LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

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

Institut für Informatik

Explaining Outliers in ARTigo

Alexander Fischer-Brandies

Bachelorarbeit

Beginn der Arbeit:	27.6.2016
Abgabe der Arbeit:	11.11.2016
Betreuer:	Prof. Dr. François Bry
	Dr. Clemens Schefels
	Dr. Arthur Zimek
	Dr. Erich Schubert

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 11.11.2016 Alexander Fischer-Brandies

Abstract

The ARTigo project is based on a social image tagging game developed at the Ludwig Maximilian University of Munich. It provides us with a large database of images described by tags which were chosen by the players of the ARTigo game.
The *outlying properties detection problem* (*ODP*) defines the problem of finding explanations for the most outlying properties of a certain object.
This work intends to provide a practically working solution to the ODP problem within the context of the ARTigo database.

Acknowledgements

I want to thank Dr. Arthur Zimek and Dr. Erich Schubert who inspired me to work on the subject of data mining during their excellent introductory seminar on this topic. Also their support throughout this work was of great value to me.

I likewise want to thank Dr. Clemens Schefels for supervising me during the whole process of creating this work.

Contents

1 Introduction

The ARTigo Project[ART16] is a collaboration of several institutes of the Ludwig-Maximilian University of Munich involving contributors from the Institute of Informatics and the Institute of Art History among others. It provides an online game with the aim to supply images of artworks with tags. The player has one minute for describing an artwork with tags he can freely choose. He gets points for tags that have been used by other players as well. Thus a winning strategy consists in describing each artwork as precise as possible. Those tags are then saved in a database and offer a great foundation for research utilizing data mining techniques. This work is aimed at detecting the most unusual characteristics shown by a specific piece of artwork (if they exist at all!). As an introductory example, assume you are interested in discovering the most unusual and unexpected properties of an object which is described by the tags *BLUE, RED, CUBISM, BRAQUE, PICASSO, WOMAN, HAT* and *TREE*. Now the question is which of those tags explains the outlierness of this object with the highest plausibility. For instance one could argue that *TREE* is a tag which is used very rarely within the context of the whole dataset (because there are not many images with trees...). Thus *TREE* might be an excellent attribute to distinguish our object from most of the other images. In a similar way it could be said that *BLUE* is a color which descibes many of the objects and the occurrence of this tag is not very unexpected at all. More interesting explanations can be given if we combine multiple tags. Assume that both *BRAQUE* and *PICASSO* are both relatively common attributes with the context of our database. Nevertheless the combination of both tags can be rare or even unique. Then the combination of *BRAQUE* and *PICASSO* would provide a much more accurate description of the outlying properties than the single attributes would provide. In order to be able to compare the quality of different sets of attributes regarding their explanatory value a criterion that measures its outlying properties is needed. In this work I am especially examining the measure proposed by [AFP09]. Having found an applicable measure the second challenge is that the whole search space of combinations of attributes is exponentially big and thus cannot be explored completely by a naive algorithm. A dataset with n attributes leaves us with a whole of $2^n - 1$ combinations to be considered. I tackle this problem by using an adaption of the algorithm introduced by [AFP09].
This work is organized as follows. Section 2 gives an overview of related work. Section 3 provides an overview of the ARTigo project, the ARTigo game and the ARTigo Analytics Center. Furthermore it describes how the data obtained from the ARTigo database looks like. Section 4 is related to the detection of outlying images in ARTigo. Section 5 covers the main topic of this work which is the detection of the most outlying characteristics of a certain image. Section 6 mentions the most helpful tools I utilized in this work while section 7 summarizes the results and provides a prospect of potential future work.

2 Related Work

GWAP (*game with a purpose*) or *human-based computation game* are terms used to label online-games using human capabilities in order to reach a certain goal. The basic idea behind this concept is that human beings are much better capable of fulfilling certain tasks than machines. For instance the ARTigo Image Tagging Project utizes the players ability of describing pieces of artwork - which is a task computers cannot fulfill in a satisfying way yet. Thus the ARTigo Project can be characterized as *GWAP*. A well-known work about the concepts of human computation is [LvA11].
The ARTigo Project provides us with a database of tags with which our objects of interest have been described. This setup shows many characteristics of text mining which can be considered a sub-discipline in data mining. It is easy to imagine that not all words used to describe a certain object might have the same relevance. For example colors like GREEN or BLUE are used very frequently to describe a lot of images. Thus the occurence of those tags in a specific object might be relatively less relevant. In order to judge which tags are more significant different measures of word importance can be applied. The simplest choice is the raw *term frequency* (TF), which is the number of times that a tag appears in an object. However there exist various kinds of normalization that can be used instead. Another important concept in judging word importance is the *inverse document frequency* (IDF). IDF measures whether a tag is common or rare across all objects from our database. It is the logarithmically scaled inverse fraction of the objects that contain a certain tag, obtained by dividing the total number of objects by the number of objects containing the tag, and then taking the logarithm of that quotient.
$IDF(t, DB) = log\frac{N}{1+|\{o \in DB:\ t \in o\}|}$

