technische universität
dortmund

Bachelor Thesis

Mining of geo-tagged Word-Clouds from bibliographic data in computer science

Stefan Joschko - van Ackern
July 2016

Supervisors:
Prof. Dr. Kristian Kersting
M.Sc. Alejandro Molina

TU Dortmund University
Department of Computer Science
Chair for Artificial Intelligence (Chair 8)
www-ai.cs.uni-dortmund.de

Contents

Chapter 1

Motivation: The Atlas of Scientific Research

Computer science is a relative young science. Nevertheless, it is a large and rapidly growing one. Over the last years computer science has seen a extreme growth. The number of scientific fields, science journals, proceedings and papers is increasing every day. The DBLP database used in this thesis (refer section 2.1.1) has stored over 200 thousand new computer science related publications each year since 2009. This are over 500 publications each day. Therefore it has become especially difficult to keep track of different computer science topics and research areas. It is also difficult to get an overview on how computer science is progressing in different areas of the world. For instance, a European student that wants to study computer science in the USA will have a trouble deciding in which region of the USA he or she should study, based on interest. A database like DBLP that stores millions of publications could help that student to decide, but the sheer amount of available content makes a decision, just by looking at the data, almost impossible. However with *machine learning* algorithms it is possible to extract representative keywords for different areas in the world. Since almost every publication is published under some kind of affiliation to a university, a company or an institute and because those affiliations have a fixed geo-location, it is possible to geo-reference such publications. Extracted keywords from those publications then serve as a representation of the scientific research at their corresponding location. A web based world map can then show those keywords at their estimated locations, grouped together in *word-clouds*. This approach is of course not limited to computer science and cloud also be used to show keywords for other science areas or even for completely different geo-referenced datasets. However, the application presented in this thesis, called *The Scientific Research Atlas*, will focus on computer science related informations.

1.1 Goals of this Thesis

The goal of this work is to give users a tool to quickly gain an impression of the current computer science research in different areas of the world. The application is web based and free to use. It shows world-clouds that visualize the scientific research in different areas of the world and is based on free open geo-referenced bibliographic data. In particular the implemented application should fulfill the following tasks:

Easy to use The tool is not meant for experts and should be quick and easy to use, without having the user deal with complicated controls.

Representative The shown word-clouds should represent the scientific research of their corresponding area as best as possible and should only contain meaningful keywords.

Comperable Different word-clouds that are displayed at the same time should highlight similarities and differences between them.

Dynamicly processed To enable users to freely explore the world map, the displayed word-clouds should be computed dynamically, to give users the best possible result for every arbitrary area.

Meaningfull geographical clustering Word-clouds should represent geographically meaningful areas. In the best case even border bounded areas like continents, states, countries and cities. The number of shown word-clouds should depend on the structure of the given area. It should neither be too low nor to high.

1.2 Related Work

Word-clouds, also called *tag-clouds*, have become popular in recent years, especially with the grows of the Internet, thanks to their aesthetically pleasing way to visualize large texts and word collections. They are, for instance, often used to display content of community-oriented websites like Twitter, YouTube, Reddit or Flickr. One of the biggest websites that offers free word-cloud generation is *Wordle* [14], which has made it very simple for anyone to generate word-clouds with own data. With its help, people have generated millions of publicly posted word-clouds. The Journal *Participatory Visualization with Wordle* [32] tries to examine the source of Wordle's popularity and also discusses Wordle-layouts that lead to differently looking word-clouds by changing their typography, color, and composition. Steffen Lohmann, Jürgen Ziegler, and Lena Tetzlaff [22] looked at the ability of different word-cloud layouts to support users in typical information seeking task like finding a specific tag, finding the most popular tags and finding tags that belong to a certain topic. They conclude that words with large text size and words that are near the center usually attract more attention, thus that tag-clouds provide suboptimal support

when searching for specific tags. This behavior is not ideal since users should be able to compare word-clouds in the research atlas. This problem gets partially addressed by the so called *Word-Storm* that Quim Castellà and Charles Sutton [11] have proposed. In a word-storm, words that appear in different word-clouds, have approximately the same position as well as the same color. Therefor it is more obvious for users, if a word appears in multiple word-clouds and how similar to word-clouds are overall. Word-storms are great to compare keywords from different documents.

In 2007 Shane Ahern, Mor Naaman, Rahul Nair, and Jeannie Yang from Yahoo! Research Berkeley developed an application called *World Explorer*. It is a web based visualization tool that uses a map interface to show representative tags for arbitrary locations. That way it serves as a kind of city guide and helps users find interesting locations all over the world. The data is taken from the website *Flickr*, which allows users to upload pictures, assign a geo-location and write a short description. Those arbitrary descriptions were used to generate the tags. Therefore generated tags are completely based on community created content. In the corresponding paper [5] they also discuss the benefits of browsing through generated keywords with the help of a world map. They conclude that their application does a good job in giving users an overview over unknown areas and also helps them remember back to places where their already been. However searching for a specific tag was often difficult for users, which in the paper is described as "the needle in the haystack" problem. Nonetheless most users seemed to use the map controls dragging and zooming quiet intuitively. In a second paper [18] they also explain which data structure and algorithms they use to summarize their large collection of geo-referenced data. They introduce a hierarchical clustering algorithm as well as a method of ranking photographs for different zoom levels. Yet, this computation is done as a prepossessing step and not dynamically.

1.3 Structure

The upcoming chapter 2 will explain the creation of the scientific research atlas in detail. This includes the three steps Data Preparation, Word Selection and Visualization. Accordingly, section 2.1 introduces the used databases, section 2.2 shows how keywords are extracted for arbitrary given areas and section 2.3 gives insights into the implemented visualization. Chapter 3 illustrates results generated by the scientific research atlas and makes assumptions about the quality of created word-clouds. Furthermore it evaluates the contained keywords with help of different coherence measures. Chapter 4 concludes the creation of the scientific research atlas and suggests future work that could enhance the application.

Chapter 2

How to Generate the Atlas

This chapter explains the specific technologies and algorithms that were used, along with the design choices that have been made, to create the atlas. The development process of the scientific research atlas can be divided into 3 basic steps. At first geo-referenced informations need to be gathered, which are usually stored in relational databases or in plain text files. Texts will be in a raw format, which reveals the demand to tokenize and filter such data, in order to make it usable for further classification. This first step is explained in section 2.1. Only the most important words should appear on the map, hence the system needs a way to pick words out of the available pool of words. Accordingly section 2.2 examines different approaches to rank words based on their importance. The last step explains the visualization and is content of section 2.3. It unfolds a closer look at the technologies that were used to build the web based world map and explains how word-clouds are generated. Additionally it shows the design choices concerning the user interface and how they contribute to a smooth user experience along with a pleasing design.

2.1 Step 1: Data Preparation

The Digital Bibliography & Library Project (DBLP) database, which is used to extract the needed keywords, consist over 90% out of conference and journal posts, as well as informations about authors. Details about the used data are shown in Section 2.1.1. Since DBLP holds over three million publications, stored texts need to be tokenized and filtered to make them usable for later algorithms. This process is explained in Section 2.1.2. At last, the filtered words need to be mapped to their corresponding geo-location. The system needs a fast way to decide which words are contained in a arbitrary given area. Furthermore to show more than one word-cloud at a time, this area needs to be divided into geographically bounded groups, which get represented by a word-cloud.

2.1.1 Used Data

This thesis will focus on data related to computer science. However the application is not limited to computer science data and one could use other geo-referenced datasets to visualize different informations. The database used for the research atlas in this thesis is called DBLP [2]. This database is a bibliography for open bibliographic information on computer science journals and proceedings. It was created 1993 at the university of Trier. The database has continued to grow over time and holds over 1.33 million journal articles and 1.78 million conference/workshop papers today. Since 2009 DBLP has stored over 200 thousand new publications each year. Besides that it also stores informations about over 3.3 million authors. However it does not contain any geographical data or informations about affiliations. In 2013 scientists used different data mining techniques to create a DBLP version that connects papers and authors with corresponding affiliations. The related paper *GeoDBLP: Geo-Tagging DBLP for Mining the Sociology of Computer Science* [15] explains, the creation in detail. The resulting GeoDBLP database assigns a specific geo-location to each publication and therefore makes it possible to create the scientific research atlas in the first place. However the only usable text from a stored publication is the title. On the one hand titles are a very compressed explanation of the content and therefore contain important words for the described publication. On the other hand a title does not always contain all important words and most words usually only appear once. Often a title is not even a complete sentence and the order of words does not represent their semantic connection in every case. Consequently it would be beneficial to have more available text than just the title, to allow deeper analyses that take into account semantic relations between words. To solve this problem a second database is used. The Aminer Citation Network Dataset [31] [1] also stores publications from DBLP, but in addition, it contains the abstracts for many publications. Considering that abstracts are a short textual explanation of a whole publication, they deliver more information than just a title by itself. Besides the abstract the Aminer Citation Network Dataset also contains the citations for each document. Those citations could be used to decide how important a publication is, since important publications tend to get cited a lot. Yet, DBLP and the Citation Network are two different databases and inconveniently don't share some kind of direct identifier between publications, which would make it easy to connect entries from both databases with each other. The solution is to join papers on their titles directly. But since titles seem to differ between the databases in some cases, all titles were first converted to lowercase and all punctuation was removed. Then, entries from the different databases that share the exact same title were joined together. It turns out that the Citation Network does not contain all publications in DBLP. The connected dataset contains about 0.72 million publications, that now have a title as well as an abstract. This is about $\frac{1}{4}$ of all publications. However, the amount of total words is about 5 times greater than before.

Therefore this dataset was chosen over the GeoDBLP database that only holds titles. In summary, the resulting used dataset holds 0.72 million entries that each store informations about the title, the author, and the publishing year of a publication as well as the affiliation that it was published on together with the corresponding geo-location.

2.1.2 Prepossessing

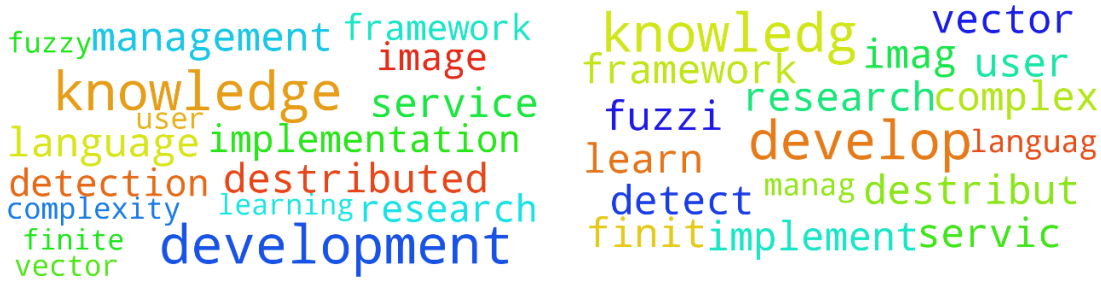
Each raw title and abstract from DBLP is stored as a single string. Besides the words they also hold whitespaces, numbers and punctuation. In order to make single words available for the later algorithms, those texts need to be tokenized and filtered. The first step of the tokenization is to remove any punctuation from all texts. Next, all words are converted to lowercase. This is important seeing that later algorithms would consider same words with varying case-sensitive spelling as two different words, otherwise. The last step is to split words at whitespaces to single strings. This completes the tokenization. The 0.72 million titles and abstracts together hold around 0.4 million unique words and 106 million words in total. A lot of words can be filtered out to decrease the amount of needed storage and the execution time of later algorithms, thus to improve the results by filtering meaningless words. Three different types of words were filtered:

Non-English words About 80% of the all words are contained by the Natural Language Toolkit (NLTK) words corpus [13, Chapter 2, Section 4], which is basically a list of almost every existing English word. In consideration that it makes no sense to work with more than one language at a time and because English is by far the most common language in the word collection, all non-English words were removed.

Stop words Stop words are words that appear very often throughout texts but are meaningless in terms of information retrieval. Some examples could be: as, we, your, she, is, etc. A list of stop words can be derived from a given collection itself [21], however the most common and simple approach is to use a prebuild list. Accordingly stop words were removed from all texts with the help of the English NLTK stop words corpus [13, Chapter 2, Section 4].

Too frequent/too seldom words Some words appear very rarely in all publications. Those can be considered as not important. For this reason every word that appears in less than 5 publications gets filtered out. Also, words that appear in more than 8% of all documents get filtered out. Those words are usually so common that they can be considered as a kind of stop word as well. The number of 8% is derived through testing, by reason that this containment only removes fairly meaningless words like: approach, high, based, different etc. (The full list can be found in the appendix).

Stemming was not applied to the words, simply because stemmed words usually don't look appealing in word-clouds, as Figure 2.1 illustrates. After the filtering, the amount of



(a) **(b)**

Figure 2.1: Image **(a)** shows a word-cloud with normal words. Image **(b)** shows the same words but stemmed. While some words stay the same like "user", other words like "imag" or "manag" look very unnatural and are difficult to understand, neither look very appealing.

	Total words	Unique words
Original word collection without filtering	106 736 199	417 518
After removing non-English words	86 742 118	42 134
After removing stop words	46 898 847	42 000
A. r. words that appear less than 5 times between documents	46 853 524	23 750
A. r. words that appear in more than 8% of all documents	36 911 964	23 691

Table 2.1: The number of remaining words after each filtering step. Note that the amount of unique words is about 10 times smaller after removing all non-English words from the original word collection, while the number of total words only gets reduced by about 20%. After that the number of words almost gets sliced in half, by only removing 134 stop words. Removing rare words has a high impact on the unique words, while having almost no impact on the total words. When removing too frequent words, the opposite is the case.

unique words shrunken down to about 23 thousand words. Table 2.1 shows the word counts throughout the filtering process. In the end the data was reduced to about $\frac{1}{3}$ of its original size. However, the amount of unique words was reduced to about $\frac{1}{17}$, which shows that a lot of unique words were quite seldom or non-English and accordingly are considered to be unimportant in the first place. Removing too frequent words seemed to have a positive influence on later algorithms, since those filtered 59 words appeared almost 10 million times throughout the word collection and therefore had a high value in most scorings, despite being almost meaningless. One problem that arises is that some words don't make sense on their own and need to be displayed in context to their surrounding words. One example is the phrase "support vector machine". A single word on its own does not tell a lot, while all three words combined stand for an important research topic in computer science. It makes sense to connect those words together into one phrase. Further more, the word "vector" is used in a lot of different contexts in computer science, that are not directly related to each other. By inserting the word into phrases, those different contexts can be distinguished

raw	"Statistical Relational Artificial Intelligence: Logic, Probability, and Computation."
punctuation	"Statistical Relational Artificial Intelligence Logic Probability and Computation"
lowercase	"statistical relational artificial intelligence logic probability and computation"
tokenize	"statistical","relational","artificial","intelligence","logic","probability","and","computation"
stop words	"statistical","relational","artificial","intelligence","logic","probability","computation"
phrases	"statistical","relational","artificial_intelligence","logic","probability","computation"

Table 2.2: This table shows the tokenization and filtering of an example title. Note that the identifying of phrases depends on the neuronal network and its training data. In addition, words that appear too frequent, too seldom, or are non-English will also be removed afterwards.

from each other, since later algorithms will consider for example "vector_approximation" and "support_vector_machine" as two different words. To build such phrases a neuronal network as described in [25] can be used. First it needs to be trained with all existing titles and abstracts and is then able to identify common phrases in every given sentence. The intuition is that words which appear often together and seldom on their own probably are related to each other. In behalf of the reason that those key phrases need to be displayed on limited space, only phrases containing two and three words were created. Table 2.2 shows an example of all prepossessing steps described in this section. This completes all preprocessing steps.

