LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT FÜR
PROGRAMMIER- UND MODELLIERUNGSSPRACHEN

Institut für Informatik

The Analytics Center: Devising a Citizen Science Data Mining Tool for the ARTigo Image Tagging Project

Florian Hoidn

Bachelorarbeit

Beginn der Arbeit:	15.4.2014
Abgabe der Arbeit:	16.9.2014
Betreuer:	Prof. Dr. François Bry
	Dr. Clemens Schefels

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 15.9.2014 Florian Hoidn

Zusammenfassung

Die durch das ARTigo-Projekt generierte Schlagwörterdatenbank ist nicht nur für die Kunstwerk-Bildersuche erforderlich, sie enthält auch kunsthistorisch relevante Informationen, die durch geeignete Methoden des Datenabbaus erschlossen werden können. Ziel der Arbeit ist die Entwicklung eines Analysezentrums, das es dem Benutzer gestattet, die statistische und clustering theoretische Beschaffenheit der ARTigo Datenbank frei zu erforschen. Für die Datenanalyse werden dem Benutzer fünf Komponenten zur Verfügung gestellt, aus denen sich das ARTigo-Schlagwort-Analysezentrum zusammensetzt: Schlagworthäufigkeitsgraphen, Poisson-Überschätzungswerte, Assoziationsregeln, Schlagwort-Vektor Nachbarschaften und Schlagwort Cluster nach dem DBScan Verfahren. Alle fünf Komponenten werden im Zuge dieser Arbeit vorgestellt und im Rahmen der ARTigo Analytics Center Web-Applikation implementiert.

Abstract

The tag database generated by the ARTigo project is not only required for artwork image search, it also contains art historically relevant information that can be tapped with the help of suitable data mining techniques. This work aims at elaborating an analytics center that allows users to freely explore the statistical and clustering theoretical structure of the ARTigo database. For this data mining task, five components that together form the analytics center are made available to the user: tag-frequency graphs, Poisson overdispersion values, association rules, tag vector neighborhoods, and tag clusters generated by the DBScan clustering Method. In the course of this work, all five components will be introduced and implemented as a part of the ARTigo Analytics Center Web application.

Danksagung

Ich bedanke mich bei Dr. Clemens Schefels für die hervorrangende Betreuung dieser Arbeit. Ebenso bedanke ich mich bei Prof. Dr. François Bry für die Unterstützung und die Möglichkeit, ein so schönes Bachelor-Projekt wie dieses machen zu dürfen. Den Mitwirkenden am ARTigo Projekt und allen Teilnehmern der PMS Kolloquien seien die hilfreichen Anmerkungen und Kommentare gedankt. Nicht zuletzt gilt mein Dank den ARTigo Nutzern, ohne deren Engagement auf der ARTigo Spieleplattform diese Studienarbeit, die mir viel Freude bereitet hat, nicht möglich gewesen wäre.

*Contents

1 Introduction

The ARTigo[1] social image tagging project is a "game with a purpose", or GWAP, inspired by Luis von Ahn's ESP Game, [LvA11]. It consists of a growing library of tagging games together with a tag based image search. The idea was to create an ecosystem of games in which image tags are not only generated but also further refined by ARTigo players. One part of this ecosystem is the ARTigo game which gave the project its name. In it, players competitively tag images of artworks with keywords they consider descriptive or fitting in some intuitive sense. They score points if the same keywords have been used before by other players. The games are played online via a Web browser. Details on ARTigo's features can be found in [WBBL13]. The project is a collaboration of various institutes of the LMU and was funded by the DFG.
By participating in ARTigo's various games, users create keyword databases which help to categorize the featured artworks by their content, a perceptual task that could not be handled algorithmically. In return, ARTigo provides its users with smart and enjoyable games. And, of course, it is the users who profit from ARTigo's semantic image search that is fueled by their tags. The platform offers a huge collection of appealing artworks and the game mechanics encourage an entertainingly straightforward approach to describing these images. High user participation and fast growing databases suggest that the project thus far succeeds in keeping players entertained. Also, the games in ARTigo's expanding library are designed to complement one another. In sum, ARTigo aims at creating increasingly detailed and subtle semantic data about the featured artworks.
Of course, generating data is not an end in itself and providing access to the available Information is a task in its own right. While ARTigo features an image search based on information retrieval, it currently lacks a tool for querying, visualizing and highlighting salient information hidden in the "big data" produced by the project. This shall be changed in the course of this work. Various means for finding and representing patterns and clusters within the data will be explored and implemented as a new feature on the project's website: the *ARTigo Analytics Center*. The tool will help ARTigo users to spot thematic trends within certain artistic epochs, determine how specific and, therefore, useful to the ARTigo project their tags are, and discover clusters and interesting associations between the motives that they have helped to recognize within the image collection.

The remainder of this work is structured as follows: Section 2 on related work connects the idea behind the Analytics Center with the spirit of the ARTigo project and GWAPs in general. It also introduces two applications that serve as a model for the Analytics Center's frequency plot component, namely Harvard's Bookworm[2] and Google's NgramViewer[3] project.
Section 3 introduces the theory behind the Analytics Center's five components: frequency plots in the spirit of the Ngram Viewer, the Poisson overdispersion value of a tag, the confidence of association rules that hold between sets of tags, the cosine similarity measure and

[1] http://www.artigo.org
[2] http://bookworm.culturomics.org
[3] https://books.google.com/ngrams

the DBScan cluster expansion algorithm.
Chapter 4 details how the ARTigo Analytics Center is implemented. The chapter roughly mirrors the preceding one. It documents implementation details of each component introduced in the theoretical section.
Finally, in Chapter 5, possible future projects regarding the Analysis Center are sketched out.

2 Related Work

Implementing a data mining tool for the ARTigo project makes a lot of sense. As we shall see, it complements the idea behind a citizen science project like ARTigo perfectly, because it showcases how intriguing the results of the users' efforts are. Also, it may further a general appreciation of good and interesting tags among players of the ARTigo game, which could improve the quality of tags that are used in game sessions.
When it comes to the implementation and design of the Analytics Center, there is number of applications with similar ambitions to provide users with powerful tools for knowledge discovery and representation. We will pick out one popular technique that is commonly seen in such projects, namely that of presenting n-gram word frequency plots to the user. The technique was introduced by applications such as Bookworm[4] that was developed at Harvard university and its successor the NgramViewer[5] by Google labs, [MSA+11]. As the Analytics Center features similar frequency plots for tags, it is worth while to look into these applications in greater detail.

2.1 Games with a Purpose

In [LvA11], Luis von Ahn promotes a paradigm shift in favor of human computation in the computer sciences. He came to the realization that humans do far better than even the best known algorithms when it comes to a certain class of tasks, to which he refers to as artificial intelligence (AI) tasks. Among these are: perceptual tasks, natural language analysis, and reasoning and planning. And not only are people generally really good at these tasks, it turns out that they are also eager to help, if one provides them with the right incentives. It therefore doesn't come as a surprise, that various applications on the Web try to tap into this potential. Von Ahn mentions Amazon's Mechanical Turk, a Web platform where entrepreneurs offer small amounts of money for the completion of AI tasks to the site's community. Similar monetary incentives can be found on platforms like TopCoders, where participants solve programming tasks in sponsored online competitions. Games with a purpose or GWAPs follow a somewhat different approach. There, the incentive to participate isn't monetary reward but rather an engaging gaming experience and the greater good of collectively created knowledge databases. Examples of GWAPs are the foldIt project, in which players fold three dimensional models of proteins into a stable form, von Ahn's ESP game, and ARTigo, both of which aim at creating tag databases for image search.
The relation between the ARTigo project and the Analytics Center is best understood, if one considers what incentives players have to participate on the ARTigo platform. According to Alexander Quinn and Benjamin Bederson, [QB11], users of human computation applications and GWAPs may, among other things, act out of altruism − viz. a motivation to help others just for the sake of doing them a favor − or seek entertainment. Christoph Wieser adds educational interests and the will to improve image search to the list of incentives particularly relevant for ARTigo users, [Wie14].
Now, while the Analytics Center isn't itself a human computation application or a GWAP,[6]

[4] http://bookworm.culturomics.org
[5] https://books.google.com/ngrams
[6] Although it is conceivable to derive GWAPs from it, see Section 5.3.

all these possible incentives could benefit greatly from it, as it gives people access to ARTigo's database and allows them to discover truly interesting results about artworks as well as other peoples' play styles that could never have been obtained if it wasn't for their collective effort in ARTigo's games. Thus, it is only fair to thank the players for their altruistic ambitions by allowing them to explore the intriguing digitized collective knowledge that they have helped to create. Also, it will hopefully appeal to ARTigo users and warrant an ongoing interest in the platform, if the Analytics Center aims at encouraging art historical inquiries that are fun to engage in and educational.