- N: number of objects in DB: $N = |DB|$
- $|\{o \in DB :\ t \in o\}|$: number of objects o, where tag t appears. To avoid division by zero in case tag t does not appear in object o 1 is added to that term.

There exist several common variants of IDF as well.
TF and IDF can be combined to TF-IDF in the following way:
$TF\text{-}IDF(t, o, DB) = TF(t, o) \cdot IDF(t, DB)$
A good outline of TF-IDF and related topics can be found in [LRU14]. Another work that is concerned with similar methods in text mining - including the cosine distance which is applied in this work - is [Sin01]. [BG05] is covering the challenging topic of *multidimensional scaling* (*MDS*) in great detail. It's basic idea is to allow for better visualisation by projecting a high-dimensional numerical space onto a lower-dimensional one while preserving the distances between its objects as well as possible.

2.1 Outlier Detection

Detecting outliers is a well-known task in statistics. Here usually one has to model a set of data points using a statistical distribution. Points that do not fit well with the postulated model are then considered to be outliers. An overview of many statiscal methods used for finding outliers is given by [BBL78]. Practically the biggest challenge when applying statistical methods is to find an appropriate statistical distribution. Also in our case we do not have any knowledge about possible underlying data distributions.
The concept of distance based outliers was introduced by [KN98]. They a giving the following both simple and intuitive definition: *"A point p in a dataset is an outlier with respect to parameters k and d if no more than k points in the dataset are at a distance of d or less from p"*. Compared to statistical methods this definition has the advantage that it does not require any knowledge about statistical distributions in advance. The disadvantage of this definition is that it requires the user to specify the distance d which can be defficult to determine. Furthermore it does not provide a ranking for the outliers.
An efficient and relatively standard algorithm for mining distance based outliers from large datasets that solves both problems mentioned above is formulated by [RRS00]. The basic idea is to calculate the distance of a point from its k^{th} nearest neighbor. Each point is then ranked on the basis of that distance and the top n points in this ranking are declared to be outliers. In general every metric such as the "Manhattan" or "Euclidean" metrics can be applied for measuring the distance between two points. However a metric distance function is not strictly necessary and depending on the application area a nonmetric distance function can be used, too. This makes the algorithm applicable to a wide range of application domains. However there is no obvious measure available to calculate the distance between two objects from the ARTigo database. The reason is that the ARTigo database only provides us only with categorical data and thus the methods provided by [RRS00] are not directly applicable.
The concept of *local outliers* was introduced in [BKNS00]. The *LOF* (*local outlier factor*) algorithm proposed by the authors identifies outliers with respect to the densities of their neighborhoods. The basic idea is to compare the density of each object o with the density of the k nearest neighbors of o. Objects with a low density compared to their local neighborhood are then considered to be outliers.
A formalized method that allows for theoretical comparison and generalization of local outlier detection methods is given by [SZK14]. By abstracting the notion of locality from the distance-based notion, the framework given by the authors facilitates the construction of abstract methods for many special data types that are usually handled with special algorithms. Those methods might also be applied to the problem of detecting and explaining outliers in the ARTigo dataset and the methods applied in this work.
In [WQZ+03] the authors present a definition of outliers based on a hypergraph model which is suitable for categorical data. The authors design an algorithm called Hypergraphbased-OutlierTest (HOT) which is able to mine such outliers. Its aim is to discover a portion of the dataset in which the value assumed by some objects on a single additional attribute is infrequent with respect to the mean frequency of the values in the domain of that attribute. The objects with the biggest deviations on that part of the database are then considered to be outliers.