2.1.3 Clustering

The quality of the research atlas is strongly dependent on the quality of the underlying geo-referenced data. The geo-position for a given publication comes, in case of the geoDBLP database, from its corresponding affiliation. Most affiliations obviously publish more than one publication, hence some affiliations are located in the same place. So not every publication has a unique location. In fact, the 0.72 million publications correspond to around 13 thousand unique positions. Figure 2.2 shows the distribution of those locations over the world. It reveals that most affiliations are located in Europe, North America, and some regions of Asia. In those regions exists enough data to generate meaningful word-clouds, while in other regions, like Africa, the amount of usable data is very small or not existing at all. Despite the fact that later algorithms will only take into account the location and texts of a given publication and not the affiliation, it is sufficient to do any geographical computation with those 13 thousand unique locations. In this case latitude and longitude, serve as a reference to identify which publications correspond to which position. An aspect of the scientific research atlas is to give users the ability to compare different areas with each other. For this reason an arbitrarily given area needs to be divided in at least two parts in order to show multiple word-clouds. To make the parts as meaningful as possible a *cluster* algorithm can be used to identify groupings of locations. Clustering is the task to group given observations in a way that objects contained in the same group are more

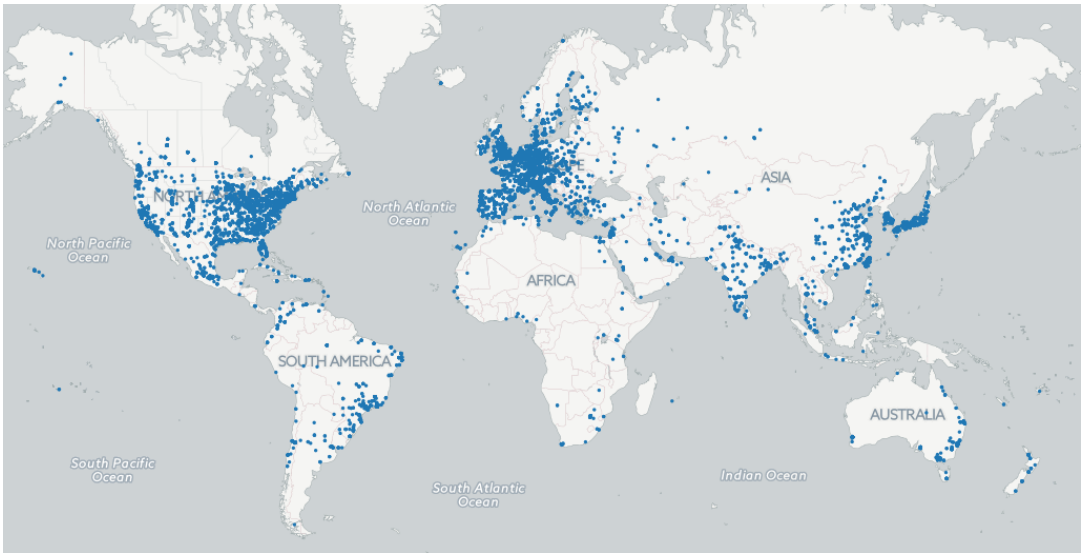


Figure 2.2: The geo-locations of all 13 thousand unique locations. Each location is represented by a blue dot. Most affiliations are located in Europe, North America, and parts of Asia.

similar to each other than to objects of other groups. As an example, figure 2.3 shows the unique locations that are located in Europe, clustered into 5 groups. One popular algorithm for cluster analysis is *k-means*.

Lloyd's algorithm

The k-means algorithm aims to partition given observations into k clusters so that each observation belongs to the cluster with the nearest centroid (also called *mean*). The centroid-based clustering optimization problem is NP-hard [6]. Therefore most k-means algorithms try to approximate a solution by computing a local optimum rather than the global optimum. One popular implementation of the k-means algorithm is the *Lloyd's algorithm* [20], which is often referred to as the standard k-means algorithm. Formally, observations and cluster are defined as follows:

- The observations are a set $X = (x_0, x_1, ..., x_n)$, where each observation x_j is a d-dimensional vector
- $\boldsymbol{S} = \{S_1, S_2, ..., S_k\}$ is the set of k clusters
- A cluster S_i is a set of observations $S_i \subset X$

In this case the observations are a 2-dimensional vector consisting of a geo-location represented by latitude and longitude. To cluster those coordinates the Lloyd's algorithm tries to minimize the *within-cluster sum of squares* objective function:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2 \tag{2.1}$$

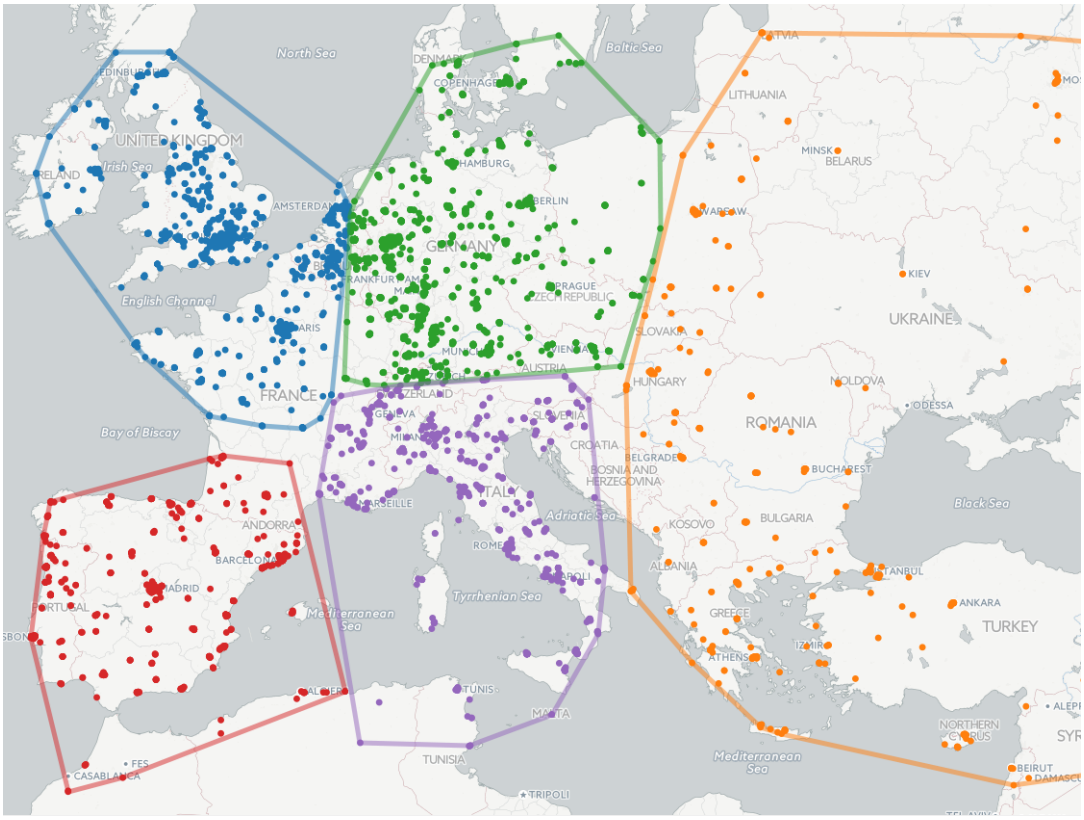


Figure 2.3: West-Europe clustered with k-means, with $k = 5$. Latitude and longitude were treated like euclidean coordinates. It shows that k-means still produces useful clusters.

where $x_j \in X$, $S_i \in \boldsymbol{S}$ and μ_i is the centroid (or mean) of S_i. In other words: The Lloyd's algorithm tries to find the configuration of clusters S, so that the sum of all distances between all observations and their corresponding centroids is minimal. In the present case, the within-cluster sum of squares is equal to the euclidean distance. However geo-locations represented by latitude and longitude on the geographic coordinate system are not euclidean[1]. Treating them as they are euclidean coordinates introduces inaccuracy into the clustering. The distortion is higher at the ice caps and lowest at the equator. In practice treating geo-locations like euclidean coordinates worked well as figure 2.3 shows, since no affiliations exist near the ice caps and clusters are still accurate enough. The Lloyd's algorithm implementation is shown in Algorithm 2.1. At first k random centroids are chosen for the k clusters in $\boldsymbol{S}$. After that, every observation point x_j gets assigned to the cluster with closest centroid. Once all observations are assigned, every cluster gets a new centroid by computing the mean value of all observations that are contained in it. These two steps get repeated until the clusters do not change anymore, or the maximal iterations threshold is exceeded.

[1]The geographic coordinate system of the world is built by latitude and longitude. Longitude lines are perpendicular while latitude lines are parallel to the equator

Algorithm 2.1 K-Means [23]

Input: Observations $X = (x_0, x_1, ..., x_n)$, Cluster Count k, max-iterations

Output: Final set $\boldsymbol{S} = \{S_1, S_2, ..., S_k\}$

$C \leftarrow (c_0, c_1, ..., c_k)$ ▷ choose k random centroids
counter $\leftarrow 0$
while counter $<$ max-iterations $\wedge$ $\boldsymbol{S^{t-1}} \neq \boldsymbol{S}$ **do**
    $\boldsymbol{S^{t-1}} \leftarrow \boldsymbol{S}$ ▷ save clusters to look for changes in next iteration
    counter $\leftarrow$ counter $+ 1$
    for all $S_i \in \boldsymbol{S}$ **do**
        $S_i \leftarrow \{x_j \text{ if } \textsc{closest}(x_j, c_i)\}$ ▷ assigning observations to cluster by distance
    for all $c_i \in C$ **do**
        $c_i \leftarrow \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ ▷ $c_i \leftarrow$ mean value of all observations in S_i
function $\textsc{closest}(x_j, c_i)$
    return $||x_j - c_i||^2 \leq ||x_j - c_p||^2$ for all $p = 1, ..., k$

(a) **(b)**

Figure 2.4: Both regions were clustered with Lloyd's algorithm and $k = 5$. While on **(a)** it produces decent results, on **(b)** the three clusters in the middle are too close together. This will cause problems later on, since shown word-clouds will most likely intersect with each other.

According to [19], the run time of k-means is given as $\mathcal{O}(dkni)$, where d represents the dimension of the observations, n is the number of observations and i the number of iterations. In case of the research atlas, d is 2, n can not get bigger than ∼13000 and i is locked to maximal 200 iterations. Therefore it is ensured that the algorithm is relatively fast.

Determining the number of clusters

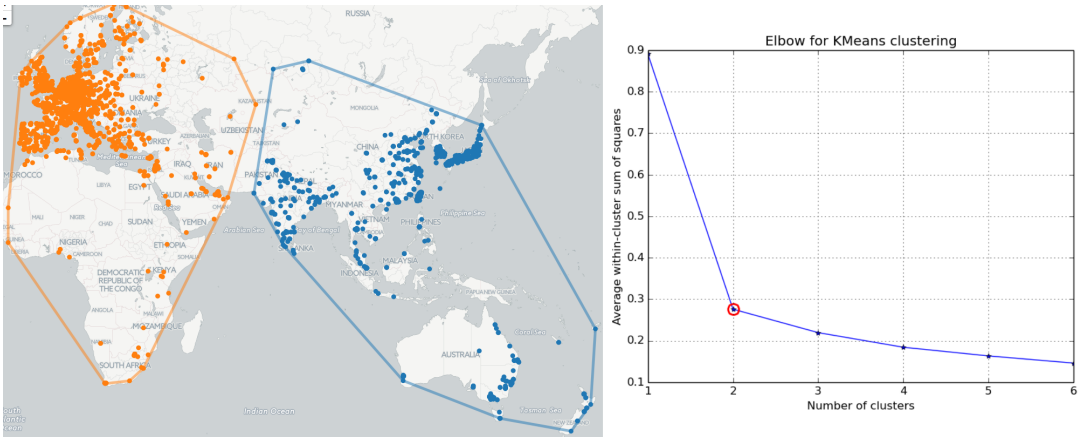
When using Lloyd's algorithm, the question arises on how to choose k. This will then determine how much word-clouds are shown at the same time for a given arbitrary area. Since each area has a different geographical texture as well as a varying number of contained unique locations, ranging from 0 to over 13 thousand, it is not sufficient to choose a static k. This problem is illustrated by figure 2.4. There are several approaches on how to

automatically choose k, given a set of observations. One is to run the clustering algorithm multiple times with a varying k and determine how good a clustering with a certain k is in relation to other clusterings. To compare different clusterings with each other, they can be ranked using a measure like for instance the *Silhouette*-method [12] or *cross-validation* [30]. However, some of those can take several seconds or even minutes to compute, depending on the measure. The method that is used for the research atlas is the *elbow*-method, by reason that it is easy and fast to compute. It uses the average within-cluster sum of squares

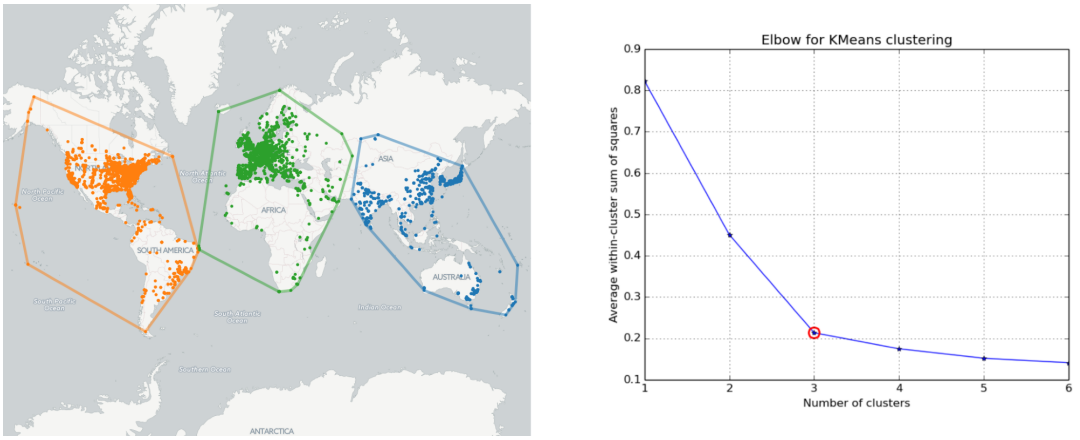
$$\frac{1}{|\boldsymbol{S}|}\sum_{i=1}^{k}\sum_{x_j \in S_i} ||x_j - \mu_i||^2 \tag{2.2}$$

of each clustering for comparison. The lower the average within-cluster sum of squares is, the better is the clustering. The goal is to find a k that gives a great improvement over the algorithm computed with $k-1$, but at the same time being not much worse than the algorithm computed with $k+1$. This can best be described using a visual representation as in figure 2.5. In practice, the Lloyd's algorithm is computed 5 times, with k ranging from $k=2$ up to $k=6$. After that, the angles between the lines get computed and the k with the highest angle in being chosen. However one problem sometimes showed up while testing the atlas. Some areas of the world hold just a very limited amount of locations. In those areas it can happen that all points inside it are grouped in a small space, while the rest of the area is completely empty. An example is Hawaii, which can be found in figure 2.2, far on the left side next to the text "North Pacific Ocean". For Hawaii, the elbow-method estimates that $k=2$ is the best solution, which is the correct and desired behavior when a user zooms close onto it. However, when the user zooms out again to a point where Hawaii is just a little spot on his screen, but still the only spot that holds locations, the algorithm still decides that $k=2$ is the best solution. This leads to the same problem that is shown in figure 2.4b, where clusters are too close together and therefore do not provide enough space to properly display separate word-clouds. To account for that problem a constrain is added to the elbow-method. If the cluster centroids are too close together in comparison to the viewed area, it is not chosen. I this case the algorithm tries to find the second best k and checks the constrain again. This is repeated until a k is found that results into clusters which are not too close together. If no $k>1$ fulfills that, $k=1$ is chosen, which effectively results in no clustering at all. The specific threshold at which two cluster centroid are considered to be too close together, was derived via testing. For the research atlas the threshold is $\frac{1}{6}$ of the diagonal of the whole viewed area.