2.2 Bookworm and the Google Ngram Viewer

Let us turn towards a concrete way in which the evolution of word usage over time is sometimes illustrated, namely by plotting so called n-grams of word frequencies in textual collections. An n-gram, as defined in [MSA+11], is a sequence of n words separated by $n-1$ white spaces. A 1-gram is, e.g., any sequence of non-whitespace characters. English words are 1-grams and so are numbers, arbitrary character sequences or typographical errors. A 2-gram is a chain of two such 1-grams, and so on. Statistical tools like Bookworm[7] and the NgramViewer[8] compute and plot frequencies of n-grams up to a certain length $-$ 5 in Google's case $-$ as a function of time. At each point in time, the frequencies are computed relatively to the total number of words, i.e., 1-grams, within a collection of books that were published that same year.[9] Google's Ngram Viewer is based on more than five million digitized books from Google Books. In order to filter out misspellings, Google only considers an n-gram at all, if it occurs at least 40 times within the corpus of words. In this way, the frequencies of every single n-gram up to length 5 within the corpus of books is precomputed and stored in a vast database, ready for download.[10]
The Ngram Viewer was initiated by Jean-Baptiste Michel et al. as a part of Havard's Culturnomics project. Bookworm is based on an earlier version of the technology. As opposed to Google's application, it can freely be used on any collection of digitized books. A typical plot is depicted in Figure 2.2. Bookworm offers a list of input boxes to and from which the user can easily add or delete boxes. Each n-gram entered into one of these boxes is rendered in the interactive plot. Hovering the mouse cursor over one particular point of a line causes the whole line to appear bigger. Also, an info box will pop up at that point, containing, e.g., the exact coordinates of that point. Clicking creates a list with links to each book, published that year and containing the queried n-gram. By dragging and releasing the mouse one can zoom into an interval on the x-axis.
Features like this make Bookworm a noteworthy contender to Google's Ngram Viewer. What is unique about the latter, though, is its query language.[11] The Viewer offers just one input box. Different n-grams must be separated by commas. Google offers a variety of options regarding semantic and syntactic aspects of the queried n-grams. There is, e.g., an option to toggle case sensitivity. Furthermore, special modifiers allow the user to query

[7] http://bookworm.culturomics.org
[8] https://books.google.com/ngrams
[9] A text with k words, contains $k-(n-1)$ n-grams. As k is normally much greater than n, the error in the computed frequency is marginal.
[10] http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
[11] A specification is given at https://books.google.com/ngrams/info.

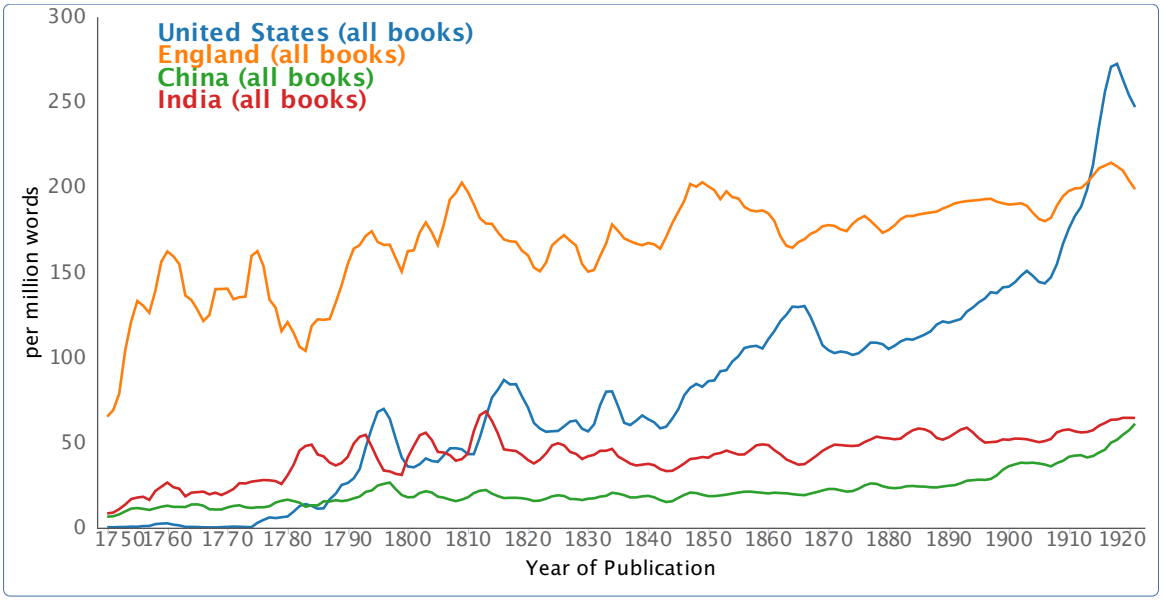


Figure 1: A typical Bookworm n-gram line plot based on the Open Library collection.

different grammatical word forms, or disambiguate a query grammatically, e.g., by adding "_VERB" to a word, so that only uses as a verb are counted. It is even possible to have Google count only those sentences, where certain predicate-object relationships hold. For instance, querying "object=>predicate" yields the frequencies of sentences where "predicate" is applied to "object".
Furthermore, it is possible to apply arithmetic operators on individual n-grams in order to scale them or merge them into one. The available operators are $+, -, *, /$ with the interpretation that at each time point the values of two plots are added, subtracted, multiplied or divided. Thus, for instance, in the year 2000 the frequency of "car" was ca. 0.009% and that of plane around 0.004%. Hence, (car + plane) yields 0.009% + 0.004% = 0.013% and (car + plane)*10000 yields 130% in 2000. Details are once again specified on Google's Web page.[12]
As the ARTigo database contains tags, i.e., mostly individual 1-grams and hardly any sentences, grammatical structure will presently not be our concern − although such features will be reconsidered in the future work section. There are certain other features of the Ngram viewer, though, that presently are not but certainly could be applied to ARTigo's data set. For example, in the Ngram Viewer, each n-gram may contain a wild card instead of one of its n words. If it does, the query yields the ten most common n-grams matching the queried pattern.

[12] https://books.google.com/ngrams/info, accessed september 15, 2014.

3 Theory Behind the ARTigo Analytics Center

The new Analytics Center consists of five different components, namely tag frequency plots, the Poisson overdisperion of a queried tag, the confidence of association rules between sets of tags, a ranking of the closest, viz. most similar, tags, and a DBScan clustering in its proximity. In this section, the theory behind these components is introduced.
The idea behind this choice is to provide the user with a wide array of potentially interesting statistical and structural data about ARTigo's tags. Tag frequency plots can be generated with relative ease and require hardly any explanations, which made them an obvious choice right from the start. It was also clear from early on that it would be worthwhile to provide a means for assessing tag specificity, as this matters a lot in information retrieval contexts such as ARTigo's image search. The Poisson overdispersion is such a means, and a very powerful one at that. It wasn't immediately clear whether a component for the analysis of association rules, which are relatively costly computation wise, could be offered. However, this turned out to be feasible, from which the Analytics Center benefits greatly. Regarding more subtle structural information, similarity (or distance) measures form the very basis of any clustering or knowledge discovery process in a database. The fourth component, a ranking of the nearest neighbors of a tag according to the cosine similarity measure, serves as a good point of departure for actual knowledge discovery. To this latter end, the fifth component, DBScan cluster expansion, then provides a genuine clustering technique. Although conceptually simple, it is particularly well suited for the exploration of thematic clusters in the tag database. One of its benefits is, that it doesn't require an entire clustering of the database, but can be used to uncover only the structure to which the queried tag belongs.