2.2 Outlier Explanation

The above mentioned work only considers the identification of outliers. [KN99] provides a first attempt to also find explanations of why an identified outlier is exceptional. This might be useful to evaluate the correctness and validity of the identified outliers. More importantly additional knowledge about the data can be gained by knowing why a given object might be special. The problem is, as with some of the outlier detection methods considered above, that the proposed methods can only be used with numerical data.
A piece of work which appeared to be exceptionally useful for the task of explaining the outlierness of objects from the ARTigo database is [AFP09]. The authors focus solely on categorical data. Given a data population characterised by a number of attributes the task is to single out sets of attributes that best explain the abnormality of a certain object. In order to do so the authors present a novel criterion that measures the abnormality of sets of attributes featured by a given abnormal individual with respect to the reference population. Also a linear space algorithm for computing the top outlying properties (according to that measure) is presented. Furthermore the authors also concern themselves with different notions of locality and the task of detecting subspaces of the dataset in which valuable outlier explainations can be discovered.
[AFP13] is a paper by the same authors which extends the findings from [AFP09] by introducing the concept of *groups* or *subpopulations* of exceptional individuals. This concept might also be applied to the ARTigo dataset, although it is not considered in this work.
A third paper by the same authors which utilizes the same measure of outlierness is [AFPM13]. Numerical attributes are widely used and important in databases. Thus the authors analyze the outlying property detection problem within a new context that also takes numerical attributes into account.
Another very recent work focussing on the numerical domain and dealing with the same problem of detecting outlying properties is [VCR$^+$16]. The paper covers both critical questions: *"How to design effective scoring functions that are unbiased with respect to dimensionality and yet being computationally efficient? How to efficiently search through the exponentially large search space of all possible subspaces?"*

3 About ARTigo

The ARTigo Image Tagging Project[ART16] is a collaboration of several institutes of the Ludwig-Maximilian University of Munich. Its image portfolio currently contains more than 200 000 images. Among others the contributors to the project involve the Institute of Informatics and the Institute of Art History. The project's website [ART16] explains the ARTigo game in the following way: *ARTigo is an online game with the aim to supply artworks with tags. The same artwork is simultaneously shown to you and a co-player. The game now consists in describing the artwork accurately with tags. A tag can describe what you see on the artwork or remarks on style, quality or emotions. For each artwork, you have 60 seconds for your input. You get points for the tags identical to the ones entered by your co-player or another player in a previous round. The more tags per artwork are matched, the more points you get [...]"*. The project also created a couple of variants of the original ARTigo game.
The entered tags are then saved in a database upon which this work performs its research. Built upon the tags provided by the players the ARTigo Project also provides the user with a search engine. The user can search for specific tags as well as for certain additional information provided with each image. This includes search for the artist, title, location and the year or a range of years in which the object was created.
The way the ARTigo project makes use of the information provided by its users shows a textbook example of *human computation*. In fact the whole project was inspired by a discussion about *games with a purpose* ($GWAP$) according to the information on the project's website.
Besides offering the foundation for the creation of a search-engine for artworks the database of the ARTigo Project also allows for various scientific research including the application of statistical and data-mining techniques. The ARTigo Analytics Center is a web platform that allows users to analyze the data provided by the ARTigo database with analytical methods provided by the platform. A motivation for creating this application is to enable users to perform research on the ARTigo data without too much effort. The tools provided currently include frequency plots, poisson dispersion, association rules, neighborhood analysis and clustering methods. The methods for explaining outliers provided by this work have also been added to the platform's services. Currently the ARTigo Analytics Center is still in the development stage and has not been released to the public yet.

Let us now consider how the data received from the ARTigo project that is used throughout this work looks like. Figure 1 shows a table of the data after it has undergone a few preparation and normalization steps. Those tasks had already been performed and evaluated by [Ngu16] in his work on clusters in ARTigo. I am using the same dataset throughout this work in order to achieve consistency and allow for comparison among the results.
Nevertheless I want to touch on the steps that have been applied. Firstly only tags appearing at least 4 times with an image are considered. This step helped to sort out typographical errors and tags only a few players decided were suitable to describe an image. This leaves us with a total of approximately 38000 images. The highest number of tags that is used to describe a single image is 84 and overall there are 8616 different tags.
In a second step the $TF\text{-}IDF$ has been calculated for each image and tag. The motivation