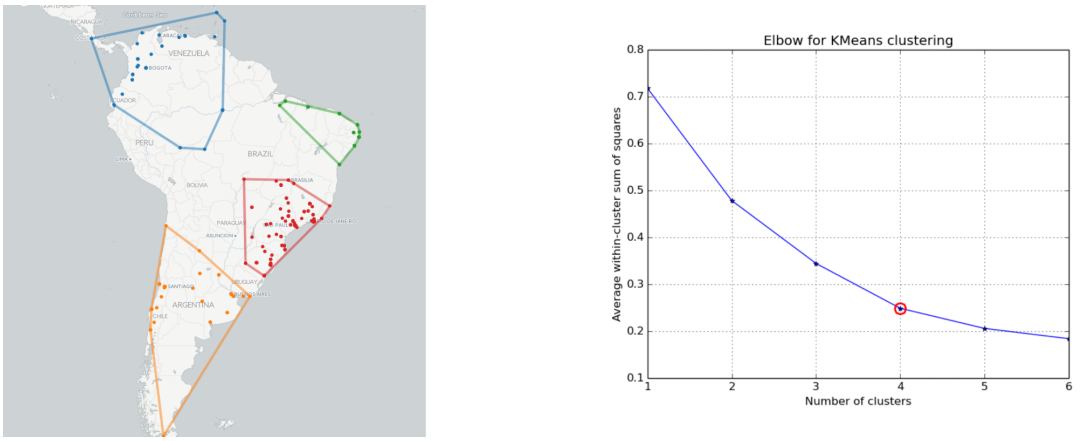
Computing the within-cluster sum of squares is fast because for each clustering with a different k, the distance between each observation and its corresponding centroid has to be computed once. Since in the worst case, the number of observations can not get bigger than $\sim$13 thousand, this turns into a constant. As mentioned above, the run time of k-means is given as $\mathcal{O}(dkni)$, where d represents the dimension of the observations, n



(a)



(b)



(c)

Figure 2.5: Shown are three clustered areas. The diagrams on the right of each image shows the normalized average within-cluster sum of squares for the Lloyd's algorithm executed with $k = 2$ up to $k = 6$. The elbow is the k, where the graph has the highest curvature. This point looks a bit like the elbow of a human arm, which gives it the name elbow-method. On **(a)** and **(b)** the elbow is quite easy to distinguish visually. On **(c)** it is much harder to locate the elbow. The more evenly distributed the locations are, the less pronounced is the elbow.

is the number of observations and i the number of iterations. For the elbow method the k-means algorithm is executed 5 times with k ranging form $k = 2$ up to $k = 6$, which means that the complexity of all 5 runs in total is given by $\mathcal{O}(\sum_{k=2}^{6}(dkni))$. Since d is 2, n can not get higher than ∼13000 and i can not get higher than 200, this ensured that the elbow method is relatively fast.

2.2 Step 2: Word Selection

The selection of keywords is the core of this application. It defines whether or not the application is useful and informative. In section 2.1 is shown how a given area is clustered into k groups. Each of these clusters can consist out of thousands of publications with each holding hundreds of words. An algorithm is needed that is able to rank all of those words by their importance and is also fast enough, since this computation has to be done in real time. Just displaying the most frequent words is not sufficient in most cases since those words are usually not the most representative ones, neither do they highlight the differences between tiled areas. In the best case, keywords for the distinct cluster should highlight similarities as well as differences between the shown areas. When looking at a big area like a continent, clouds should show the general distribution and tendencies of computer science topics. Meanwhile on city level, word-clouds should present more specific keywords, like the research areas of a university. At first the publications and their word need to be converted into a more usable an fast to access representation in order to rank them with algorithms. A common representation for that is the Bag of Words (BOW), which is explained next. Afterwards the ranking of terms is described.

2.2.1 The Bag of Words model

The BOW model is a way to represent multiple documents in a single matrix that is often used in natural language processing and information retrieval. A BOW is defined as follows [9]:

- A corpus is a collection of M documents and defined as $D = \{d_1, d_2, ..., d_M\}$, with d_m beeing the m-th document of the corpus
- T is the set of all terms (words) that occur at least once a corpus D
- The BOW representation of the document d_m is a vector of weights $(t_{1m}, t_{2m}, ..., t_{Nm})$, where $N = |T|$
- t_{nm} is representing the frequency of the n-th term in the m-th document

In other words, the BOW representation transforms a document collection to a matrix, where rows represent document vectors and columns represent terms. A example of this

d_1	"People like computer science"
d_2	"We learn computer science and social science"
d_3	"Computer can learn"

↓

	people	like	computer	science	we	learn	and	social	can
d_1	1	1	1	1	0	0	0	0	0
d_2	0	0	1	2	1	1	1	1	0
d_3	0	0	1	0	0	1	0	0	1

Figure 2.6: The bag of words representation for three documents. Note that the term "science" appears twice in document d_2. All terms that are not contained in a document have the value zero. Therefore each document vector, regardless of its word count, has the length $|T|$.

matrix view is shown in figure 2.6. Since documents are treated like vectors, they now can be compared using classical distance and similarity measures, like for instance term frequency–inverse document frequency (tf-idf). A possible downside of the BOW model is that the corpus tends to get fairly big, since each document vector has the length $|T|$. A matrix with the 0.75 million publications from DBLP and the filtered 25 thousand words would contain 18.75 trillion entries and would take up 75 Gigabytes of disk space, stored as integers[2].

However most entries will be zero, since a document usually only contains a fraction of all words in T. Therefore with a simple trick the needed space can be reduced drastically. A *sparse matrix* is a matrix representation that only stores the entries with a non-zero value. All not explicitly stored entries are considered to hold the value zero. For the DBLP data the sparse matrix is 384 times smaller. An additional optimization can be made concerning the unique location mentioned in section 2.1.2. Considering that publication with the same geo-position will always be in the same cluster, the document vectors can be added together to one vector, which then represents the term frequency (tf) of all words associated with a location, rather than the tf of each publication it self. This reduces the the amount of rows from 0.75 million down to 13 thousand, which again saves a lot of space. With all that in place, the documents are now represented in a way that is first of all smaller, second allows fast access to single word counts, and third allows the computation of similarity measures like tf-idf, in a short period of time.

[2]A integer is considered to take up 32 bit of space

2.2.2 TF-IDF

The tf-idf [29] is a numerical measure that is intended to reflect the importance of a word inside a document. It is often used as a weighting factor in the information retrieval and text mining community. The basic idea behind tf-idf is that words that appear frequently throughout a lot of documents inside a collection, are probably not especially representative for a single document. The words that best represent a document, consequentially are words that appear often inside that single documents but not very often among all other documents. The term frequency $\mathrm{tf}(t, d)$ is the frequency of the term t in document d, which is exactly what the BOW represents. The inverse document frequency (idf) of a term t

$$\mathrm{idf}(t) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right) \tag{2.3}$$

is a measure that is computed by dividing the number of all documents by the number of documents in which the term t appears in. The more frequent a term is throughout documents, the lower its idf is. Because of the logarithm, a term that appears in every documents has the value 0. Since the idf will be computed on a small number of clusters, this is likely to happen with a lot of terms. To account for that the idf is smoothed

$$\text{smooth-idf}(t) = \log\left(1 + \frac{|D|}{|\{d \in D : t \in d\}|}\right) \tag{2.4}$$

by adding 1 to the term. The tf-idf for each term t in each document d is calculated by multiplying the tf with the idf of t.

$$\mathrm{tfidf}(t, d) = \mathrm{tf}(t, d) \times \text{smooth-idf}(t) \tag{2.5}$$

Accordingly, terms that appear often in a single document but not often overall have a high tf-idf. Table 2.3 shows a short example to explain the tf-idf further. The tf-idf has to be computed for each term inside a cluster. The terms with the highest value are then the most important ones and get displayed by the word-clouds. As mentioned before, each row of the BOW matrix represents a location and stores the tf of each term that appears at that location. The tf of each cluster can be computed by adding all rows that represent unique locations contained by that cluster. For k clusters, this results in a new matrix with k rows. So to calculate the clusters the $\sim$13 thousand rows have to be added together. But only a fraction of the entries hold values, which accelerates the process. Each column represents a term, which enables the computation of the idf by counting the zeros in that column. Considering the amount of zeros in the row of term t is l, then the idf of t is $\text{smooth-idf}(t) = \log\left(1 + \frac{k}{k-l}\right)$. When multiplying each value in each row with its corresponding idf, the resulting matrix contains the tf-idf for each term in each document. Figure 2.7 shows how those computation steps look like. As it turns out, the subjective quality of selected keywords with help of tf-idf depends on the size of the given arbitrary area or rather on the amount of available data in it. For a smaller area like for instance

	Text				smooth-idf(t) a	b	c	d	idf(t) a	b	c	d
d_1	aa	b		d	2.8	0.9	0	0.7	2.2	0.4	0	0
d_2			cc	dd	0	0	1.8	1.4	0	0	0.8	0
d_3		b	c	d	0	0.9	0.9	0.7	0	0.4	0.4	0

Table 2.3: The table shows an example of the tf-idf, computed for every term in every document, at first with smooth-idf and second with normal idf. In this table a letter is meant to be a term. The higher a value, the more important is the term for the corresponding document. Even though the values are a bit higher when using smooth-idf, the proportion of the importance between the terms "a", "b" and "c" stays the same. For example the term "a" is very important for document d_1, in both cases, since d_1 is the only document that contains it, hence also twice. The big difference is that term "d" has the value 0 in every document when using the normal idf, by reason that "d" is contained in every document and therefore $\text{idf}(d) = \log\left(\frac{3}{3}\right) = \log(1) = 0$.

one that contains a few cities, the generated keywords are subjectively meaningful (more on the evaluation of generated word-clouds in chapter 3). However, for a large area with a lot of data in it, like the whole world, generated keywords tend to show terms with the highest tf. Furthermore generated word-clouds are very similar. Given the big amount of data, almost every term appears in every cluster and therefore has the same idf. When all terms have the same idf, the tf alone is the deciding factor on how high the tf-idf is for each term. To avoid this behavior, tf-idf is only used for areas containing less than 1000 locations. For larger areas a different method is used. In case of a large area, like a continent, the desired behavior of the scientific research atlas is to show more general keywords.

2.2.3 Using LDA for Word Selection

Latent Dirichlet Allocation (LDA) is a generative statistical model that can be used to analyze collections of texts. It was proposed by David M. Blei, Andrew Y. Ng and Michael I. Jordan [9] and is inspired by the Probabilistic Latent Semantic Analysis (PLSA) model proposed 1999 by Thomas Hofmann [17]. The definition and explanations in this section are taken from the corresponding LDA publication. LDA is a *Mixed-Membership* model, so for a given set of documents containing words, LDA generates K topics, where each document is a mixture of a small number of those topics. Each word contributes a certain amount to one or more topics. The number of topics K has to be chosen manually. Topics are latent, which means they describe an unobservable semantic level. LDA uses the BOW representation and every term t is directly related to document d by probability $p = (t|d)$. However this is a limited representation, as it only reflects the tf. By introducing

		t_1	t_2	t_3	t_4	t_5	t_6
Cluster 1	Location 1	0	0	15	0	32	66
	Location 2	14	0	0	0	45	0
	Location 3	7	20	77	0	0	6
Cluster 2	Location 4	45	0	0	57	0	0
	Location 5	0	0	0	12	23	0
	↓						
	L1 + L2	14	0	15	0	77	66
	L4 + L5	45	0	0	69	23	0
	↓						
	Cluster 1	9.7	0	16.5	0	53.4	72.5
	Cluster 2	31.2	0	0	75.8	15.9	0

Figure 2.7: This figure shows the computation steps that are applied to the BOW matrix after the clustering is done. At first locations that are contained by the same cluster are added together. Then the tf-idf value is computed for every term in every cluster.

topics which describe a semantic level between terms, this limited representation can be softened and unobservable relations between terms can be captured. $p = (t|z)$ describes the proportion of term t in topic z and $p = (z|d)$ describes the proportion of topic z in document d. In other words, a document is described by a number of topics in varying amounts, while each term contributes to one or several topics in a certain amount. The structure of LDA is shown in figure 2.8. θ is a distribution function that distributes topics per document, therefore is contained by the outer plate. For every document exists N terms $t_1, t_2, ..., t_N$ as well as N unobservable variables $z_1, z_2, ..., z_N$ that store the topic of the corresponding term. The hyper-parameter α and β only exists once. α controls the per-document topic distribution, while β controls the per-topic word distribution.

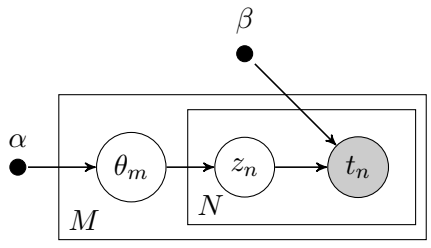


Figure 2.8: The structure of LDA. The plates (boxes) represent replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document (according to [9])

The used algorithm is an online variant of LDA proposed by Matthew D. Hoffman, David M. Blei and Francis R. Bach [16]. In their paper they also investigated the performance capability of this algorithm. They conclude that the algorithm is able to handle massive document collections. Also, they managed to create results with over 3.3 million documents taken from Wikipedia[3] in under 3 hours.

LDA was used to generate 10 topics for the collection of titles and abstracts of all 0.72 million publications. For the computation of LDA, publications were not grouped together to unique locations, instead each publication was trained as a separate document. A shortened version of one of the generated topics looks as follows:

Topic 1: 0.023*learning, 0.022*mining, 0.015*classification, 0.013*clustering, 0.010*feature, 0.009*detection, 0.009*ontology, 0.009*accuracy, 0.009*selection, 0.008*training, 0.007*prediction, 0.007*similarity, 0.005*machine, 0.005*pattern, 0.005*statistical, 0.005*neural, 0.005*measure, 0.005*quality, 0.005*evaluation, 0.004*extraction

The number before each word describes how much it does contribute to the topic. Topic 1 is evidently related to *artificial intelligence* as it contains a lot of the technical terms, used in this research area. A second topic looks like this:

Topic 2: 0.038*image, 0.025*video, 0.018*spatial, 0.014*visual, 0.011*motion, 0.010*recognition, 0.009*face, 0.009*segmentation, 0.009*object, 0.009*human, 0.007*shape, 0.006*space, 0.006*robot, 0.006*location, 0.006*map, 0.005*resolution, 0.005*camera, 0.005*color, 0.005*region, 0.005*temporal

Topic 2 contains mostly words that are related to *computer vision* and especially *image processing*. Other topics out of the 10 can roughly be titled with: products/commerce, hardware/security, math, service/management, web, signal/frequency, complexity/datastructures and energy. The number of 10 topics was derived through testing as more topics did not seem to give more benefit and showed subjectively worse results on the atlas, while less topics started to miss certain important words that were present before. Topic 1 and Topic 2 show that LDA was able to derive topics that can be identified to represent specific topics of computer science, which can roughly be considered as a research area. The most contributing 150 words from each topic were used to build a new text corpus which holds only 1500 unique words instead of 23 thousand. This new collection now holds an equal number of terms per topic that describe different computer science research. When creating a BOW with grouping publications by locations, while also removing all words that are not in the new derived text collection, the resulting matrix holds the 13 thousand unique locations as rows and the 1500 terms as columns. However, tf-idf is not used to rank the words in each cluster, since generated word-clouds are still very similar. Instead the ranking is done by comparing the tf of a term inside a cluster with the average tf between all clusters. To achieve this, the tf of a term inside a cluster is subtracted by the average tf

of that term. The words with the highest values are then selected. Consequentially shown terms for a cluster are the ones, which differ the most from the average tf. A example is shown in figure 2.9.