3.1 Tag Frequency Plots

The Analytics Center offers frequency charts in the spirit of the Ngram Viewer. The idea

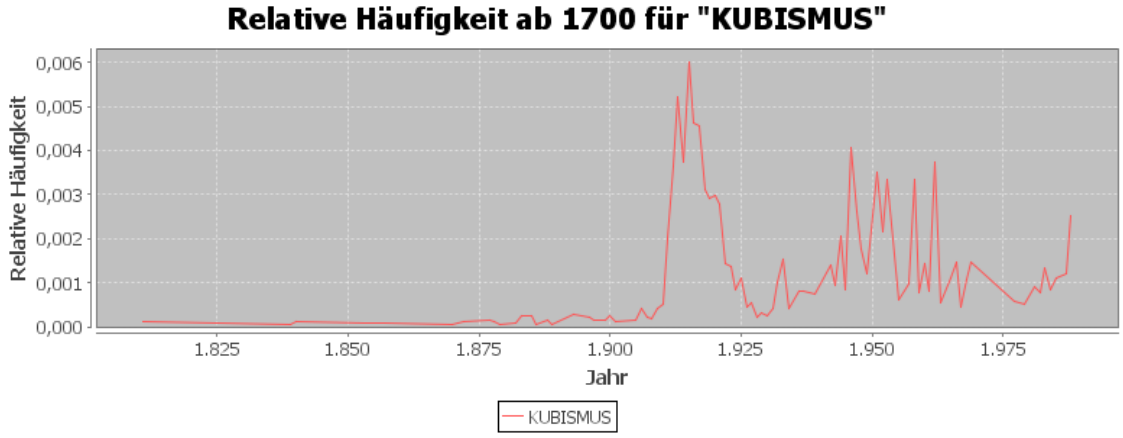


Figure 2: A tag frequency chart for the word "KUBISMUS" (viz. cubism) in the ARTigo database. Only tags that have been used five times or more were considered.

is, of course, to help discover thematic trends throughout the epochs, thus to visualize the "zeitgeist" of a given time and how it was affected by historical events. For instance, Figure 3.1 displays a line plot for the frequency of the word "KUBISMUS", i.e. cubism, an epoch that began 1907 and lasted to 1914, ended by the outbreak of World War I. These historical facts do seem to be reflected within the graph, which peaks during that time. The tag frequencies also suggest somewhat of a revival of the typical forms of this art style after World War II finally lost its grip on Europe. At least this is what ARTigo players seem to have seen within paintings and sculptures dated post 1945.
Obtaining tag frequencies over time is relatively straightforward. Like in the Ngram Viewer, each year can be mapped to the set of artworks that are datable back to it.[13] Once identified, the artworks of a given year can be associated with the set of all individual tagging events that occurred for them in game sessions.[14] In order to stay true to ARTigo's database scheme let the function $\# : validTags \times timeInterval \to \mathbb{N}$ be defined as follows:

$$\#(tag, year) := \\ |\{t \in tagging | tagName(t.tag\_id) = tag \wedge imgDate(t.resource\_id) = year\}|$$

Which yields the number of times tag was used in $year$.[15] Here, $validTags$ is a set of tags that are valid in a sense that can be adjusted to one's needs. Thus, it may contain tags that have been used a certain number of times, are in German, have been used within a certain period of time by the users or conform to some other constraining factors.[16] The set $timeInterval$ is meant to indicate that the plotted time interval can of course be specified. The function

$$tagFrequency(tag, year) := \frac{\#(tag, year)}{\sum_{t \in validTags} \#(t, year)} \tag{1}$$

then yields the frequencies of interest.
As ARTigo users are free to enter tags that consist of multiple words and the sets of tags are otherwise unstructured, no extra formalism for n-grams is required. Note, however, that longer word sequences rarely occur, so that the statistical significance of 2- or more gram queries may suffer.

3.2 The Poisson Overdispersion

The Poisson overdispersion measures how much the use of a tag concentrates on a few pictures. It does so by comparing the actual distribution of its uses with what one would

[13] See Section 4.1 for difficulties related to the dating of the featured artworks.
[14] Note that while each tag exists only once in the database, stored in a table called *tag*, each artwork may have been tagged multiple times with an individual tag. Thus, there are multiple tagging events relating individual artworks and tags. These tagging events are stored in a table called *tagging*.
[15] The function $tagName(tid)$ yields the actual tag for a given tag identifier, $imgDate(rid)$ maps ARTigo image identifiers to their dates of creation. These auxiliary functions have been implemented in the Postgres SQL database.
[16] Adjusting the constraints on $validTags$ may in itself constitute a means to obtain results of theoretical interest. For instance, one could consider only those tags as valid, that have themselves been used within a certain time, in order to assess how ARTigo users have perceived images at different times. I thank Dr. Norbert Eisinger for pointing this option out to me.

expect if the tag didn't describe the content of the images at all but was used completely randomly instead. Thus, it indicates how specific, yet, at the same time easy to recognize and salient the feature that a tag describes is to the players. The technique has been applied to ARTigo's data set by Elena Levushkina, [Lev13]. To give a few motivating example for this interesting measure, in [Lev13] it was discovered that "ROLLSTUHL" (wheelchair) and "BILLARD" (billiards) are among the most specific tags according to that measure. These results can be revisited and confirmed with the help of the Analytics Center. Also, as ARTigo's tag and image database has grown since the time of Levushkina's research, new tags with high Poisson overdispersion values have emerged. One example is "STABHOCHSPRUNG" (pole vault). Searching for any of these keywords indeed reveals small numbers of matching images that feature the related motives in an arguably salient way.
As mentioned, the idea behind this measure is to assume that for a given tag, a tagging event is equally likely to occur for any of the artworks − as if the content of an image would not matter at all. Under this assumption, one computes how many artworks should have been tagged, given the total number of times that it has been applied in game sessions. To illustrate, if a tag was applied only one single time, then, of course, it can also have been applied to only one single artwork. However, if it was used, say, a hundred times, then one would expect the number of artworks to which it was applied to be larger − but not quite one hundred.[17] Once this number is determined, one compares it to the number of images that were actually associated with the given tag. If it is much smaller, the assumption that the content of an image is irrelevant for the users' decision to apply the tag must have been far off from the truth.
The first step towards obtaining the expected number of tagged images under the, presumably false, assumption of equal probabilities, is to apply the Poisson distribution. The standard motivating example for this distribution goes as follows: assume that, on average, the phone rings 3 times per hour − thus, 3 being the expected number of phone rings per hour − how likely is it that it rings, say, five times? The answer, ca. 10% probability, is computed with the Poisson distribution.[18] Transforming this distribution to taggings per artwork rather than phone calls per minute, we obtain

$$p_{tag}(k) = e^{-\lambda_{tag}} \cdot \frac{\lambda_{tag}^k}{k!}$$

where $\lambda_{tag} := \frac{cf_{tag}}{N}$ is the average number of times each image is tagged with tag, which, assuming equal probabilities for all images, simply is the expected number.[19] Now, as $p_{tag}(k)$ yields the probability that an image is tagged k times, $k \in \mathbb{N}_0$, $p_{tag}(0)$ is the probability that an image isn't tagged at all. Accordingly, $1 - p_{tag}(0)$ is the probability that an image is tagged at least once and $N * (1 - p_{tag}(0)) = N * (1 - e^{-\lambda_{tag}})$ is the expected number of images tagged with tag. Division by tag's document frequency, df_{tag}, viz. the actual

[17] Consider the following: the image to which the tag was applied in the n^{th} tagging has the same chance as all the others to be tagged again in one of the remaining $100 - n$ taggings.

[18] $p(5) = e^{-3} \cdot \frac{3^5}{5!} = 0.1008\ldots$

[19] The connection to ARTigo's database can be established via relational algebra as introduced in [Cod70]: $cf_{tag} := |\sigma_{tagName(tag\_id)=tag}(tagging)|$ is the collection frequency of tag, $N := |\pi_{resource\_id}(artresource \bowtie tagging \bowtie validTags)|$ is the total number of artworks that have received a valid tag.

number of images tagged with it,[20] yields the Poisson overdispersion

$$poissonOverdispersion(tag) := \frac{N \cdot (1 - e^{-\lambda_{tag}})}{df_{tag}} \tag{2}$$

There are different possible use cases where this measure might come in handy. One may, e.g., compare the recognizability of different artists or art styles. Popular German painter Franz Marc is, e.g., easily recognized, much more so than his less popular contemporary August Macke. Picasso, who isn't yet contained in ARTigo, is nevertheless recognized more often than some slightly less renowned artists who are. Cubism is more easily recognized than, e.g., expressionism and so on. Players might be interested in figuring out what motives others tend to recognize. ARTigo game designers could use the measure in a possible reward system for more specific tags.
A potentially spurious effect was observed, though: not all of the tags with very high Poisson overdispersion values seem to make sense at first glance. For instance, "stephanie" has a remarkably high value of almost 40 − pole vault, for instance, "only" has close to 18, which already sets it apart very distinctively from most tags. Not only is this an oddly high value, it also seems to be in complete dissonance with the intuition that a high Poisson overdispersion should indicate high specificity and salience. However, a keyword search for "stephanie" reveals the cause of this: the word itself appears in one particular portrait of a young woman, making it a very salient choice. There are similar cases whenever actual strings of characters occur in a picture.