38151	3_winterhalter_1217.jpg	BÄUME	2.8952194603844323	BLAU	0.728436460278974	BLUMEN
38152	3_winterhalter_1607.jpg	ARMREIF	5.2436682540856605	DAME	2.7938986293007178	FÄCHER
38153	3_winterhalter_1942-8.jpg	ARM	0.5272995484885196	ÄRMEL	3.3374924705727516	AUGEN
38154	3_winterhalter_1976-18.jpg	ARM	0.31367562884445266	ARME	0.3343345052686029	AUGE
38155	3_winterhalter_2461.jpg	BART	2.8125012797572237	BRILLE	7.907079615594493	BUCH
38156	3_winterhalter_523.jpg	BAUM	1.5267428100310225	BÄUME	0.772058522769182	BLÄTTER
38157	3_wocher_PK_I_501-74.jpg	ANTIKE	0.6339999759669787	AQUARELL	1.659408823984327	ARM
38158	3_woensam_I%20%202876.jpg	DRUCK	1.3379879676251358	FAHNE	2.473515583330146	FEDER
38159	3_wolf_2630.jpg	ALPEN	1.9704641223375234	BACH	1.8889120720662025	BAUM
38160	3_wouwerman_2715.jpg	ANGRIFF	0.896487944813934	BERG	0.46748043103899706	BRAND
38161	3_wtewael_2730.jpg	BAUM	0.9395340369421677	BÄUME	0.8908367570413639	FRAU
38162	3_zeitblom_44.jpg	ALTAR	5.091968249902455	FLÜGELALTAR	2.4187241203664445	GEMÄLDE
38163	3_zick_1975-31.jpg	AUDIENZ	3.4472636535463876	BLAU	1.2361345992612893	DIENER
38164	3_zick_2499.jpg	ALT	0.45724730805665353	APOSTEL	1.415340352719275	AUFGESCHLAGEN
38165	3_ziem_1983-15.jpg	AQUARELL	1.5556957724853067	BAUERN	3.6213413236894665	BAUM

38166 rows × 170 columns

Figure 1: Sample from the ARTigo database: tag counts normalized by $TF\text{-}IDF$

behind this is to put a weight on the relevance of each tag. An explanation of $TF\text{-}IDF$ is provided in Section 2.

4 Outlier Detection in ARTigo

"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" - definition of outliers due to Hawkins (1980)
Outlier detection is a large field within data mining. It's purpose is to discover objects with unexpected characteristics in the context of a reference dataset. In the context of ARTigo an outlier is an image that was tagged in a way that does not fit well with the way the other images in the database were described by the players of the ARTigo game. This might be because the image's tags do only appear in the database very rarely. For instance an image showing a very uncommon motive in the context of the whole database will thus often be described by uncommon tags, too. Such an image would then be considered to be an outlier. Another kind of outlier is an image that was described by tags that are not uncommon by themselves, but make up a very uncommon combination. This is the case if those tags do usually not appear together within the tagging of one and the same image.
Taking up this idea it can be stated that an outlier is an image with a tagging that is very different from the tagging of most of the other images. This leads to the question of how the similarity or the distance of a pair of images can be measured. Considering that there are approximately 8000 different tags the set of all images can well be interpreted as points in an 8000-dimensional Euclidean space. For tags that have not been used to describe an image the value is zero. Otherwise the value represents the number of times a tag has been used with an image. This means that each image is zero in most of the dimensions and only takes on different numerical values in the dimensions that represent the image's tagging.
A common way of measuring distances between pairs of vectors in high-dimensional positive spaces is the cosine distance. It is determined by the cosine of the angle between two vectors. The cosine of two vectors a and b is determined by the inner product:
$\langle a, b\rangle = \|a\| \cdot \|b\| \cdot cos(\alpha)$
So the cosine distance of two vectors a and b with angle α is:
$cos(\alpha) = \frac{a \cdot b}{\|a\| \cdot \|b\|} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \cdot \sum_{i=1}^{n} (b_i)^2} \in [0, 1]$
If two vectors are very similar and point in the same direction their cosine distance is 1. If the cosine distance is 0 it means that the vectors are orthogonal.
The cosine distance is a measure often used in text mining. More information on its application methods can be found among others in [Sin01].
In order to now visualize the similarities between different images *multidimensional scaling* (*MDS*) can be used. This effectively is a transformation into a lower-dimensional linear space which aims at preserving the distances between objects as well as possible. This lower-dimensional space can then be used for visualisation purposes and also standard outlier detection methods can be applied to it.
Detailed information on MDS can be found in [BG05]. In my work I used the implementation of MDS by [SKE+15]. That implementation is $O(n^2)$ and uses $O(n^2)$ memory according to its author Dr. Erich Schubert. With the ARTigo dataset it performed the transformation (using cosine distance as distance measure) into a 6-dimensional Euclidean space using less than 13 GB of memory with a Intel(R) Core(TM) i7-6500U @ 2.59 GHz CPU in less than 10 minutes. Figure 2 visualizes the ARTigo data after it has been transformed into a 3-dimensional Euclidean space by MDS. It shows that most images are located in relatively