BOW

	t_1	t_2	t_3	t_4
Cluster 1	3	8	2	9
Cluster 2	8	8	1	3
Cluster 3	1	8	9	0

$\longrightarrow$

BOW - average tf

	t_1	t_2	t_3	t_4
Cluster 1	-1	0	-2	5
Cluster 2	4	0	-3	-1
Cluster 3	-3	0	5	-4

Figure 2.9: Figure of the ranking method that is used for the BOW on large areas. Each tf gets substracted by its average tf. The terms with the highest values for each cluster are chosen to be displayed in the corresponding word-clouds.

2.3 Step 3: Visualization

The goal for the visualization is to be as simple and intuitive as possible to encourage users to freely explore computer science research in different places all over the world. This section explains how the visualization is realized and which technologies are used in order to accomplish the goals defined in chapter 1. The atlas shows word-clouds on a world map, so the first thing that is explained is the used map. Next, it is shown how the word-clouds itself are computed. To give users a feedback on what the application is doing and to highlight which geographical area a word-cloud represents, a few extra visual effects were added to the visualization.

2.3.1 The Map

The map that is used for the research atlas is OpenStreetMap (OSM). It suits the requirements well, since it is, unlike for example Google Maps or Bing Maps, open source. All data can be used freely as long the applications mentions that its map data is provided by OSM. The database is user-based which means that people can help extending the map by contributing geo-graphical informations, for instance the route of a forest bicycle trail that was not tracked yet. The data is stored in a topological data structure, so each polyline/polygon such as a street, a river, or a country border is decoded as a set of nodes, stored as coordinates and ways that connect the nodes. Relations represent the relationship of existing nodes and ways, like a long motorway that spans over several ways. Key–value pairs (called tags) can be attached to nodes, ways and relations to store any additional informations, like for example $maxspeed = 50$. A tile server is needed to actually make the data visible. A tile server reads the necessary data from OSM and creates

small images (tiles) that, on the client side, plug together to one continuous image and show a certain area at a specific zoom level. When the user starts to move the map to a new position or changes the zoom level, the server renders the new area in form of tiles and sends them to the client. The new tiles then get plugged to their desired positions or build a new image. A example of such tiles is shown in figure 2.10. How the server renders

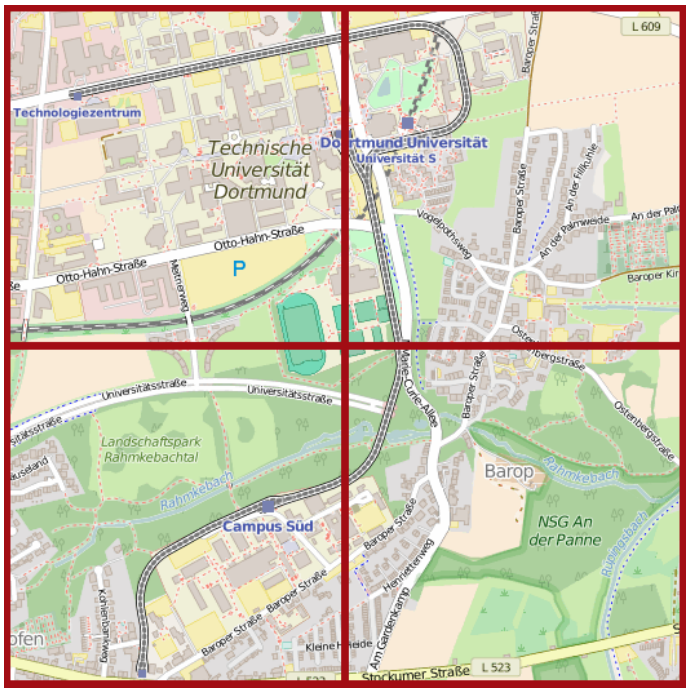


Figure 2.10: Four images (each 256 × 256 pixel) generated by the standard OSM tile server. On the client side they plug together and create one continuous image. If the user moves the map, the application requests new tiles from the server and plugs the revived images onto the corresponding position.

the images is a question of configuration. Usually images hold more general information when zoomed far out and become more and more detailed the further the map is zoomed in. Colors, shapes, degree of detail, fonts, etc. can be specified by the developer. Since the focus of the research atlas lies on the generated word-clouds, the map should have a fairly minimal design with no bright colors, terrain data, topology or other unnecessary information. However building a nice looking style from scratch is a difficult task, since it requires advanced skills and talent in map design. Luckily a lot of free designs can be found on the web. Informations about the specifically chosen design can be found in the appendix. It is mostly white and just holds very basic informations like the names of countries and cities. Therefore the focus is on the word-clouds, while still giving the users enough information for orientation. The design is used in the figures 2.2, 2.3, 2.4 and 2.5.

2.3.2 Cloud-generation

Depending on the color, typography and composition, word-clouds can have a very distinct appearance, as already discussed in section 1.2. Therefore different word-clouds are computed with various algorithms. The most basic layout is a word-cloud with a circular or rectangular shape that places words randomly inside the area. In this case the word size indicates the importance of a word but its position has no particular meaning. The first step to generate such a word-clouds is to decide which words should be in it, as shown in section 2.2. The second step is to decide which size, font, rotation and color each word should have and how big the resulting image should be. The color of words inside the research atlas is chosen randomly but with regard to the fact that same terms in different word-clouds have the same color. Terms are also only placed horizontal, which makes every word equally easy to read. The size of a word is determined by its term frequency inside the corresponding cluster. However, the tf between words inside a cluster can vary a lot, which in some cases leads to word-clouds that have a few extremely big words, while others are too small to read them properly. For this reason a base 2 logarithm is applied to the tf of every term. This gives a good balance between having words with different sizes, while still ensuring that each words is readable. The actual placements of the words takes place in the third step. Given the set of words $W = \{w_1, w_2, ..., w_n\}$, an algorithm needs to assign a position $p_w = (x_w, y_w)$ to each word $w \in W$ in such a way that all words are close together but do not intersect. Such an algorithm, explained in [11] and based on the Wordle method [14] is shown in Algorithm 2.2. It is a greedy algorithm that places words

Algorithm 2.2 Spiral Word-Cloud generation

Input: Words W, Iterations Threshold max

Output: Final positions $\boldsymbol{p} = \{p_w : w \in W\}$

```
1: for all words w ∈ {w_1, w_2, ..., w_n} do
2:     count ← 0
3:     while w is intersecting ∧ count < max do
4:         move p_w one step along the spiral path
5:         count ← count + 1
6:     if w still intersecting then
7:         Restart with a larger frame
```

along a spiral shape on a frame. Words get placed in order of their size, by starting with the largest one. Additional words are placed one by one on random positions. In the case an initial placed word intersects with any other word, it gets moved in small steps, along the spiral until it does not intersect anymore. A example is shown in figure 2.11. As mentioned in section 1.2, it is difficult for humans to compare different word-clouds with each other. To address that problem the also mentioned word-storms can be used. The idea

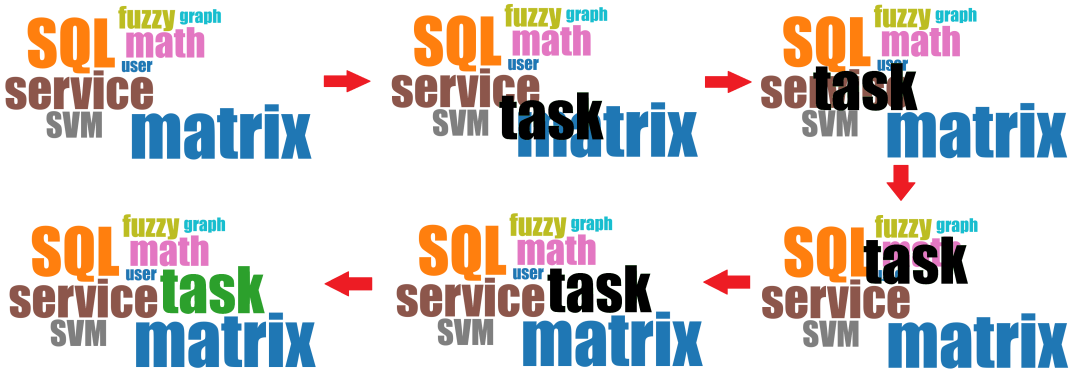


Figure 2.11: A example of how a word is placed into a word-cloud. The word "task" gets placed somewhere on the frame. As long as it intersects with other words it gets moved around in a spiral shape. When a free spot is found, the word is placed and the algorithm continues with the next word until all words are placed.

is to take the randomly generated word-clouds and compute the mean position for each word that appears in multiple word-clouds. Next, the spiral algorithm is modified, so that it tries to place words at specific locations rather than random ones. By recomputing the word-clouds with the specified mean position for each word, the word-clouds then have the same words at approximately the same position. However, the recomputing has to be done multiple times to achieve a satisfying result. The exact algorithm is shown in detail in the corresponding word-storm paper [14]. Besides the position, the same words get the same color, size and orientation. Frequent words get a lower alpha channel[3] value, which makes it possible to get an idea on how often a word appears between word-clouds. Words that are unique in a word-cloud stand out a bit as they are completely opaque, while words that appear in several word-clouds are a bit transparent. Finally the generated word-clouds are stored as an image that has a format of 4:3. On client side they can then be scaled to the right size, to make them fit on different screen sizes.

2.3.3 Design / User Interface

To give users more feedback on what the application is doing and to make it more appealing to the eye some visual effects are added to the scientific research atlas. To indicate which geographical area a word-cloud represents, the map highlights the corresponding area under word-clouds. The solution is to mark all unique positions that are contained by a cluster with points of the same color on the map. Points from different clusters get a different color to make them distinguishable. While this works well for a small amount of locations, the graphical performance of a normal web browser suffers when it has to render several thousand points, thus too many points can be confusing to users. If an area holds more

[3]The alpha channel controls the transparency of every pixel in a image. Usually the value is between 1 and 0. 1 is opaque, 0 is completely transparent

than 500 unique locations in total, the atlas switches from displaying single locations to showing just the border of the whole covered area. To do so, the *convex hull* of each cluster is calculated. The convex hull is the smallest convex set of points that contains every point of the cluster. Figure 2.3 shows the unique points as well as the convex hull of 5 clusters. The algorithm that is used to compute the convex hull for each cluster is *Quickhull*[7]. It is a recursive algorithm that uses a *divide and conquer* approach. Therefore its best case run time is $\mathcal{O}(n * \log(n))$ and its worst case run time $\mathcal{O}(n^2)$, with n being the number of points. The number of points is ∼13 thousand in the worst case. A full explanation of the Quickhull algorithm can be found in [7]. Having the convex hull of each cluster allows to compute the centroid of each hull. For a convex hull with n ordered vertices $(x_0, y_0), ..., (x_{n-1}, y_{n-1})$ this can be done with [8]:

$$
\begin{aligned}
C_x &= \frac{1}{6A}\sum_{i=0}^{n-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \\
C_y &= \frac{1}{6A}\sum_{i=0}^{n-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)
\end{aligned}
\tag{2.6}
$$

where A is defined as:

$$
A = \frac{1}{2}\sum_{i=1}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)
$$

(C_x, C_y) then is the centroid for the corresponding hull. When a cluster has a convex hull (this is the case when the cluster contains more than 2 points) the center of the corresponding word-cloud is placed at the center of the hull, rather than on the centroid that was estimated by the k-means clustering algorithm. This makes it easier to distinguish which word-cloud belongs to which cluster and leaves more space for the word-clouds. Consequentially word-clouds can be displayed a bit bigger without intersecting on the screen. Figure 2.12 shows the difference between the centroids estimated by k-means and the centroids of the convex hulls. To make sure that displayed word-clouds do not intersect with each other, the pixel distance between all centroids is computed. Afterwards all word-clouds, which are stored as images, get scaled down to a the size of the smallest distance. Another visual effect that was added regarding the computation time. The applications does not have any buttons to request new word-clouds. Yet, informations refresh as soon as the user moves the map to a new location. To indicate that the server is computing and the user should wait to see desired results, a spinner loading symbol appears in the middle of the screen. However, while the spinner appears immediately, the actual request is started after a short delay to reduce the load on the server. This is done in case that the user searches for a specific place and therefore moves the map around quickly in small steps. Also, the application does not request new data, when the user drags the map a very short distance, by reason that small adjustments of the viewport should not directly result in a new request and displayed word-clouds are still visible anyway.

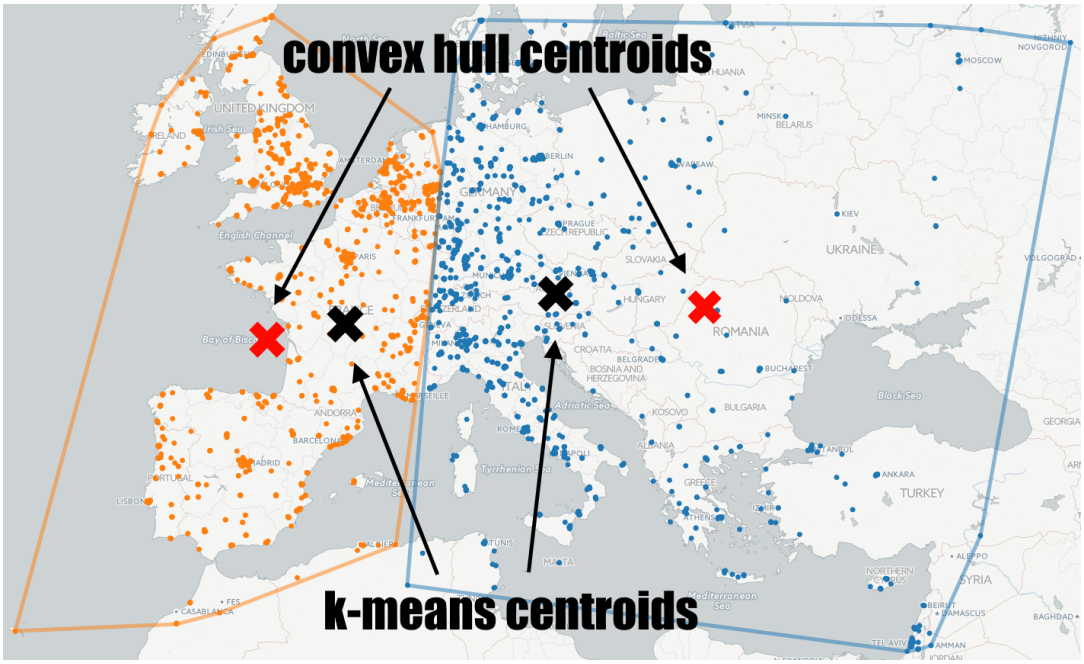


Figure 2.12: This figure shows the difference between the centroids that were estimated by k-means and the centroids of the convex hulls. While the k-means centroids are the mean value between all points in the cluster and therefore lean towards the center of gravity, the convex hull centroids are visually in the middle of the cluster. Word-clouds that are displayed on the convex hull centers are easier to distinguish and have more space.