3.3 Association Rules

The third component of the Analytics Center enables users to engage in association rule mining, also known as market basket analysis, [Zim14]. Retailers like to use this technique to identify products that are frequently bought together. There, one is interested in the confidence of an association rule, which determines how likely it is that a customer will buy particular products given what she already has in her shopping cart. Accordingly, in the Analytics Center, an association rule is the probability with which an artwork is tagged with a certain set of tags, given that it has been tagged with another set of tags.
Association rules are a special sort of conditional probability. Let $validTags$ once again be the set of tags in which we are interested in and let $X, Y \subseteq validTags$, $X \cap Y = \emptyset$, be disjoint sets of such tags. Furthermore, let

$$cover(X) := \\ \{a \in artresource | \forall x \in X \exists t \in tagging(t.resource\_id = a.id \wedge t.tag\_id = x.id)\}$$

be the set of artworks that have been tagged with each tag in X at least once. On this basis we define the confidence in an association rule as follows:

$$confidence(X \Rightarrow Y) := \frac{|cover(X) \cap cover(Y)|}{|cover(X)|} \tag{3}$$

[20] $df_{tag} := |\pi_{resource\_id}\sigma_{tagName(tag\_id)=tag}(artresource \bowtie tagging \bowtie validTags)|$

The antecedent parameter, X, is sometimes referred to as the rule's "body", while the consequent, Y, constitutes its "head". Intuitively, an association rule is the relative frequency of artworks that have been tagged with all tags in X and in Y among the artworks that have − at least − been tagged with all tags in X.[21] The number of artworks in $cover(X)$ is often referred to as the support of X, viz. $support(X) := |cover(X)|$.
A couple of things should be pointed out: First, according to the definitions above, $cover(\emptyset)$ is the entire collection of artworks (as the set membership condition is trivially true). Thus, $confidence(\emptyset \Rightarrow Y)$ yields the relative frequency with which artworks have been tagged with each $y \in Y$ at least once, which is a convenient information that will be displayed by default. Second, as the support of a set of tags monotonically approaches and (normally) reaches zero with increasing set size, association rules aren't always defined, which necessitates division by zero handling in the implementation. Third, association rules do not account for the number of taggings associating a given artwork and tag. This is a good thing, in a way, because it sets association rules further apart from the vector space measures introduced below. Thus, being similar in meaning according to cosine similarity doesn't necessarily imply a strong association, although the notions are intimately related.
For instance, the confidence of the rule $\{Portrait\} \Rightarrow \{Mann\}$, viz. the likelihood that an image displays a man given that it is a portrait, is around 69% − whereas women seem to be portrayed in only around 42% of the cases.

3.4 Inverted Index and Similarity

For information retrieval (IR) in ARTigo, i.e., image search, each artwork is represented as a vector in $\mathbb{N}_0^k$, where k is the number of valid tags. Each of the $n = 1 \ldots k$ integer values that compose the vector representation of an artwork holds the number of times the n^{th} tag has been associated with the picture. Intuitively speaking, an artwork is identified with its description − or rather, with the tag cloud that serves as its description in ARTigo. Given two such vectors, one can compute how similar or close they are in the vector space. If they turn out to be close, there is a good chance that the corresponding pictures are fairly similar in an intuitive sense. A search query is then represented in this same vector space, such that the most similar images can be computed and returned as an answer to the query, [LM06] and [Sin01] describe various IR techniques, [Lev13] and [Wie14] discuss IR in ARTigo.
Transposing a document-term matrix yields counting vectors that contain information about the tags rather than the artworks. Intuitively speaking, one attempts to understand the meaning of a term by considering how it is used, i.e., to what sort of things it applies, as opposed to understanding what is depicted in an image with recourse to how it is described. For convenience's sake, we define

$$invertedIndex(tag, img) := \\ |\{t \in tagging | tagName(t.tag\_id) = tag \wedge t.resource\_id = img\}|$$

The function simply returns the number of times that the artwork img has been associated with tag.

[21] Note how closely association rules resemble conditional probabilities. Assuming a suitable probability function P, the connection is the following: $X \Rightarrow Y = P(cover(Y)|cover(X))$.

One of the most common metrics, viz. distance measures, in such vector spaces is of course the euclidean distance. In our context it can be defined as follows:

$$euclideanDistance(t_1, t_2) := \sqrt{\sum_{i \in artresource} (invertedIndex(t_1, i) - invertedIndex(t_2, i))^2} \tag{4}$$

One common IR problem is that the vector spaces are extremely high dimensional and distance computations therefore expensive. A common solution is the cosine similarity measure: two vectors are considered similar, if the angle between them is small, i.e., if its cosine is close to one. Thus,

$$cosineSimilarity(t_1, t_2) := \frac{\sum_{i \in artresource} invertedIndex(t_1, i) \cdot invertedIndex(t_2, i)}{\sqrt{\sum_{i \in artresource} invertedIndex(t_1, i)^2} \cdot \sqrt{\sum_{i \in artresource} invertedIndex(t_2, i)^2}}$$

yields the cosine similarity of two tags, t_1 and t_2. Note that the denominator is the product of the euclidean lengths of the vectors. Due to a relatively great variance in vector lengths, normalizing these vectors − which is often a convenient trick to boost performance − doesn't lend itself. In any way, the cosine distance can then be defined as follows:

$$cosineDistance(t_1, t_2) := 1 - cosineSimilarity(t_1, t_2) \tag{5}$$

The nearest neighbors of a given tag are oftentimes conceptually related to it in a surprisingly comprehensible way. In order to introduce a running example that will be reused below: when querying, e.g., the nearest neighbors of "MEER", viz. sea, one obtains in this order: coast, waves, beach, water, sail, ocean, shore, ship and so on − viz. suitably maritime notions. As the radius around a tag increases, less and less interesting neighbors begin to crop up − e.g., colors that happen to be vaguely related to the tag or very common motives like "MENSCHEN", i.e., people.

3.5 DBScan Cluster Expansion

The fifth component of the Analytics Center provides the user with a clustering mechanism that is based upon − but goes well beyond − the analysis of a tag's immediate neighborhood. Exploring clusters falls within the domain of knowledge discovery in data, or data mining. The notion of a cluster is a rather intuitive one: it describes particularly dense areas in the vector space representation of a dataset. As some of the pioneers of data mining have conceived it, [FPSS96], clustering is a nontrivial process at the end of which stands the discovery of new and interesting patterns within the data. "Nontrivial" means that there is no general way to arrive at an interesting clustering. Clustering often requires *ad hoc* assumptions such as how many clusters one expects to see within the dataset. Furthermore, it isn't easy to formally specify the intuition behind what counts as a cluster and what doesn't, as the concept is inherently vague. The beneficial upshot is, however, that there

exists a great variety of different clustering algorithms to choose from.
One such algorithm is DBScan, [EKSX96], a density based procedure that is particularly well suited for the Analytics Center. It discovers densely populated areas of arbitrary shapes by clustering data points which can be reached on a densely populated path in the vector space. Figure 3 illustrates the idea, [Zim14]. The data is given in two dimensions, viz. they lie on the plane at the base of the depicted "volcano". Its height reflects the density of the points at each spot in the two dimensional vector space. The gray plane that cuts right through the top of the volcano reflects a possible parameterization of the DBScan algorithm. Intuitively speaking, DBScan considers each connected region above said plane as one cluster. Thus, in the picture, one cluster has been discovered − namely the volcano's rim. Little bumps that may exist below the gray plane aren't densely populated (viz. high) enough to count as a cluster under the given parameterization.

The algorithm comprises a method that expands a cluster of a desired density around

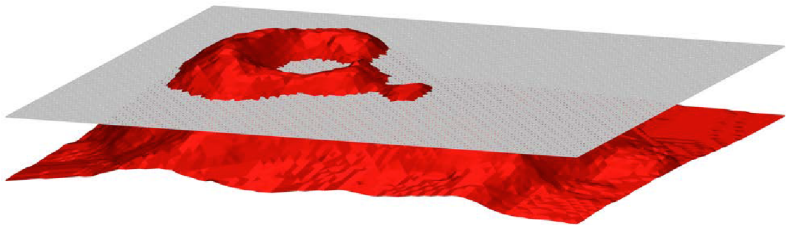


Figure 3: DBScan discovering circularly shaped area. Image source [Zim14].

a given tag. This can be carried out independently from the expansion of other clusters and is, therefore, relatively fast. The great efficiency of DBScan's cluster expansion is one important advantage over other clustering approaches, as it allows users of the Analytics Center to experiment with different parameterizations live and discover completely new clusters that haven't been precomputed. Listing 1 specifies the algorithmic details:

Data: tag, ε, $minPts$
Result: Cluster, C, around tag, such that there are $minPts$ points in ε range of each contained point.

```
seeds := rangeQuery(tag, ε) ;
if |seeds| ≥ minPt then
    C:=C ∪ seeds ;
    seeds := seeds\ {tag} ;
    while |seeds|0 do
        o:=nextBest(seeds);
        neighbors := rangeQuery(o, ε);
        if |neighbors| ≥ minPt then
            seeds:= seeds ∪ neighbors ;
            C :=C ∪ neighbors ;
        end
        seeds := seeds\ {o} ;
    end
end
return C;
```

Algorithm 1: Cluster expansion based on specification in [Zim14].