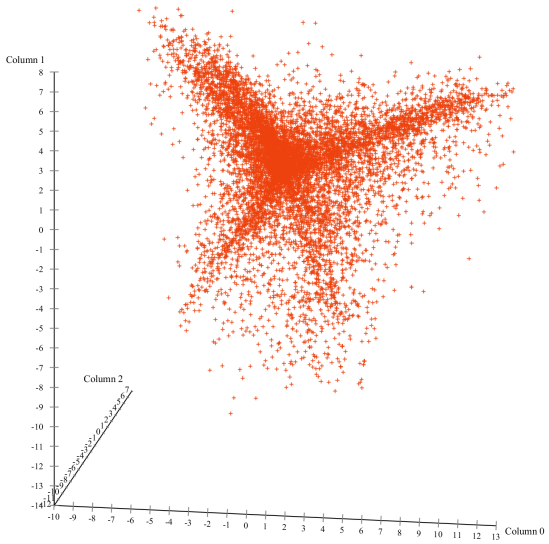
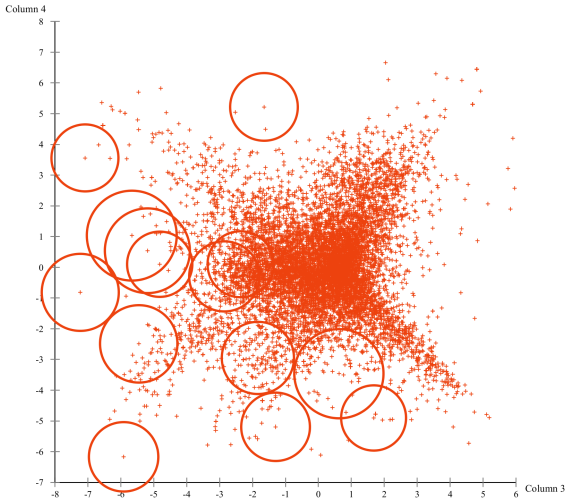


Figure 2: Visualization of the ARTigo data after transformation into a 3-dimensional Euclidean space by multidimensional scaling (MDS)

dense areas. Also some points show the characteristic of not having many neighbors within their close distances and might already be considered outliers.
In order to find those outliers algorithmically the standard kNN (k^{th} nearest neighbor) outlier detection method proposed by [RRS00] can be used. This algorithm ranks each object based on the distance to its k^{th} nearest neighbor. The objects with the biggest kNN-score will then be considered to be outliers and should match with the objects we already expected to be outliers by looking at the visualization graph. The only input parameters that need to be specified with the kNN-algorithm is k and a proper distance function. Since we are now dealing with a n-dimensional Euclidean space the most logical and simple choice is the Euclidean distance:
$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$, with a,b being vectors from our n-dimensional Euclidean space.

The kNN-algorithm has also been implemented by the ELKI framework [SKE+15]. Figure 3a shows a 2-dimensional graph of the 6-dimensional Euclidean space with the top outliers beeing marked by circles. The top outliers were determined by the kNN-algorithm with $k = 4$. Different kNN-scores are represented by different magnitudes of the circles' radii. Figure 3b shows the top 16 outliers and their kNN-scores. Figure 4 shows the distribution of different outlier-scores. It can be seen that most of the images have an outlier-score between 0 and 1. Those images cannot be considered to be outliers. A score bigger than 2 occurs only with the top 500 outliers while only the top 50 outliers have a score bigger than 3 and only 4 images are rated with an outlier-score bigger than 4.
Now that the top outliers have been detected let us go on and consider the question of how the most unexpected properties of those outliers can be determined.



(a) The 2-dimensional graph of the 6-dimensional projection; top outliers marked by circles with radii representing their outlier-scores

	ID	score
0	5_B14289.jpg	4.795839
1	5_B21078.jpg	4.476357
2	5_B1918.jpg	4.420566
3	5_B32060.jpg	4.122180
4	5_B34249.jpg	3.868162
5	5_B18550.jpg	3.868162
6	5_B24391.jpg	3.813173
7	5_B42664.jpg	3.737243
8	3_baldung_90.jpg	3.710085
9	5_B33950.jpg	3.693754
10	5_B70247.jpg	3.682131
11	5_B7917.jpg	3.674139
12	5_B26779.jpg	3.664672
13	5_B8827.jpg	3.636734
14	1_7876.jpg	3.618606
15	1_29157.jpg	3.572430