Chapter 3

Reading of the Atlas

This chapter illustrates results that the scientific research atlas generates and tries to examine their quality. How informative the shown results are is difficult to evaluate since opinions about the quality of generated word-clouds are very subjective and the absence of similar applications does not allow direct comparison with other works. The two big aspects of the atlas are the content wise performance of the word-clouds as well as the visual performance of the application. Section 3.1 illustrates the visuals of the atlas and discusses the general usability as well as the readability of the shown keywords. The next section 3.2, compares different results that the atlas has generated and shows reasons why keywords of some kind appear in certain areas. Furthermore it shows which assumptions can be made based on the generated results. At last, an evaluation that analyses word-clouds based on coherence measures is given in section 3.3.

3.1 Illustration

Figure 3.1 shows the whole world. The clustering algorithm estimated three clusters which

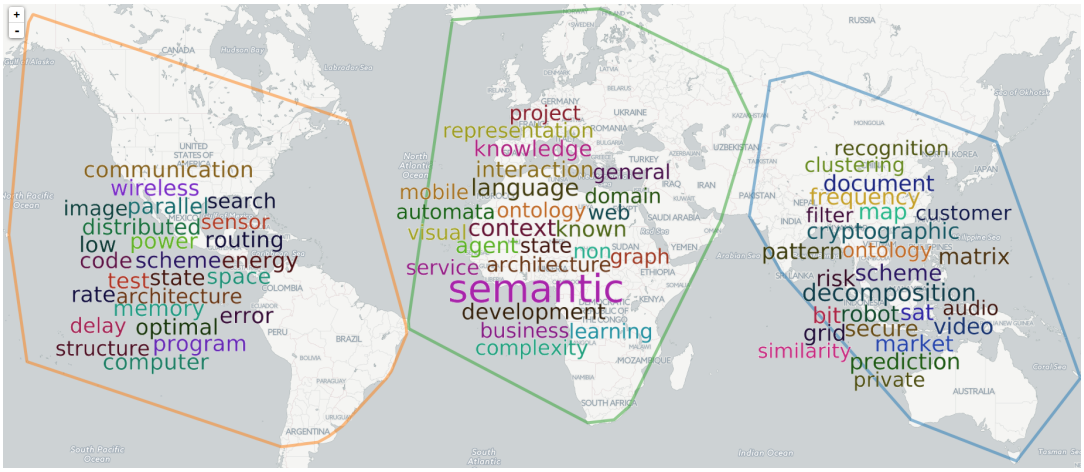


Figure 3.1: The scientific research atlas showing the whole world.

are in line with the distribution of unique locations shown in figure 2.2. Each cluster contains one of the three areas that are populated with most of the available data, Europe, Asia and America. The space between clusters is large enough to display all three word-clouds at the same time, thus all words are big enough to read them comfortably. However, the atlas is optimized to be used on a computer or laptop screen, so on screens with very limited size like smart phones, words will eventually become unreadable. Borders and country names are displayed to allow orientation, the map and fonts are white and gray, therefore the contrast of word-clouds is high enough to be distinguishable. The convex hull of each cluster underlays the corresponding word-cloud to interrelate the contained keywords with the geographical area. The color variation between words is chosen randomly. An exception are the words that appear in multiple word-clouds like "state" and "architecture", which can be found in the left and mid word-clouds. These words share the same color and are located at approximately same positions. However, the word-clouds are mostly very different which is easily distinguishable as the word-clouds have significantly varying shapes and colors. Comparing this figure to figure 3.2 where a part of England is shown, highlights the advantage of word-storms further. Here the shown word-clouds

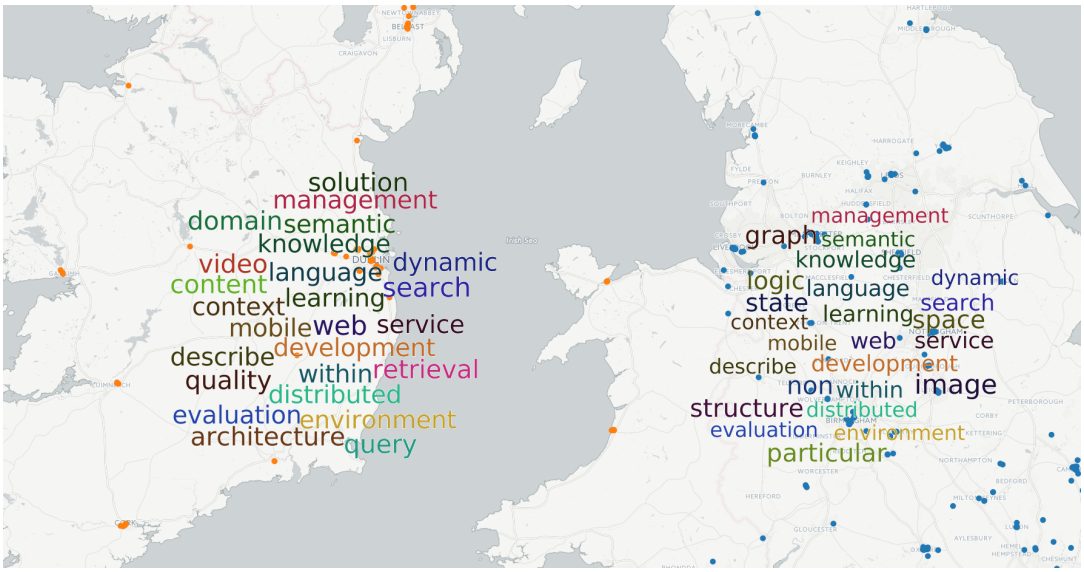


Figure 3.2: The scientific research atlas showing a part of England.

are more similar and same words can be found easily, which facilitates the comparison of the various areas. Instead of a convex hull, colored points underlay the word-clouds and therefore identify the specific unique locations that contribute keywords to the word-cloud. Since word-clouds are displayed in between the points of a certain color its obvious which word-cloud corresponds to which color, even without any extra indication. When looking at an even smaller area as the case of Hawaii, which is shown in figure 3.3, the fluctuation between word sizes becomes more significant, as on smaller areas with less data, the tf between words usually varies more. This lets some words stand out and makes the word-

cloud appear more compact. Looking at Hawaii again on a lower zoom level (figure 3.4),

Figure 3.3: The scientific research atlas showing the island of Hawaii.

the clustering algorithm decides to just create one cluster instead of two, as all points of Hawaii are now grouped together in a small spot of the viewed area. The atlas allows to

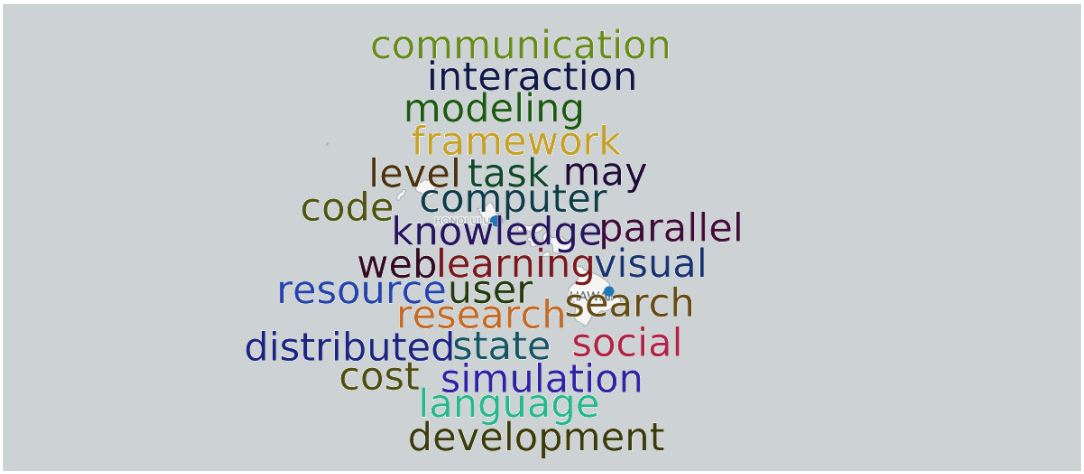


Figure 3.4: The scientific research atlas showing the island of Hawaii.

zoom in very far to see street names and differentiate single buildings. This can be helpful if one is interested in a single point and wants to examine where it is located exactly, to do further investigations.

When a desired position is found by a user it takes several seconds before results arrive. This is a noticeable delay but acceptable considering that users will most likely stay at the same position for several seconds or even minutes, since reading and comparing word-clouds is time-consuming. The experience of the research atlas is not geared towards user task completion or task speed. Table 3.1 shows a list that plots the computation time of

Task	Time
Finding all publications inside an arbitrary area	$\sim$ 5ms
Clustering	$\sim$ 500 ms
Computing tf-idf matrix	$\sim$ 200 ms
Computing convex hull + centroids	$\sim$ 50 ms
Generating the wordstrom	$\sim$ 2000 ms

Table 3.1: Table of computations times. Note that those numbers vary depending on the given area. Used hardware: Intel Core i7-860 Processor (8M Cache, 2.80 GHz) and 8GB ram.

each processing step. Those numbers are approximated and vary depending on the given area and computation power of the executing hardware.

3.2 Interpreting generated Results

When looking on the whole world shown in figure 3.1, it can be observed that the keywords themselves are very diverse. While they do not directly identify research areas like for example "artificial intelligence" they do show important but normal technical terms. Terms like "sensor", "search", "space", "computer", "scheme", "grid" are frequently used in most texts related to computer science, yet on their own do not deliver any profound information about specific research. On the other hand, terms like "risk", "neuronal", "routing", "robot", "sensor" or "mobile" are more specific and deliver an idea on different research areas that the various regions focus on. However the words are distributed too randomly to offer any meaningful observations or to identify any specific trends or topics. Such large areas that contain many different countries, have too diverse research to be able to explain them with 30 words per word-cloud. For example, the word "web" shows up in the central word-cloud located over Europe and Africa. This indicates that the word "web" does show up more often in this region than in the other two. But still, this does not necessarily mean that this region focuses more on web, as "web" is just one word out of many that is related to web. Therefore the cloud would have to show many words directly related to a certain topic to allow such assumptions. The small amount of shown words consequentially limits the ability of the atlas to summarize research on large areas.

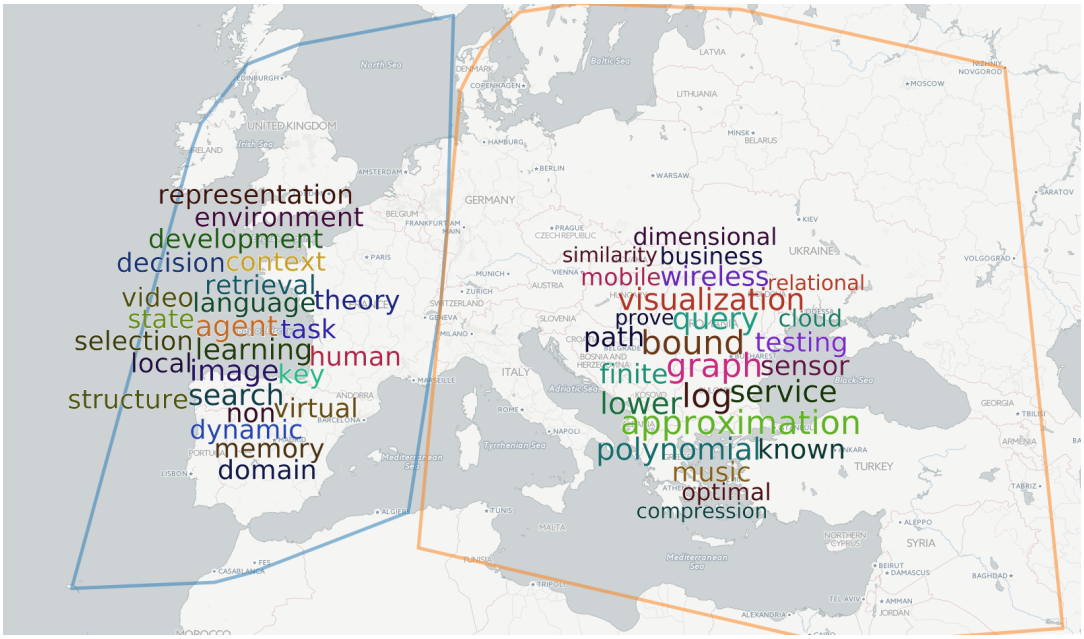


Figure 3.5: The scientific research atlas showing a part of Europe.

The same problem can be observed on Europe, shown in figure 3.5. Displayed words are still spread out over many different topics and do not allow profound assumptions. However, the subjective quality of word-clouds increases when considering smaller areas.

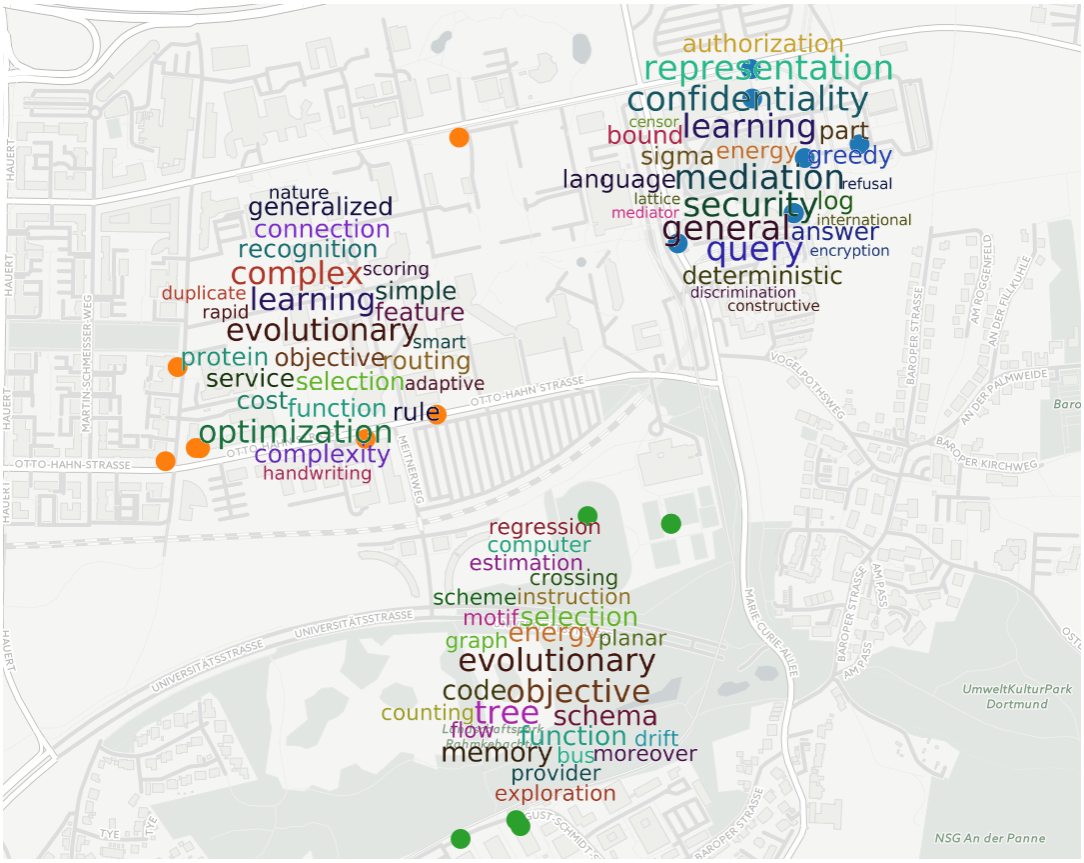


Figure 3.6: The scientific research atlas showing the TU Dortmund university in Germany.