Intuitively, the algorithm checks whether there are sufficiently many neighbors at a certain distance around the queried tag. If so, it adds the tag to the cluster and proceeds by checking the same thing for all those neighbors nearby, the neighbors of the neighbors and so on.

What is interesting about this procedure is that it can uncover relations between tags that aren't immediately similar to one another, but indirectly connected through common aspects or an overarching theme. For instance, the tags "MEER" and "WALD", viz. sea and woods, aren't very close to each other at all. However, with the right parameterization one can uncover a cluster that contains both of them, along with a great variety of tags that describe landscapes − i.e., as part of a landscape theme, if one wills. In order to achieve this, one can first query the nearest neighbors of a tag and then set the DBScan parameterization so that the former, along with its closest neighbors, serve as the seeds of a larger cluster. Returning to the example from above, one can, for instance, expand the landscape cluster around "MEER" by allowing its neighbors coast, waves, beach, and water to be contained, but not sail, as it doesn't strictly describe a landscape feature.

4 Implementation of the ARTigo Analytics Center

To begin with the technical side of this work, let us survey very briefly how ARTigo's GWAP ecosystem and database are composed. As of to date, the ARTigo ecosystem includes three fully developed games that are post beta-status, namely ARTigo, ARTigo Taboo, and Karido. The data they generate flows into an SQL relation called *tagging*. It presently contains over 7.38 million entries and relates ARTigo's images, of which there are over $190 \cdot 10^3$, with $230 \cdot 10^3$ distinct tags, of which about $67 \cdot 10^3$ are in German and have been entered at least twice. Data about the images is stored in the table *artresource*, data about the tags lies in *tag*. The *tagging* table also contains information about the game session in which each individual tagging event occurred, specifying, e.g., who entered the tag and when it was entered. See [Wie14] and [Ste11] for details.
Additionally, three games currently undergo beta testing, ARTigo Quiz, Tag-a-Tag, and Combino. These games aim at refining and linking established tags with respect to individual images. The data they create is stored in tables such as *tagrating* (ca. $128 \cdot 10^3$ entries), *metatagging* ($< 4 \cdot 10^3$ entries), *combination* (ca. $46 \cdot 10^3$) and *combinedtag* (ca. $32 \cdot 10^3$).
Also, each query entered into the Solr-based search engine is stored in a table called *searchquery*. It contains ca. $17 \cdot 10^3$ queries.
From these numbers, the importance of the *tagging* table becomes apparent as it is the largest by far and contains most of the information that is presented in the Analytics Center. However, due to its sheer size it is impracticable to work with this table directly, such that custom made auxiliary tables have to be derived from it for the individual analysis tools to work at an acceptable speed.
Regarding the concrete implementation of the Analytics Center project, it seemed most natural to implement the Analytics Center as similarly as possible to the ARTigo platform, in order to make the former easily embeddable into the latter. Thus, as it stands, the Analytics Center was implemented as a standalone Seam 2.2.0 GA Web application running on a JBoss 4.2.3 GA Web server.

Let us turn to the implementation in more detail. Figure 4 displays how the graphical user interface, the Java implemented Seam components, and the database and application scoped cache relate to one another.[22]
Each of the Analytics Center's five components comes with its own set of tasks regarding implementation. In most data mining endeavors, preliminary steps are required before statistical or clustering tools can be applied, as some portions of the data may be irrelevant or contain errors or unsuitable data formats. In [FPSS96], Usama Fayyad et al. introduced a layered preparation process consisting of selection, preprocessing, and transformation steps that help one to get off the ground and start the actual data mining process. In the Analytics Center, each component requires individual steps of these kinds.
Regarding the graphical user interface, user queries need to be interpreted for each component individually and one needs to decide which parameters users get to input. Also, the

[22] The picture is inspired by a sort of diagram that Dan Allen uses in his thorough Seam introduction [AR09]. It seemed to me to be very fitting for a Seam project, as it shows in great detail how the view, controller, and model components pass messages between one another.

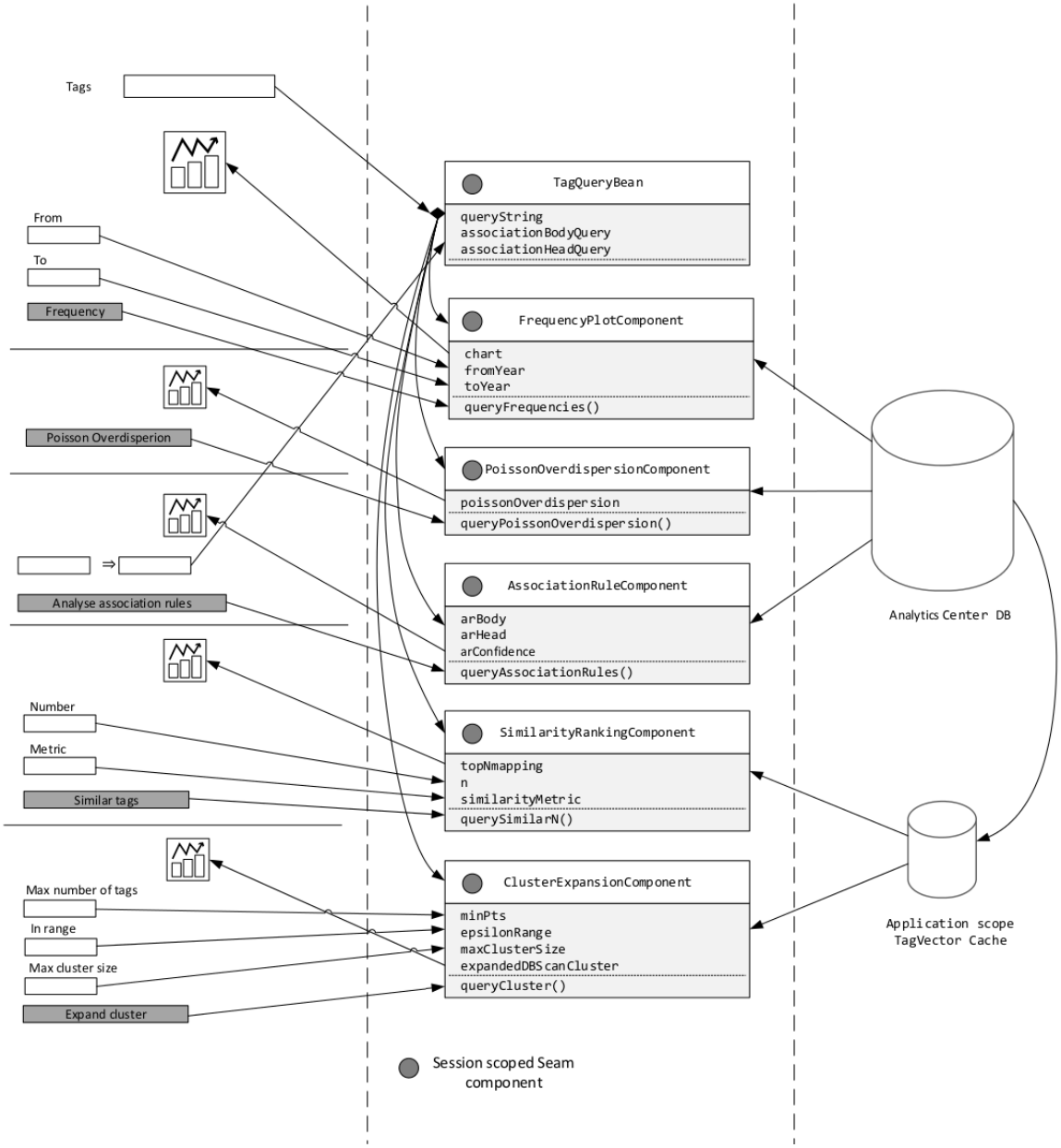


Figure 4: A general description of the structure of the Analytics Center inspired by similar diagrams in [AR09]. The arrows that point from components of the graphical user interface to field variables and methods of the Seam components in the middle, indicate how the HTML and Java components are linked via Seam. There is, for instance, a TagQueryBean component that gets a query string injected from the main input box at the top of the HTML view. This component is, in turn, injected into all the other components by Seam. The arrows relating Seam components with the database and TagVector cache indicate which components use database queries and which consult Java objects that were loaded into a cache at server start up.

graphical representation of the resulting data is an important matter − one that leaves a lot of room for experimentation in future work.

