ROI`s effect in that point will be maximal. If the point is outside of the ROI, the same algorithm is used like in the case of polyline ROI.

D. Case study

For presenting the functionalities of the RegionRank project, an example is considered. A group of tourists visiting Cluj-Napoca would like to serve a meal, close to a main road, after which they would like to take a walk in a park. So, they would search for a region of the city which is close to a main road, has some green areas and a restaurant nearby.

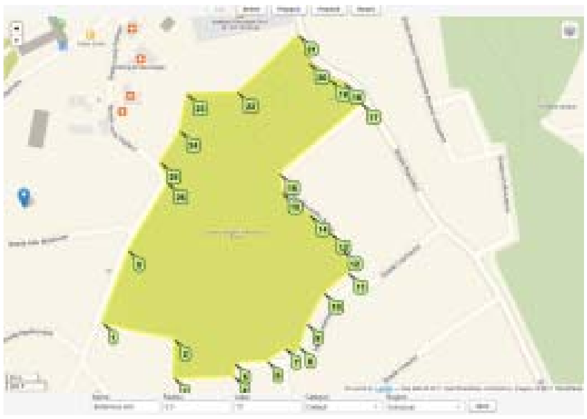
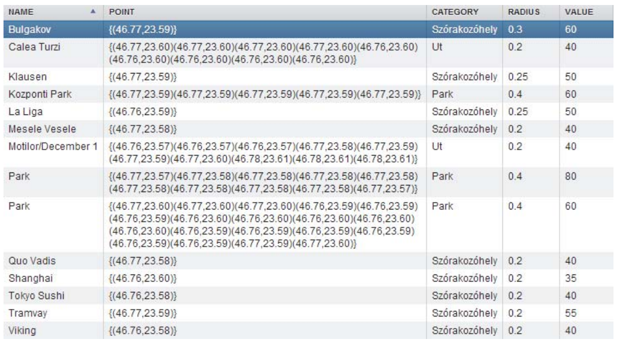


Figure 6. ROI Shape Editor on the admin user interface

The used database contains a data set providing categories corresponding to the certain ROIs. For each ROI the application stores the necessary coordinates to uniquely define the shape and location of the ROI, the radius of the ROI, a value which shows the magnitude of the ROI's effect and other optional meta-data. The administrators can list the ROIs in a table that provides easy management of these as seen in figure 7.



NAME ▲	POINT	CATEGORY	RADIUS	VALUE
Bulgakov	{(46.77,23.59)}	Szórakozóhely	0.3	60
Calea Turzi	{(46.77,23.60)(46.77,23.60)(46.77,23.60)(46.77,23.60)(46.76,23.60)(46.76,23.60)(46.76,23.60)(46.76,23.60)(46.76,23.60)}	Ut	0.2	40
Klausen	{(46.77,23.59)}	Szórakozóhely	0.25	50
Kozponti Park	{(46.77,23.59)(46.77,23.59)(46.77,23.59)(46.77,23.59)(46.77,23.59)}	Park	0.4	60
La Liga	{(46.76,23.59)}	Szórakozóhely	0.25	50
Mesele Vesele	{(46.77,23.58)}	Szórakozóhely	0.2	40
Motilor/December 1	{(46.76,23.57)(46.76,23.57)(46.76,23.57)(46.77,23.58)(46.77,23.59)(46.77,23.59)(46.77,23.60)(46.78,23.61)(46.78,23.61)(46.78,23.61)}	Ut	0.2	40
Park	{(46.77,23.57)(46.77,23.58)(46.77,23.58)(46.77,23.58)(46.77,23.58)(46.77,23.58)(46.77,23.58)(46.77,23.58)(46.77,23.58)(46.77,23.57)}	Park	0.4	80
Park	{(46.77,23.60)(46.77,23.60)(46.77,23.60)(46.76,23.59)(46.76,23.59)(46.76,23.59)(46.76,23.60)(46.76,23.60)(46.76,23.60)(46.76,23.60)(46.76,23.60)(46.76,23.59)(46.76,23.59)(46.76,23.59)(46.76,23.59)(46.76,23.59)(46.76,23.59)(46.77,23.59)(46.77,23.60)}	Park	0.4	60
Quo Vadis	{(46.77,23.58)}	Szórakozóhely	0.2	40
Shanghai	{(46.76,23.60)}	Szórakozóhely	0.2	35
Tokyo Sushi	{(46.76,23.58)}	Szórakozóhely	0.2	40
Tramvay	{(46.77,23.59)}	Szórakozóhely	0.2	55
Viking	{(46.76,23.58)}	Szórakozóhely	0.2	40

Figure 7. Table of ROIs

The user interface offers the possibility for switching between heatmap mode and classical visualization mode, as presented in figure 8. This interface provides the features of ROI clustering, result filtering and the user can also acquire additional information when clicking on a ROI.

Users can select the desired categories, which are included in the search and can set the importance of each category/criterion according to their needs. The result is returned in the form of a heatmap. Those parts of the heatmap that are more desirable present higher values (warmer colors) than those that are undesirable (colder colors). The default color map can also be easily changed.

Figure 8 shows RegionRank`s answer for a search considering the criteria of the presented case study.

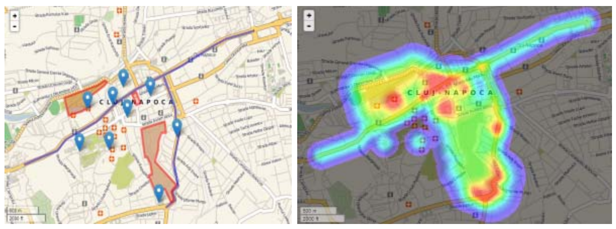


Figure 8. (left) ROIs on a map and (right) the generated heatmap

VI. CONCLUSIONS AND FURTHER DEVELOPMENT

RegionRank implements a web search service based on a new approach. The project extends the classical POI concept and introduces the generalized concept of ROI. The application supports various data formats for import and export, implements clustering algorithms and can handle different map providers.

The project achieved to give easily evaluable and analyzable results in the form of heatmaps on an easy-to-use web interface. When dealing with a complex search having multiple criteria, a user is not forced to do several searches for each criterion, and then summarize the results, because RegionRank aggregates the criteria and presents the result in an easily understandable way.

RegionRank in its current state is a prototype that fulfills the requirements of the initial specifications. There are several further development possibilities. For a better evaluation of the user`s needs a social ranking system should be implemented. In this way, users could rate ROIs and send feedback to the system after getting the desired heatmap, so the application could generate more accurate results. RegionRank can be easily transformed into a cloud-based application using a SaaS (Software as a Service) platform. The application could be extended with crawler modules, semi-automatically collecting publicly available data from the Internet, based on given templates.

REFERENCES

[1] G. Gan, C. Ma, J. Wu, *Data Clustering: Theory, Algorithms, and Applications*, Philadelphia, Pennsylvania, SIAM, Society for Industrial and Applied Mathematics, 2007

[2] J. Han, M. Kambe, J. Pei, *Data Mining: Concepts and Techniques* 3th ed, Morgan Kaufmann, 2011.

[3] R. Johnson, J. Hoeller and co., (2004-2012) Spring Framework Reference Documentation [Online]. Available: http://static.springsource.org/

[4] ***, (2013) MongoDB Documentation [Online]. Available: http://docs.mongodb.org/manual/

[5] M. Grönroos, *Book of Vaadin*: 4th ed, Turku, Finland, Vaadin LTD, 2013.

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 1994.