Figure 3.6 shows the area of the TU Dortmund university in Germany. While some keywords are still not especially meaningful like "simple", "general" or "complex", more substantial keywords are found as well. A lot of terms are connected to artificial intelligence, like "learning", "feature", "optimization", "evolutionary" and "selection". Other topics revolve around hardware, algorithms and optimization. Two remarkable keywords, "protein" and "nature", do even indicate research related to bioinformatics. Compared to the word-clouds generated for the whole world, the TU Dortmund word-clouds do identify research areas much clearer.

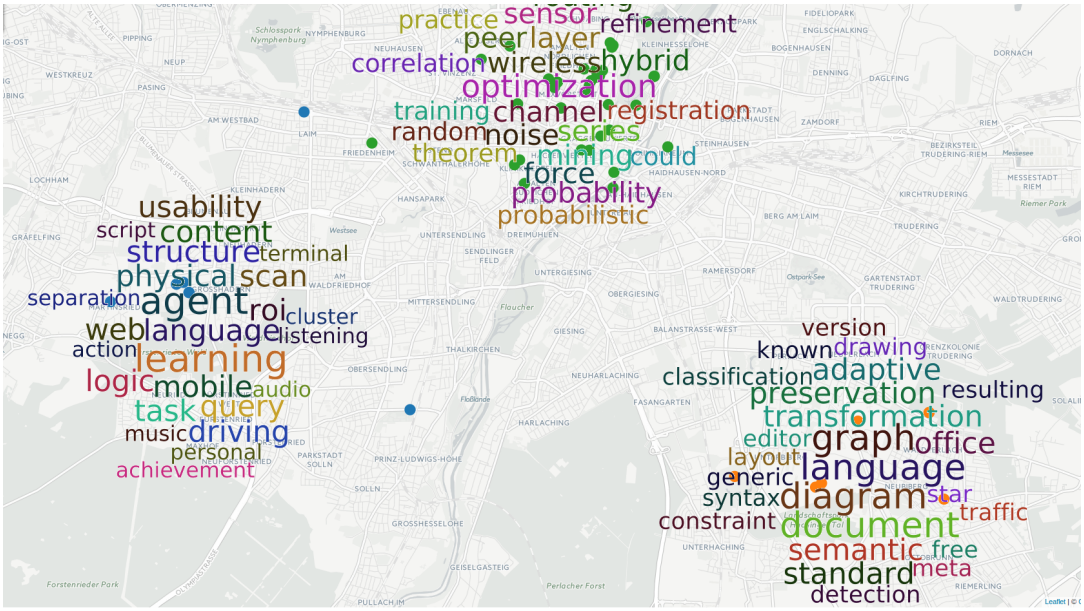


Figure 3.7: The scientific research atlas showing the city of Munich in Germany.

Figure 3.7 shows the city of Munich in Germany. Dortmund and Munich show some similarities like words related to artificial intelligence. However differences between them can also be identified. Interestingly Munich shows terms that in the broadest sense are related to sensors like "noise", "sensor", "scan" and especially to cars like "driving", "traffic", "adaptive", "mobile", "music", "sensor", "hybrid". Those results make sense, as Munich houses the headquarter of the famous car manufacturer *BMW*.

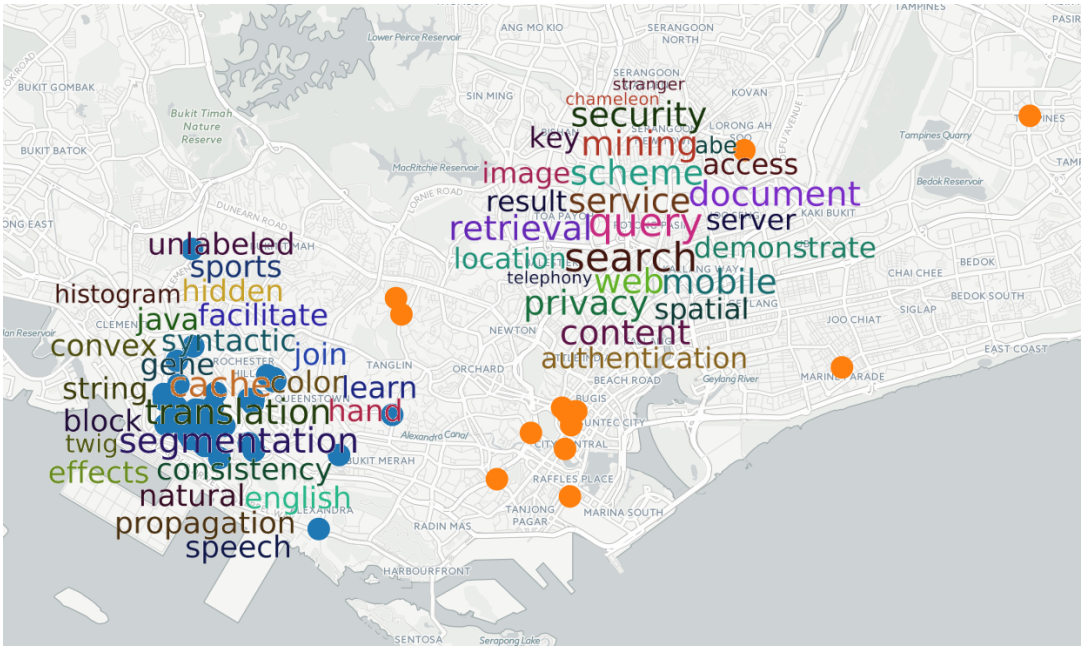


Figure 3.8: The scientific research atlas showing the city of Singapore Asia.

On figure 3.8 the city of Singapore is shown, which is located in Asia. Here a lot of research is focused on web as terms like "server", "web", "java", "mobile", "search" and "authentication" indicate. Furthermore words like "telephony","translation", "speech", and "english" connect research to phones and communication.

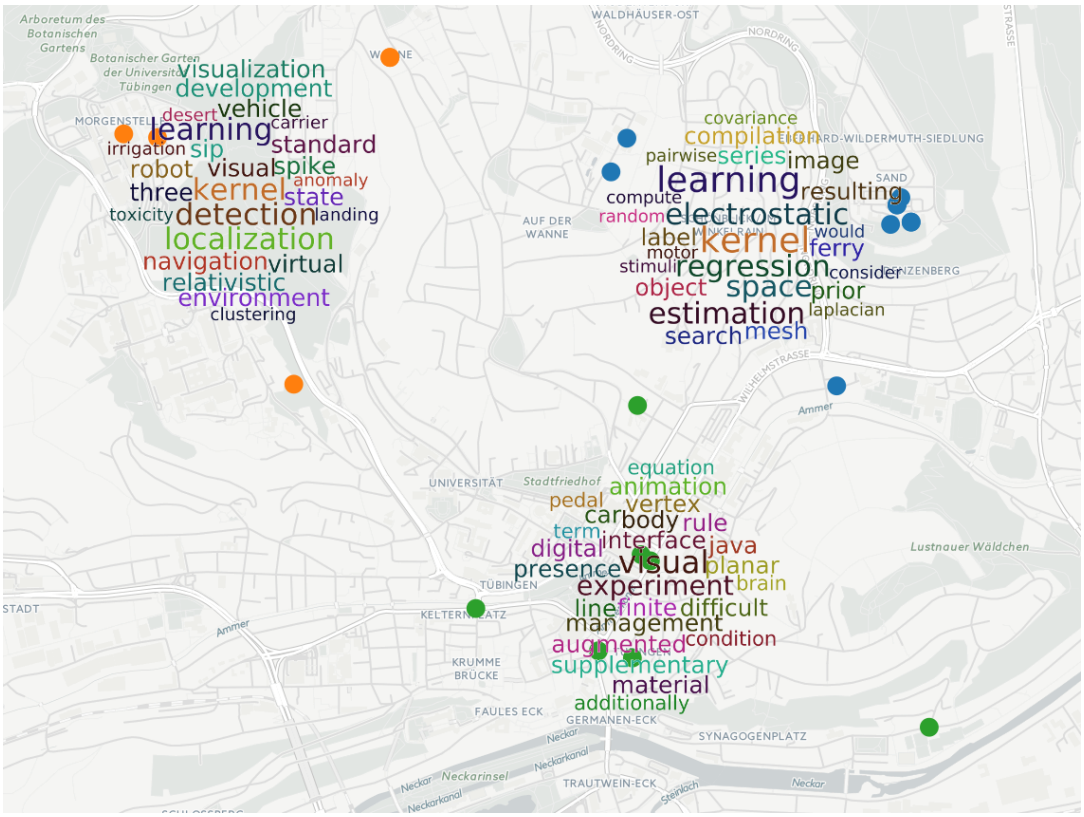


Figure 3.9: The scientific research atlas showing the city of Tübingen in Germany.

The next viewed area is the city of Tübingen in Germany (figure 3.9). The writer of the book *Learning with Kernels* Bernhard Schölkopf who is the director of the *Max Planck Institute for Intelligent Systems* in Tübingen, has several publications related to *kernel methods*. Therefore it is not surprising that the word "kernel" does show up twice in this area. Investigating further, it turns out that the chair of embedded systems of the university in Tübingen does research about *Autonomous driving*. This does explain terms like "vehicle", "car", "motor", "pedal" and maybe also "carrier". On top of that, the *Max Planck Institute for Biological Cybernetics* has a project called *Virtual Tübingen*. The goal of this project is to build a realistic virtual model of the city of Tübingen and is therefore directly related to *Virtual Reality*. Accordingly the terms "virtual", "visual", "animation", "augmented", "navigation" and "localization" do indicate such research.

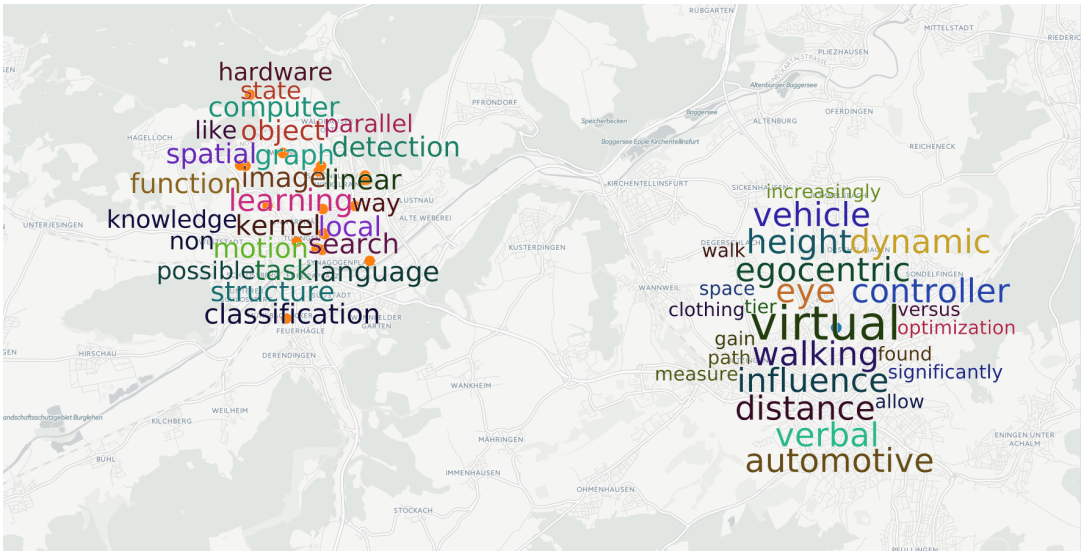


Figure 3.10: The scientific research atlas showing the cities Tübingen and Reutlingen in Germany.

One zoom level higher, but still on the same area, Tübingen is now represented by a single word-cloud on the left of figure 3.10. Because of that, a lot of the before encountered words are not present any more. The term "Kernel" is nonetheless still present. The word-cloud on the right stands out as it only represents a single point on the map. The institute represented by that point is the university of Reutlingen, found by zooming in very closely to the point. The university of Reutlingen does have a chair for computer science. A professor employed at that university published amongst others three papers *Egocentric distance perception in large screen immersive displays* , *The influence of eye height and avatars on egocentric distance estimates in immersive virtual environments* and *Egocentric distance judgments in a large screen display immersive virtual environment*[1]. Those papers clearly provide some words that are shown in the corresponding word-cloud like "egocentric", "eye" and "virtual".

[1] Namely: Prof. Dr. Uwe Kloos

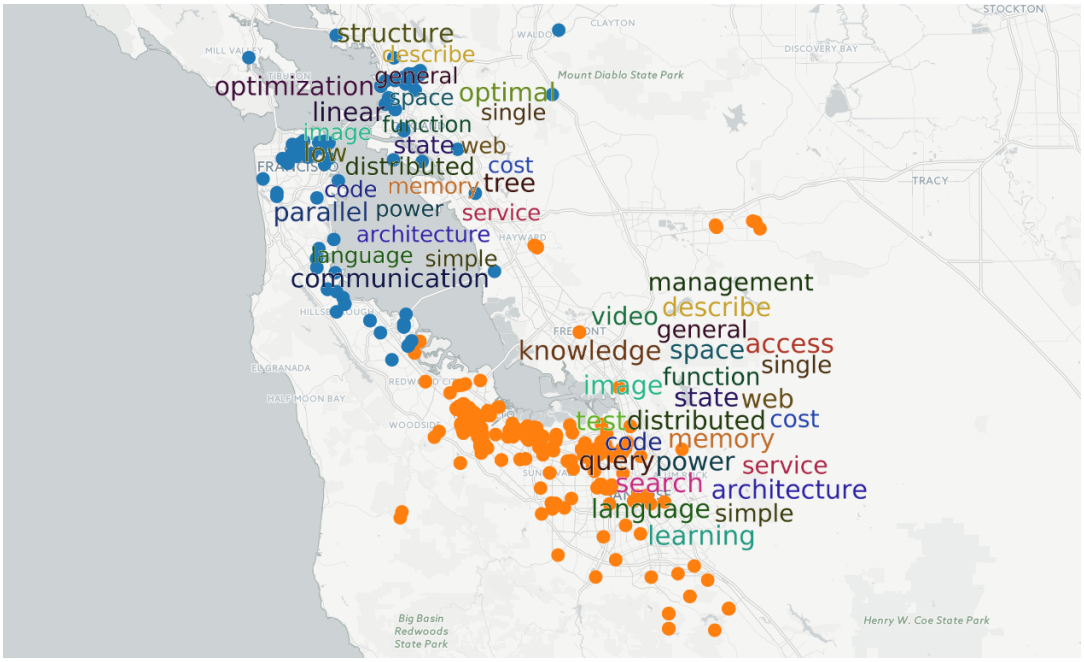


Figure 3.11: The scientific research atlas showing the Silicon Valley in San Francisco USA.

Looking at Silicon Valley in San Fransisco USA, the keywords have a more general character. The high amount of available data can also be seen by the high density of points in that area, considering that Silicon Valley houses many of the world's largest high-tech companies and start-ups. Noticeable is that all shown words have almost the same size. That can be explained by reason that the term frequencies of words are more evenly distributed on areas that hold diverse research about many different topics. As a consequence of the diverse research, shown word-clouds indicate a variety of different computer science topics. Terms like "learning" and "optimization" seem to appear in almost every shown area so far, regardless of location or size. Terms as "web", "video", "search", "image" and "communication" do not surprise as several of the larges IT companies like Google, Facebook, Apple, Yahoo!, Netflix and many others focus on such things.