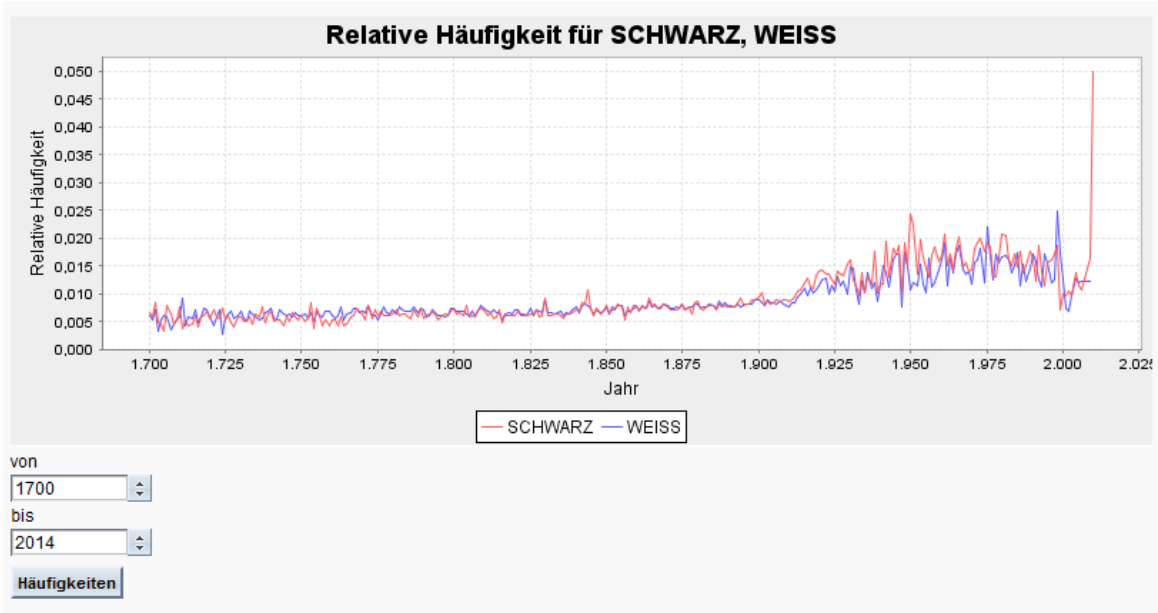
4.1 The Frequency Query Component

Preparation: To obtain frequency plots, the relatively vague dates that are available for ARTigo's images need to undergo rigid preprocessing. Typical examples for vague dates taken from the *artresource* table are "1420-1428", "um 1906-1911" or also "um 1496/97".[23] This vagueness can be due to a lack of information about the objects. However, it can also be owed to the nature of how pieces of art are created and introduced to a public audience. In contrast to the situation that Google faced with its Ngram Viewer, in arts, there does not always exist a distinct event of publication for a given artwork. Accordingly, many dates are estimates of art historians that are given as strings without a strict syntactic form. It is needless to say that this is not optimal for algorithmic analysis. Different preprocessing approaches are conceivable. One may decide to ignore artworks that aren't dated precisely, which would rule out the examples from above altogether, or one could ignore the imprecision and work with one of the boundaries or compute an intermediate date. In the database, there is, e.g., the PSQL function *normalizeDate* available that implements the approach where the left boundary of the estimation is taken to be the exact date.
Besides these preprocessing steps, precomputation is required. As mentioned in Section 2.2, the key to Google's NgramViewer and similar applications lies in precomputing every single n-gram that exists within their document collection. Regarding the Analytics Center, the situation is somewhat similar, only on a much more modest scale. In the theoretical section, Section 3.1, it was conceived that a tag frequency plot requires three different components from the database: identifiers of (valid) tags, dates of artworks, and frequencies that can be computed from the taggings that hold between tags and artworks. Unfortunately, these components are distributed in a relatively inconvenient way among three different tables, namely *tag*, *artresource* and *tagging*. Joining these tables is fairly expensive such that dynamically computing a plot when it is queried would keep the processor busy for an unacceptably long time.
Furthermore, it is, of course, important to treat German, English, or French frequency charts independently. Tests with different levels of confirmation (of German tags), namely 2, 20, 200, 2000 taggings, suggested that the confirmation has hardly an impact on the shape of the graph. This, at least, was the author's subjective impression which is rendered plausible by the fact that the top 574 tags with a validation level of over 2000 still account for about 62% of taggings.

GUI: Figure 4.1 displays a plot of the tags "SCHWARZ" and "WEISS", viz. black and white, within the tag frequency component.

In general, queries either specify single tags or a sequence of tags, separated by commas, the frequency graphs of which are then plotted together in one plot. This minimalistic approach to interpreting user queries makes it possible that one and the same query string

[23] Viz. around 1906-1911 etc.



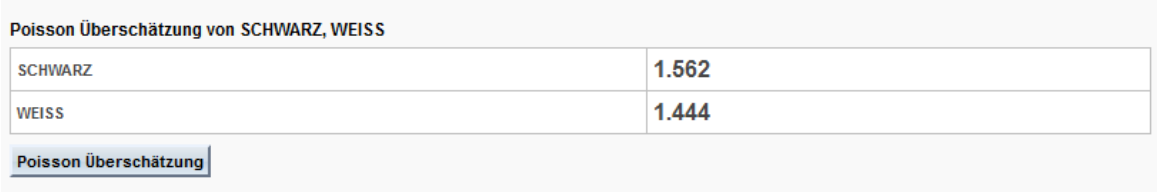
can be interpreted in each of the five components of the Analytics Center, as will become clear in the following sections. Also, giving the user the impression that more subtle queries along the lines of "Plot images that contain X and Y but not Z" are possible, would not be honest. Complex queries of this kind are precluded by precomputation. There are too many possibilities for such queries.[24] Note, however, that it is possible to combine frequency plots with association rule queries and compute an approximation of what such complex queries can be expected to yield. This will briefly be discussed in the future work Section 5.1.
Seam projects come with an inbuilt library for plotting, namely JFreeChart, which has proven to be a convenient choice, as it is a renowned plotting software that is free of charge and for which Seam specific examples can be found in [AR09]. Experiments with alternative tools such as Google's jmathplot and Xeiam's xChart also yielded beautiful results and could be used for interactive plots, if this should be desired.

4.2 The Poisson Overdispersion Query Component

Preparation: Just like in the frequency query component, the Poisson overdispersion of valid tags, introduced in Section 3.2, was precomputed and tags were counted as valid, if they have been used more than once. Choosing a low threshold for valid tags is important, as the Poisson overdispersion is related to a tag's specificity which, in turn, is high for infrequently used tags. Results for valid German tags have been stored in the table *poissonOverdispersion*.

[24] An image can be associated with any subset of valid tags. Queries of this kind had the power to pick out any subset of images. There are $2^{2^{|validTag|}}$ ways to pick out such subsets of images – even the number of digits of that number is several pages long.

GUI: Figure 4.2 shows the Poisson overdispersion query component. Queries are inter-

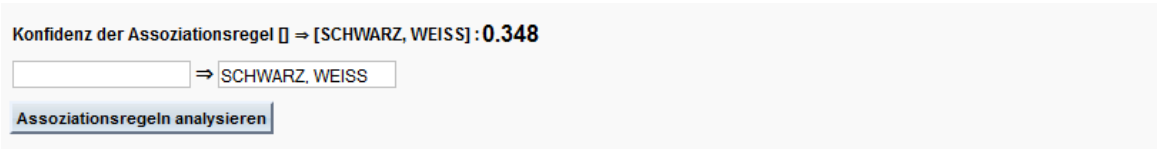


preted much like above, sequences of tags, separated by commas, can be queried at once. Their respective Poisson overdispersions are then displayed in a table. Hovering over the query button makes an explanatory text appear that instructs the user as to how the values are to be understood and gives examples.