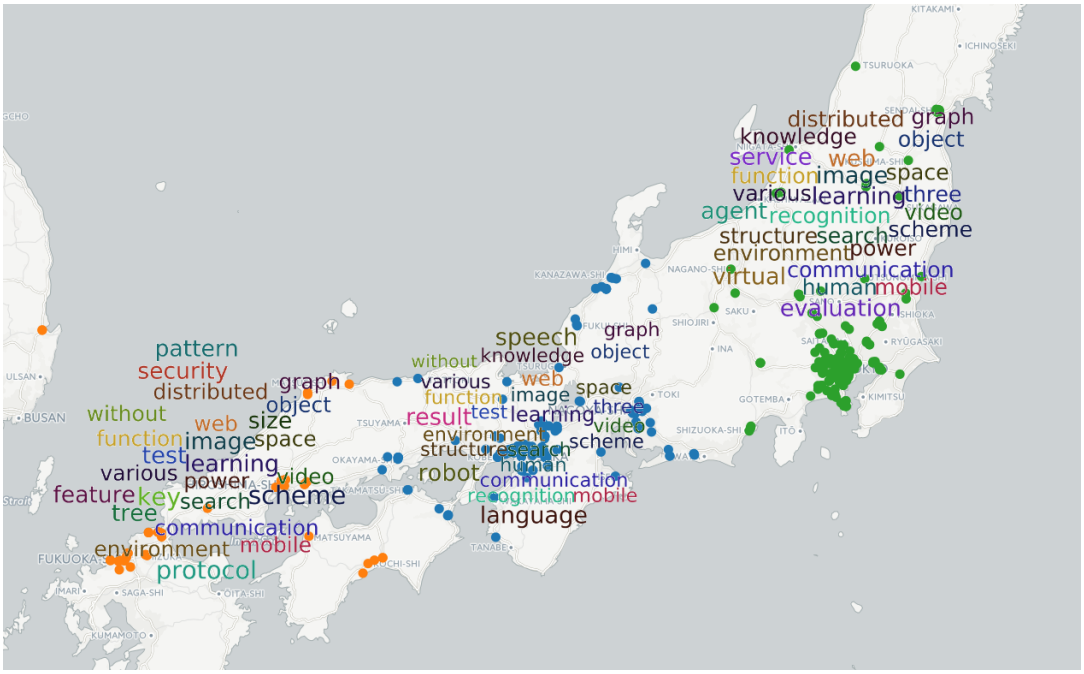


Figure 3.12: The scientific research atlas showing Japan.

Figure 3.12 shows Japan. It is relatively densely populated with many affiliations, in comparison to many other countries in Asia. Here, all three displayed word-clouds share several words like "communication", "mobile", "image", "video" and "web". This first indicates the importance of such terms in Japan and also shows that those seem to be important in all regions of Japan. Consequentially Japan seems to do a lot of research regarding communication and web. One other noteworthy term is "robot". As this term is located in the middle word-cloud it seems that robots are especially important in this area. And as it turns out, some of the largest companies in Japan that do research related to electronics and robotics, Sony, Honda, Toyota, Toshiba and others are located in that area (most of them in Tokyo).

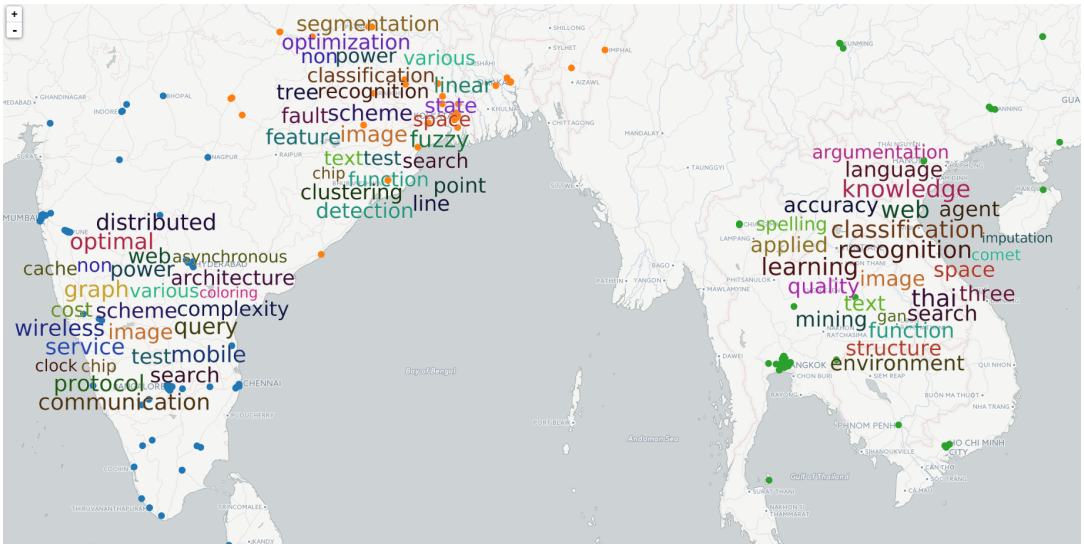


Figure 3.13: The scientific research atlas showing parts of India as well as Bangladesh and Thailand.

Looking at India, Bangladesh and Thailand (figure 3.13) the density of displayed points is relatively low, considering that the figure shows an area that is several million km^2 big and populated by over a billion people. The right word-cloud over Thailand contains the word "thai", making it the first location so far where the country name is represented as a keyword. The three shown word-clouds do not share a lot of terms and topics are spread out over many different research topics.

In conclusion, the atlas shows results that especially on smaller areas deliver useful informations about related research and reflects, at least in the shown cases, the tendencies of work done by people and affiliations at the corresponding areas. With increasing amount of contained affiliations, the word-clouds for areas become progressively general and present a more universal topics of computer science.

3.3 Coherence of generated Keywords

To evaluate generated word-clouds further, the quality of contained keywords is compared based on their coherence value measurements. The coherence describes how closely related words are in a topic. Coherence measures are used for instance to examine the quality of topics generated by a topic models like LDA, as those give no guarantees on the interpretability of their output. Several coherence measures have been proposed by Michael Röder, Andreas Both and Alexander Hinneburg in their paper *Exploring the Space of Topic Coherence Measures*[28], that aim to distinguish between good and bad topics in regard to interpretability, such that they correlate to human ratings. They created a tool called *Palmetto*[4], hosted by the university of Leipzig, which offers different coherence calculations for topics. These coherence measures are based on word co-occurrences of the English

version of wikipedia[3]. They correlate, as shown in their paper, with human ratings and therefore can be used to evaluate word-clouds generated by the scientific research atlas. The two used coherence measurements are C_{UMass} and C_{NPMI}.

Let $D(t)$ be a function that gives the number of documents that contain term t (in case of Palmetto those documents are from wikipedia) and let $D(t_i, t_j)$ be a function that gives the number of documents that contain t_i and t_j. The C_{UMass} value for two given terms t_i and t_j is defined as

$$C_{UMass}(t_i, t_j) = \log \frac{1 + D(t_i, t_j)}{D(t_i)} \tag{3.1}$$

This measure was proposed in the paper *Optimizing semantic coherence in topic models*[26]. For a sequence of up to 10 words Palmetto computes the C_{UMass} value by computing the arithmetic mean of all possible pairs.

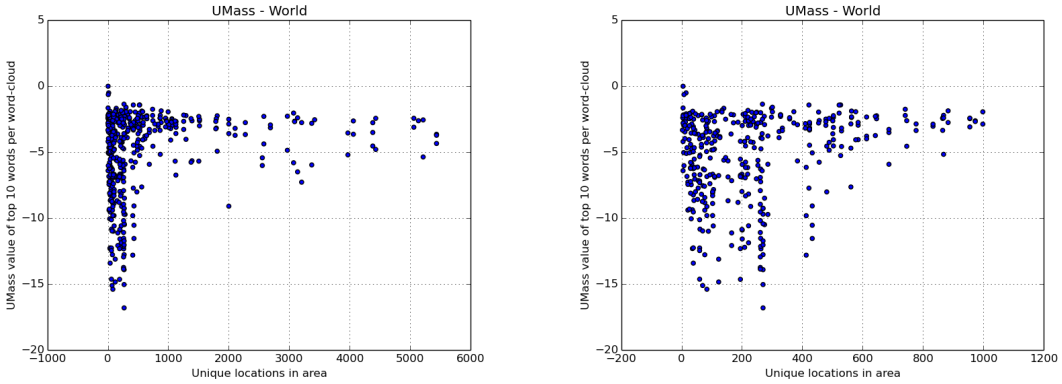
To compute the C_{NPMI}, first the *point wise mutual information* has to be defined. It is a function that gives the probability of seeing a term or respectively two terms in a random document $\text{pmi}(t) = \log \frac{D(t)}{|D|}$ and $\text{pmi}(t_i, t_j) = \log \frac{D(t_i,t_j)}{|D|}$, where $|D|$ is the number of total documents. However C_{NPMI} uses the normalized version of this function $\text{npmi}(t) = \left(\log \frac{D(t)}{|D|}\right) / -\log D(t)$ and $\text{npmi}(t_i, t_j) = \left(\log \frac{D(t_i,t_j)}{|D|}\right) / -\log D(t_i, t_j)$. The C_{NPMI} value for two given terms t_i and t_j is defined as

$$C_{NPMI}(t_i, t_j) = \log \frac{\text{npmi}(t_i, t_j)}{\text{npmi}(t_i)\text{npmi}(t_j)} \tag{3.2}$$

This measure was proposed by N. Aletras and M. Stevenson in their paper *Evaluating topic coherence using distributional semantics*[27]. Just like with C_{UMass}, the C_{NPMI} is computed for up to 10 words by computing the arithmetic mean of all pairs.

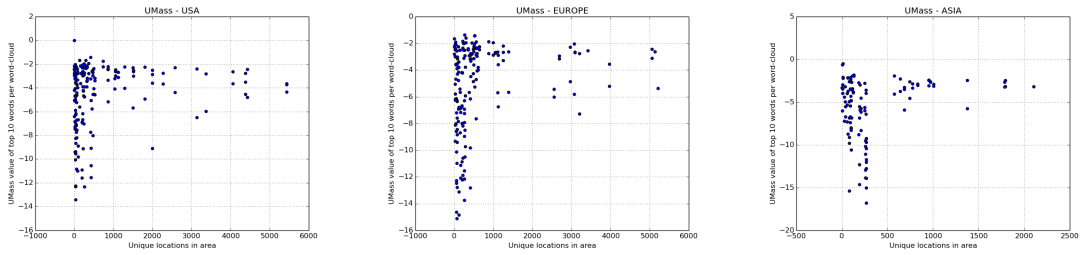
A example of a sequence of words that have a high coherence value could be [Apple, Banana, Cherry, Kiwi, Lemon, Aubergine, Broccoli, Clementine, Garlic, Melon], which has a C_{UMass} value of -0.519 and a C_{NPMI} value of 0.0346. A example of a sequence with a low coherence is [dispute, rhetorical, tutor, trigram, toll, herbal, provide, ellipsoidal, photoluminescence, dental], which has a C_{UMass} value of -11.968 and a C_{NPMI} value of -0.221. However the measures are not comparable with each other, but in both measures a higher value indicates a higher coherence.

The evaluated word-clouds were collected by scanning over different areas of the world in varying zoom levels. More specifically, clouds were taken from the USA, Europe and Asia. The map was moved around on those regions, such that the whole area was completely captures once on each zoom level. For each generated word-cloud, the top ten words were stored as well as the number of locations contained in the corresponding area. After that the C_{UMass} and C_{NPMI} were computed for the top 10 words of each word-cloud. Figure 3.14a shows C_{UMass} values for all generated word-clouds in context to the amount of locations that were inside the corresponding area. The point where the applications swaps to the word selection with tf-idf is noticeable. The word-clouds for areas that hold between 1000



(a) **(b)**

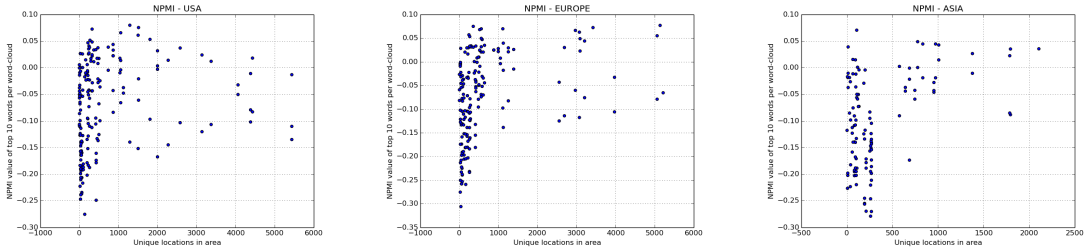
Figure 3.14: The C_{UMass} values for each collected word-cloud (y-axis) in context to the number of locations inside of each corresponding area (x-axis). Each point is representing a singe word-cloud. Image **(a)** shows all word-clouds. Image **(b)** shows all word-clouds inside areas smaller than 1000 unique locations.



(a) **(b)** **(c)**

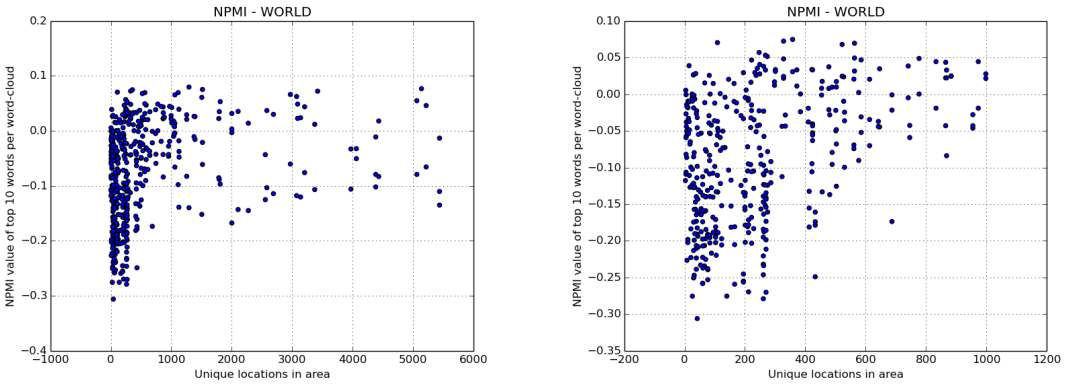
Figure 3.15: The C_{UMass} values for word-clouds divided by the regions USA, Europe and Asia. Image **(a)** shows all word-clouds from the USA. Image **(b)** shows all word-clouds from Europe. Image **(c)** shows all word-clouds from Asia

to 6000 locations, fluctuate around the mean value of -3.6. It seem to make no difference how much locations the areas hold as long as they hold more than 1000. Things change in the interval $[0, 1000]$, as at 1000 is a significant bump. To investigate this area further figure 3.14b shows interval from 0 to 1000. The mean value of the whole interval is -5.3. However the mean value of the interval $[500, 1000]$ is -3.1, which is even slightly better then in the interval $[1000, 6000]$. The interval $[0, 500]$ however has a mean value of -5.7, which is significantly worse than in [1000,6000]. It seems that in areas smaller than 500 the C_{UMass} coherence between words starts to fluctuate between -2 and -15 very randomly. Figure 3.15 shows the same graphic as in figure 3.14a but divided into the 3 different areas USA, Europe and Asia. It shows that all three areas behave approximately the same. Consequentially, the C_{UMass} coherence of word-clouds does not differ between different areas of the world. Looking at the C_{NPMI} values for the three areas (figure 3.16) the assumption is similar.