4.3 The Association Rule Query Component

Preparation: Precomputation is not an option for association rules that were introduced in Section 3.3. Instead, the AssociationRuleComponent Java class creates sanitized SQL database queries dynamically. While this renders association rule queries significantly more complex than in the precomputed cases, two important factors keep the computational costs at bay. First, no table joins are required, as the necessary data completely lies in one important auxiliary table, that will be introduced in more detail below, namely *invertedIndex*. Secondly, association rules do not account for the number of times, a tag was associated with a resource, but only for whether or not it was associated with it. So basically, to compute the support of a set of tags, one counts the resources for which there exists at least one association with each of the tags in the set. While such "WHERE EXISTS(...)" conditions are expensive, they still belong to the standard repertoire of SQL operations and moderate query response times suggest that computational complexity is not really an issue in this component. That tagging counters aren't considered has another welcome side effect, as it sets association rules apart from all the other components of the Analytics Center. Specifically, a strong association between individual tags doesn't follow trivially from their cosine similarity, for instance. To be sure, these values can be expected to be highly correlated, but they are nevertheless genuinely distinctive from one another.

GUI: Figure 4.3 continues the running example for the GUI. Association rules require the



user to input the rules body and its head, both given by sets of tags. The only difference to

the previously introduced querying mechanisms is that this time, there are two input boxes available, corresponding to those two components. Sequences of tags are once again separated by commas and interpreted as sets by the AsociationRuleComponent. The default is to interpret the query string from the main input box, as the rule's head and interprets the body as the empty set. This yields the total frequency of images that have been tagged with each single tag in the specified sequence. As the output is simply a single numerical value, there is no need for a specialized view component. Here, too, a pop up window with an explanation of the component is available.

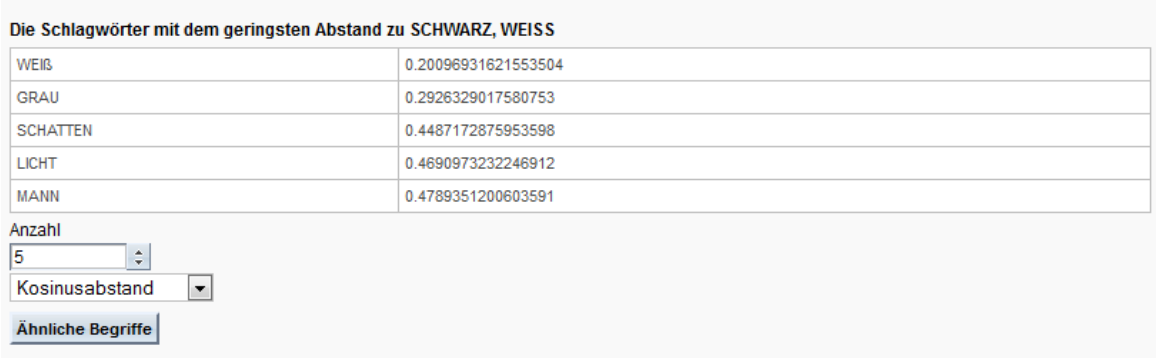
4.4 The Nearest Neighbor Ranking Component

Preparation: The nearest neighbor ranking component involves vector space computations that were introduced in Section 3.4. These are relatively costly. Moreover, precomputation is again not really an option.[25]
The following preparatory steps nevertheless allow for fast and scalable similarity computations: First, the already mentioned auxiliary table *invertedIndex* was created that maps valid tags with art resources and the corresponding number of *taggings* where it is greater than zero. This table exists in a large version in which tags with confirmation greater than 2 are stored and in a reduced version, where confirmation is significantly higher. At server start, Java *TagVector* instances are created from the reduced *invertedIndex* table and stored in a list that is managed by a so called *TagVectorFactory* Seam component. The *TagVector* class is designed with highly efficient distance computations and an option for the creation of virtual tags in mind. This is done with the help of a fast, array based way of retrieving tagging counters and by considering only the highly confirmed tags in the reduced *invertedIndex* table. The *TagVectorFactory* is used to construct *TagVector*s only when they aren't already available, thus, only when they do not belong to the highly confirmed tags. In this way, querying the nearest neighbors of a tag vector is pleasantly fast[26] if this is desired but also easily scalable in precision, as the confirmation threshold of the tags can be lowered.
The query string is once again parsed from the main input box. The question was, however, how sequences of multiple tags should be interpreted. Despite the efforts regarding computation speed, querying multiple nearest neighbor rankings would have consumed too much time. Also, having several rankings on one page was thought to decrease rather than increase readability.
The good news is, though, that there is a fairly natural solution to this problem: When multiple tags are queried, the average of their counting vectors, viz. the so called centroid, is computed. The ranking then displays the closest tags to this centroid that aren't contained in the query.[27] Like that, nearest neighbor queries for multiple tags yield concept that describe what the input tags have in common. The results are sometimes very appealing, as they seem to represent abstract notions that best subsume all of the queried tags. For

[25] Though, the situation is somewhat more moderate than, e.g., with association rules, as the size of a tag distance matrix is "only" quadratic in $|validTags|$. It is still bad enough, though, to make real-time computation very appealing.
[26] I.e. a matter of well under 10 seconds.
[27] The nearest neighbor of the centroid is called the medoid. It is returned only if it isn't contained in the query, though.

instance, a query of "Augen, Nase, Mund", viz. eyes, nose, and mouth, identifies "Gesicht", viz. face, as the concept that best subsumes all three.

GUI: Once again, Figure 4.4 illustrates the look of the component. Thanks to the rapid



distance computations, it is possible to allow users to set two additional parameters. First, they can choose the length of the ranking up to a maximum number of positions. Second, she can decide whether she wishes to use cosine distance or the more costly, but sometimes also more precise, Euclidean distance metric.
The results are displayed in a table. Possible visualizations of the vector space are certainly tempting, but hard to realize, due to the space's high dimensionality. Future work Section 5.2 addresses this issue briefly.

4.5 The DBScan Cluster Expansion Component

Preparation: The final component, DBScan cluster expansion, relies on the same data structures and techniques as the similarity ranking component. The computationally most expensive part of the algorithm, Algorithm 1 in Section 3.5,[28] is the $rangeQuery(t, \varepsilon)$ call that returns the set of tags that lie within ε range of tag t. Calling this method is roughly as expensive as computing the similarity ranking of t and it is called for each tag in the cluster. This makes DBScan clustering a relatively expensive component of the Analytics Center. However, the computation costs are justified by the algorithm's unique way of uncovering links and associations between different motives in arts. Also, the waiting time can be bounded by limiting the maximum number of tags that can be contained in a cluster.

GUI: Figure 4.5 gives an example. The query works precisely as in the nearest neighbors component. Thus, querying multiple tags results in a cluster expansion around their centroid. The algorithm comes with two parameters, $minPts$ and ε, that together determine

[28] A Java implementation of the algorithm is available at http://elki.dbs.ifi.lmu.de/

how densely populated one wants the cluster to be. Both of these parameters can be set by the user. It is also possible to have the user adjust the maximum number of tags in the cluster up to a certain value.
visualization is again fairly basic, as the result set is simply displayed to the user as an HTML table. However, here too one could easily imagine more involved modes of visualization, e.g., through dendograms in which the substructure within a cluster becomes apparent.

5 Future Work

This section outlines possible expansions of the Analytics Center. Section 5.1 describes a simple feature to approximate more specific frequency plots. Interestingly, this can be achieved by combining existing components of the Analytics Center. Section 5.2 describes how the high dimensional vector spaces could be visualized through PCA or archetype analysis. Section 5.3 concludes this work by pointing out ways in which user generated analyses could themselves become part of an ARTigo GWAP.

5.1 Using the Analytics Center for Complex Frequency Queries

Situation: It has been argued in Section 4.1 that frequency charts are available only for individual tags, among other things, because computing them dynamically would incur unreasonable costs. It has been pointed out to me at several occasions, though, that being able to query combinations of tags or to enforce the absence of other tags would be a valuable addition to the Analytics Center.

Goal: While querying such complex frequency charts directly remains impracticable, it is not just possible, but relatively easy to approximate them. That is, one could devise a query language for the frequency plot component which allows for conjunctions of tags and their negations along the lines of what is presently available in ARTigo's simple search. Thus, for instance, "MANN HUT -KRAWATTE" would plot the frequencies of images tagged with "MANN" (man) and 'HUT" (hat) but not with "KRAWATTE" (tie).
Also, recall Section 2.2 where the NgramViewer's[29] approach to a similar task has been discussed. Google enriched its query language with arithmetic operators that could also be implemented in the Analytics Center.