(a) **(b)** **(c)**

Figure 3.16: The C_{NPMI} values for word-clouds divided by the regions USA, Europe and Asia. Image **(a)** shows all word-clouds from the USA. Image **(b)** shows all word-clouds from Europe. Image **(c)** shows all word-clouds from Asia



(a) **(b)**

Figure 3.17: The C_{NPMI} values for each collected word-cloud (y-axis) in context to the number of locations inside of each corresponding area (x-axis). Each point is representing a singe word-cloud. Image **(a)** shows all word-clouds. Image **(b)** shows all word-clouds inside areas smaller than 1000 unique locations.

All three areas behave approximatively the same again. However the C_{NPMI} values seem to differ more significantly in the $[1000, 6000]$ interval. Looking at the interval $[0, 1000]$ of all three areas together (figure 3.17b), the distribution of C_{NPMI} values is similar to the C_{UMass} distribution in figure 3.14b. Here to mean value of all C_{NPMI} values is -0.07. The mean value of the interval $[1000, 6000]$ is -0.025. The interval that shows the with tf-idf generated word-clouds $[0, 1000]$ has a mean value of -0.089 while to upper half $[500, 1000]$ has the mean value -0.014 and the lower half $[0, 500]$ has one of -0.1. So just like with the C_{UMass} values, the upper half of the tf-idf word-clouds $[500, 1000]$ has the highest C_{NPMI} coherence, while the lower half $[0, 500]$ has the lowest. Accordingly, both coherence measures C_{UMass} and C_{NPMI} show mostly the same results. Areas that hold more than 1000 locations and therefore are created using the LDA word selection method fluctuate around a fixed value. Consequentially the LDA word selection method shows no

coherence increase or decrease when used on areas with varying sizes. A reason for the lower fluctuation could be the filtering with LDA. Words inside the same LDA topic have a higher chance of being coherent. Since only the top words out of each topic are selected, the chance is higher that word-clouds, generated with those words, are coherent. In the interval $[0, 1000]$ both measures show that word-clouds, generated on areas that hold just slightly less than 1000 locations, have the highest coherence on average. Going down the interval, the average coherence does decrease. However even on the smallest areas, some word-clouds do still have a high coherence. Therefore the assumption is that on small areas, the probability of the atlas to generate less coherent word-clouds increases. Because of the limited amount of data, very specific words can appear. This behavior is for instance shown in figure 3.10. On this zoom level, even single publications can highly influence the results. Therefore it is more random, whether or not word-clouds are coherent on small areas. For example, one word-cloud that had a C_{UMass} value of -14.66258 contained the words: [dispute, rhetorical, tutor, trigram, toll, herbal, lod, ellipsoidal, photoluminescence, dental]. In this case, some very exotic words are contained in the cloud. Another word-cloud that had a C_{UMass} value of -2.70521 contained the words: [object, function, solution, memory, communication, security, management, error, learning, better]. Those words clearly are more coherent. Both clouds were generated on areas that contained less than 200 unique locations. However a low coherence value does not necessarily mean that a generated word-cloud is bad by any means. For instance the words "photoluminescence" and "dental" in the above example are interesting words that excite to investigate where those exotic words are coming from.

Chapter 4

Conclusion

The previous three chapters explain the motivation for creating the atlas, the underlying concepts that are used to implement it, as well as the results that it delivers. This last chapter concludes the creation of the research atlas by discussing the results acquired in this thesis and shows which further improvements could enhance it. At first a look at how well the goals, that were defined in chapter 1, are met by the implemented scientific research atlas:

Easy to use The only needed controls to use the atlas are dragging and zooming via a mouse or touch pad. These controls are commonly used in other map services like Google Maps. With help of the spinner, users get a real time feedback on what the system is doing. By coloring the corresponding geographical area that a word-cloud represents, users are always aware which information relate to which positions on the map. For small areas, the atlas does even highlight the exact geo-positions that are represented. This results in a easy to use application, that everyone who used Google maps or any similar map service before, will understand.

Representative Most generated words are related to computer science. But especially on continent and country level the atlas has not fully met expectations. Words that directly mark research areas like for instance "data mining" are not present at all. Shown words are often contained by a certain vocabulary that is characterizing a research area. This does to a degree identify different research areas indirectly for people who are familiar with such expressions. A general problem is that since research has to be identified indirectly through given keywords, more words need to be displayed in order to see a trend. But this amount of terms would not fit in a few word-clouds. However, for smaller ares like a city, word-clouds become more meaningful and mostly give a good summary of research that has been done in a small area like a university. Therefore, while the atlas only gives limited insights into research areas on a large scale, it is well suited to help users to gain impressions

on more specific geographical areas and answer question like "On which computer science research do universities in a certain city focus on?".

Comperable By using word-storms the generated word-clouds have same words at the same position and with the same color, which helps to compare clouds with each other. It just takes seconds for a user to see how similar multiple word-clouds are. This makes them very comparable. What the atlas does not deliver is comparability between small geographic areas that are far away from each other, like comparing a university in Germany with a university in the USA. Word-storms are recomputed each time the map is moved and therefor share no similarities between multiple queries.

Dynamicly processed Clustering and word selection are done in real time and processed dynamically for every given arbitrary area. On one hand this enables the application to work with different datasets and adjusts to various amounts of available locations and text, but on the other hand also increases the computation load on server side as well as computing time. Accordingly dynamic processing does deliver some advantages but also introduces downsides mostly regarding computation time.

Meaningfull geographical clustering The application identifies groupings of locations dynamically and derives a reasonable number of clusters for every given area. The number of identified clusters can range from 1 to 5 depending on the structure of the area. Estimated clusters represent geographically bounded areas in most cases and leave enough space between them to properly display word-clouds. However the clustering algorithm is not aware of country or state borders and therefore does not deliver clusters that represent specific border bounded areas.

While not all goals were fully accomplished by the scientific research atlas, it does not disappoint on its ability to give an interesting and esthetically pleasing look into computer science research. Especially on smaller areas it delivers some striking results that give many useful insights. Yet, with some little changes and further improvements regarding keyword selection, clustering and visual performance, it cloud bear even better results. Those improvements could enhance the implemented application and can also be considered when implementing a similar new application from scratch. At first, an additional function could be added that generates word-clouds for geographical bounded areas. Besides the geo-locations, GeoDBLP does also hold informations about the country that an affiliation is in. Information about the state or nearest biggest city could also be discovered quite easily. With those information the atlas could give users the option to generate word-clouds for specific areas like a continent, country, region, city etc. This would of course introduce new problems regarding the visualization and especially the scaling of word-clouds in different zoom levels. Another way to approach this is to implement a clustering algorithm that

clusters dynamically, but still tries to create border bounded cluster if possible. This could be achieved for instance by introducing dummy variables to the k-means clustering that represent countries or cities. A possible downside is that this would introduce extra computing time. Another approach could be to use a different clustering algorithm like a hierarchical clustering as purposed in [18]. This would make it possible to pre-compute clusters with more advanced constrains, while additionally speeding up the application and also increasing the quality of estimated clusters. Improvements could also be made, regarding the quality of the content wise performance of word-clouds. Right now, the LDA topics are just used to find general words for large areas. When generating a small amount of topics, the atlas could give users the option the pick one or more specific topics and then just displaying words contained by those topics. This would give the ability to find more specific informations. Second, words inside word-clouds could be color coordinated according to corresponding LDA topics. An advantage of this would be that the average color of a word-cloud could give users an idea on which topics are particularly important for that certain area. It would even address the needle in the haystack problem described in section 1.2 as it would make it easier to find specific words. Thinking further, LDA topics could even allow for different visualizations that, instead of word-clouds, show the topic distribution over different areas in comparison. This would be especially beneficial on larger areas where keywords are too diverse to identify any real trends of research topics. Additionally LDA topic that take into account geo-referencing would allow deeper analyzes and could therefore lead to very interesting results. At last, the quality of keywords could be improved by taking into account the citations delivered by the Aminer Citation Network described in section 2.1.1. It delivers informations about all citations for every publication. Those information could help improving the ranking of words since publications that get cited a lot are usually important and consequentially, include words that can be considered as more important. Yet, this would require another keyword selection algorithm. A possible choice could be *TextRank*, which was proposed 2004 by Rada Mihalcea and Paul Tarau [24]. TextRank adds every word to a graph and then computes the importance by running Google's *PageRank* [10] algorithm on it. A possible modification would be to change the weights of this graph according to citations derived from the citation network.

The creation of the atlas revealed basic concepts and approaches that can be considered when trying to summarize and visualize geo-referenced data in a textual way. The amount of such data will increase even more over the coming years and will raise the need for algorithms and applications that help to summarize such informations in an easy and visually appealing way. A world map in combination with word-clouds, while being one of many different methods, delivers a convenient and easy to understand way to visualize large amounts of geo-referenced data.

Appendix A

Further Information

List of all too frequent words:
control, two, set, show, process, number, one, high, paper, design, different, information, use, based, also, provide, support, technique, system, research, application, efficient, performance, several, approach, method, real, used, multiple, may, propose, however, framework, important, user, new, given, data, model, present, case, novel, e, network, algorithm, level, many, study, work, well, analysis, simulation, large, implementation, time, problem, order, experimental, first

Map design used for the scientific research atlas:
The design of choice is the CartoDB-Positron map. It is published under the *Creative Commons Attribution (CC BY 3.0)* license and therefore free to use.
`www.cartodb.com`

List of Figures

List of Tables

Table of Acronyms

OSM OpenStreetMap

DBLP Digital Bibliography & Library Project

BOW Bag of Words

LDA Latent Dirichlet Allocation

PLSA Probabilistic Latent Semantic Analysis

NLTK Natural Language Toolkit

tf term frequency

idf inverse document frequency

tf-idf term frequency–inverse document frequency

Bibliography

[1] *Aminer Citation Network Dataset*. `https://aminer.org/citation`. Accessed: 2016-05-13.

[2] *DBLP Computer Science Bibliography*. `http://dblp.uni-trier.de/`. Accessed: 2016-05-13.

[3] *English Wikipedia*. `https://en.wikipedia.org/wiki/Main_Page`. Accessed: 2016-06-18.

[4] *Palmetto*. `http://aksw.org/Projects/Palmetto.html`. Accessed: 2016-06-18.

[5] Ahern, S., M. Naaman, R. Nair and J.H.I. Yang: *World explorer: visualizing aggregate data from unstructured text in geo-referenced collections*. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 1–10. ACM, 2007.

[6] Aloise, Daniel, Amit Deshpande, Pierre Hansen and Preyas Popat: *NP-hardness of Euclidean sum-of-squares clustering*. Machine Learning, 75(2):245–248, 2009.

[7] Barber, C. Bradford, David P. Dobkin and Hannu Huhdanpaa: *The Quickhull Algorithm for Convex Hulls*. ACM Transactions on Mathematical Software (TOMS), 22(4):469–483, 1996.

[8] Bashein, Gerard and Paul R. Detmer: *Graphics Gems IV*. chapter Centroid of a Polygon, pages 3–6. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[9] Blei, David M., Andrew Y. Ng and Michael I. Jordan: *Latent Dirichlet Allocation*. Journal of Machine Learning Research, 3:993–1022, 2003.

[10] Brin, Sergey and Lawrence Page: *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[11] Castellà, Quim and Charles A. Sutton: *Word Storms: Multiples of Word Clouds for Visual Comparison of Documents*. Computing Research Repository (CoRR), abs/1301.0503, 2013.

[12] Chiang, Mark Ming-Tso and Boris G. Mirkin: *Intelligent Choice of the Number of Clusters in K-Means Clustering: An Experimental Study with Different Cluster Spreads.* Journal of Classification, 27(1):3–40, 2010.

[13] Elhadad, Michael: *Natural Language Processing with Python.* Computational Linguistics, 36(4):767–771, 2010.

[14] Feinberg, Jonathan: *Wordle.* In Steele, Julie and Noah Illinsky (editors): *Beautiful Visualization*, chapter 3. O'Reilly Media, 2010.

[15] Hadiji, Fabian, Kristian Kersting, Christian Bauckhage and Babak Ahmadi: *GeoDBLP: Geo-Tagging DBLP for Mining the Sociology of Computer Science.* Computing Research Repository (CoRR), April 2013.

[16] Hoffman, Matthew D., David M. Blei and Francis R. Bach: *Online Learning for Latent Dirichlet Allocation.* In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 856–864, 2010.

[17] Hofmann, Thomas: *Probabilistic Latent Semantic Analysis.* In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 289–296, 1999.

[18] Jaffe, Alexander, Mor Naaman, Tamir Tassa and Marc Davis: *Generating summaries and visualization for large collections of geo-referenced photographs.* In Wang, James Ze, Nozha Boujemaa and Yixin Chen (editors): *Multimedia Information Retrieval*, pages 89–98. ACM, 2006.

[19] Kanungo, Tapas, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman and Angela Y. Wu: *An Efficient k-Means Clustering Algorithm: Analysis and Implementation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):881–892, 2002.

[20] Lloyd, Stuart P.: *Least squares quantization in PCM.* IEEE Transactions on Information Theory, 28(2):129–136, 1982.

[21] Lo, Rachel Tsz-Wai, Ben He and Iadh Ounis: *Automatically Building a Stopword List for an Information Retrieval System.* Journal of Digital Information Management (JDIM), 3(1):3–8, 2005.

[22] Lohmann, Steffen, Jürgen Ziegler and Lena Tetzlaff: *Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration.* In Gross, Tom, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe A. Palanque,

Raquel Oliveira Prates and Marco Winckler (editors): *Proceedings of INTERACT (1)*, volume 5726 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 2009.

[23] MacKay, David J. C.: *Information Theory, Inference & Learning Algorithms*, chapter 20, An Example Inference Task: Clustering, page 284–292. Cambridge University Press, New York, NY, USA, 2002.

[24] Mihalcea, Rada and Paul Tarau: *TextRank: Bringing Order into Texts.* In *Proceedings of Empirical Methods for Natural Language Processing*, pages 404–411, 2004.

[25] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Gregory S. Corrado and Jeffrey Dean: *Distributed Representations of Words and Phrases and their Compositionality.* In Burges, Christopher J. C., Léon Bottou, Zoubin Ghahramani and Kilian Q. Weinberger (editors): *NIPS*, pages 3111–3119, 2013.

[26] Mimno, David M., Hanna M. Wallach, Edmund M. Talley, Miriam Leenders and Andrew McCallum: *Optimizing Semantic Coherence in Topic Models.* In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 262–272, 2011.

[27] N., Aletras and Stevenson M.: *Evaluating topic coherence using distributional semantics.* In *Proceedings of the 10th International Conference on Computational Semantics (IWCS'13) Long Papers*, pages 13–22, 2013.

[28] Röder, Michael, Andreas Both and Alexander Hinneburg: *Exploring the Space of Topic Coherence Measures.* In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 399–408, 2015.

[29] Salton, Gerard and Christopher Buckley: *Term-weighting approaches in automatic text retrieval.* Information Processing and Management, 24(5):513–523, 1988.

[30] Smyth, Padhraic: *Clustering Using Monte Carlo Cross-Validation.* In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 126–133, 1996.

[31] Tang, Jie, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang and Zhong Su: *ArnetMiner: Extraction and Mining of Academic Social Networks.* In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA, 2008. ACM.

[32] Viégas, Fernanda B., Martin Wattenberg and Jonathan Feinberg: *Participatory Visualization with Wordle.* IEEE Transactions on Visualization and Computer Graphics, 15(6):1137–1144, 2009.