Approach: Recall that association rules are basically conditional probabilities for taggings, Section 3.3. They are computed with respect to the entire corpus of artworks, thus, the association rule component does not reflect how these probabilities have changed over time. However, when one assumes that the resulting inaccuracy is acceptable, "logical conjunctions" of tags $t_1, \ldots, t_n$ in the sense described above become feasible, as $p(t_1 \wedge \cdots \wedge t_n) = p(t_1|t_2 \wedge \cdots \wedge t_n) \cdot p(t_2 \wedge \cdots \wedge t_n) = p(t_1|t_2 \wedge \cdots \wedge t_n) \cdot p(t_2|t_3 \wedge \cdots \wedge t_n) \cdot \cdots \cdot p(t_n)$. As association rules do not cover "negations", one had to figure out whether it is best to change the workings of association rules or formulate the above equation in a way that omits explicitly negated tags. Furthermore, the error that springs from this heuristic approach remains to be assessed.

5.2 Visualization of the Tag Vector Space

Situation: The vector spaces are very high dimensional and, therefore, cannot be visualized graphically at present. However, as the Analytics Center shall be accessible for a public audience, an intuitive visual representation of the data is certainly worth while.

[29] https://books.google.com/ngrams

Goal: In [BT09] Bauckhage and Thurau introduced a promising approach for the visualization of large vector spaces, namely archetypal analysis. Figure 5 gives an example for the visualization of images in a tag vector space. On the border of the image lie the archetypes. Intuitively, each picture of the dataset − most of them represented by red dots in the middle − can be approximately reconstructed by "mixing together" the archetype images in a suitable way. The advantages over alternatives like principal component analysis (PCA) are that archetypes are salient members of the dataset so that it is somewhat intuitive what it might mean, if other data points are represented by them.

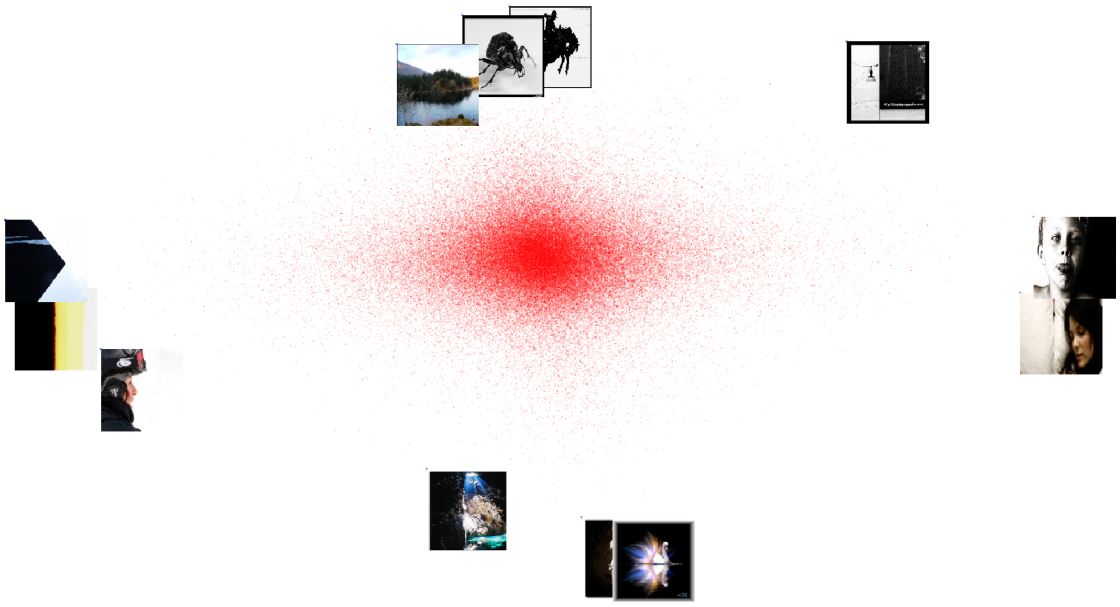


Figure 5: Representation of 50000 images by proximity to archetypes on convex hull. Image source [BT09].

Approach: In the Analytics Center, data points in the vector space are tags rather than images. Accordingly, the methods in [BT09] had to be applied to the artwork level inverted index of ARTigo's dataset, that was introduced in Section 3.4. The set of tags that result from nearest neighbor or clustering queries could then be highlighted on a two dimensional map of the vector space, much like the one depicted in Figure 5, but with tags as archetypes on the outer border.

5.3 Deriving GWAPs from the Analytics Center

Situation: Users can explore ARTigo's database and engage in private inquiries. However, it is not made explicit, what the common end to these inquiries might be. Evaluating whether a finding is interesting, is left entirely to the user and her reward consists of the

discovered knowledge in and of itself. Meta-data about the user generated queries and discoveries is lost.
However, some of the tasks that can be solved with the help of the Analytics Center lie at the heart of notoriously difficult problems. Angiulli et al., [AIP01], argue, for instance, that association rule mining, viz. the problem of finding all association rules with a certain minimum confidence, is NP-complete.

Goal: The Analytics Center's components are quite suitable to be turned into games. Peaks and ditches in frequency plots could, for example, be commented with a suitable kind of meta-tag. Maybe the frequency drop of cubism[30] in 1914 could be related to World War I – and if enough users think this explanation is plausible, it could be accepted as valid. The discovery of a tag that owes its high Poisson overdispersion to text contained in the image could be rewarded.[31] The finding of interesting association rules with high confidence would be very valuable, as has been indicated above. And the same certainly goes for the discovery of particularly dense neighborhoods or of concepts that relate distant tags through dense clusters. Also, finding all clusters for a given density parameterization is a computationally costly endeavor in which the help of ARTigo users would be appreciated.

Approach: Apart from finding suitable games based on the Analytics Center's components, one had to devise ways for evaluating user findings and for rewarding them in a motivating manner. Regarding the latter, one suggestion is to have users accumulate points while their session with the Analytics Center is active. These points along with meta-data on her findings might then be saved on the user's ARTigo account, if she was logged in during her Analytics Center session. Highscores and incentives could be provided much like on the ARTigo platform.

5.4 A Pinboard for Data Mining Projects

Situation: The Analytics Center doesn't provide users with a way to share their findings and hypotheses with other users. As already noted in Section 5.3, the community driven data mining process is not yet guided in any way, but left entirely to the user's individual curiosity and interest. Some users might find it difficult to think of interesting hypotheses, but easy to understand how to apply the Anayltics Center's tools in a meaningful way. Others may be able to devise concrete hypotheses but be unsure how to test them, lack the time to do it, or want a group of students to engage in the inquiry.

Goal: Users of ARTigo and the Analytics Center could be provided with a forum where they can form groups, create projects, post hypotheses, and present and discuss data from the Analytics Center.

[30] Recall Figure 3.1.
[31] This aspect was discussed in Section 3.2.

Approach: Users had to be able to create projects with different privacy levels. A project's initiator would be able to describe her hypothesis as a starting point for the discussion. The description could be phrased in prosaic forms, accompanied by images and data from the Analytics Center. Participants would then be free to present related findings to their group members. To this end, data from the Analytics Center had to be exportable, e.g., in pdf-format. Also, functionalities to persist and share private sessions and an option to start group sessions could be added to the Analytics Center.

References

[AIP01] Fabrizio Angiulli, Giovambattista Ianni, and Luigi Palopoli. On the complexity of mining association rules. In *SEBD*, pages 177–184, 2001.

[AR09] Dan Allen and Norman Richards. *Seam in action*. Manning, 2009.

[BT09] Christian Bauckhage and Christian Thurau. Making archetypal analysis practical. In *Pattern Recognition*, pages 272–281. Springer, 2009.

[Cod70] Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI MAGAZINE*, pages 37–54, 1996.

[Lev13] Elena Levushkina. Computerlinguistische methoden in community-basierten anwendungen. PhD Thesis, Institute for Linguistics, University of Munich, 2013, 2013.

[LM06] Amy N. Langville and Carl D. Meyer. Information retrieval and web search. *Supported by National Science Foundation under NSF grant CCR-0318575, USA*, 2006.

[LvA11] Edith Law and Luis von Ahn. *Human Computation*. Morgan & Claypool, 2011.

[MSA+11] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, 2011.

[QB11] Alexander J. Quinn and Benjamin B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1403–1412. ACM, 2011.

[Sin01] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[Ste11] Bartholomäus Steinmayr. Designing image labeling games for more informative tags, 2011. Diploma Thesis, University of Munich, 2011.

[WBBL13] Christoph Wieser, François Bry, Alexandre Bérard, and Richard Lagrange. Artigo: Building an artwork search enginewith games and higher-order latent semantic analysis. 2013.

[Wie14] Christoph Wieser. *Building a Semantic Search Engine with Games and Crowdsourcing*. Dissertation/Ph.D. thesis, Institute of Computer Science, LMU, Munich, 2014. PhD Thesis, Institute for Informatics, University of Munich, 2014.

[Zim14] Arthur Zimek. 2014.