(b) Table of the top 16 outliers and their outlier-scores

Figure 3: Results of the kNN-algorithm

5 Explaining Outliers in ARTigo

A different but quite closely related topic is the detection of oulying properties for a given object. It is important to note that such properties may or may not exist. Another important difference compared to outlier detection is that the outlier object is now part of the input parameters of an algorithm instead of the output. Let us first formalize what exactly we want to achieve by adopting the definition of the *Outlying Property Detection Problem* (*OPD*) introduced by[AFP09] and adjusting it to our problem. I want to mention that other than [AFP09] I do not consider the task of detecting the top outlying properties in arbitrary subspaces (also refered to as *local outlying properties detection*). In other words, the definition of the *ODP* problem used throughout this work only refers to a subdomain

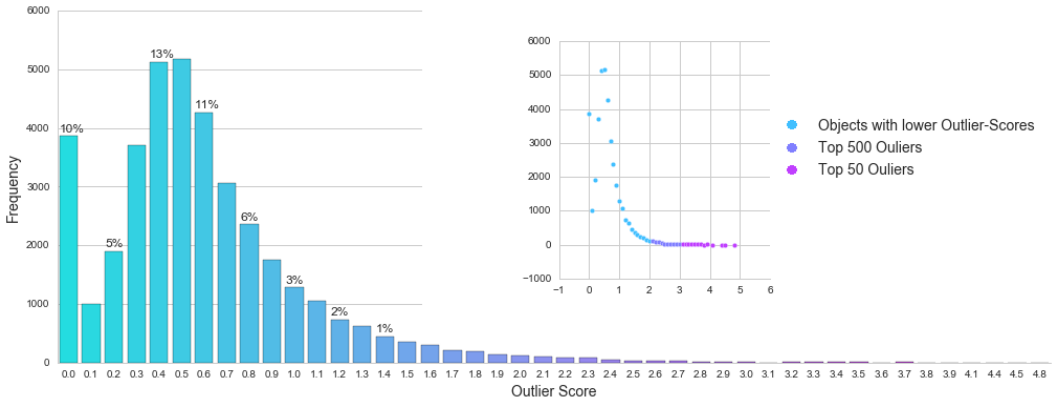


Figure 4: Distribution of outlier-scores obtained by the kNN-algorithm

of the original problem discussed by the authors. Nevertheless the theoretical insights from the more general work remain valid in the more specific case examined in this work.

Definition Given a dataset DB on the set of tags T, an object o of DB, a threshold $\theta \in [0, 1]$ and a positive integer k, the problem **Outlying Property Detection (OPD)** is defined as follows:
Among all the subsets S of the set of tags T of DB, find $S_1, ..., S_k$, $out_{Si}(o) \geqq \theta$, also called the *top-k outlying properties*, scoring the largest outlierness value $out_S(o)$.

As mentioned above there can exist less than k solutions to the problem. So the solution set of the *OPD* problem includes at most k solutions.

5.1 Measuring the outlierness of an object

How can the outlierness value $out_S(o)$ be calculated and what does it mean? In Order to define the outlierness measure of an object we need a number of preliminary definitions first. Note that i am using the term "tag" instead of "attribute" in all the definitions, because it better fits with the domain of the ARTigo dataset. In the original definitions given by [AFP09] the term "attribute" is used, which I will also use when referring to the *outlying properties detection* (*OPD*) problem in general without the context of explaining outliers in ARTigo.

Definition Let DB be a database on the set of tags T, o an object of DB and $S = t_1, ..., t_k$ a subset of the tags in T. The object then represents the tuple $(v_1, ..., v_m)$ with $v_i \in T$. The projection $o[S]$ of object o on the subset of tags S is the new object $(o[t_1], ..., o[t_k])$. The projection $DB[S]$ of database DB on the set of tags S is the new database $\{o[S] \mid o \in DB\}$. The selection $DB_{o[S]}$ of the database DB is the new database $\{o^{'}[S] \in o[S]\}$. The frequency $freq_s(o)$ of o in the database DB, $S \subseteq T$, is the rational number $\frac{|DB_{o[S]}|}{|DB|}$.

Definition The **frequency histogram** $hist_S$ of DB with respect to S is the ordered multiset of the rational numbers $\{f_1, ...f_n\}$, with f_i being the frequency $freq_s(o_i)$, $o_i \in DB[S]$, and $f_{i-1} \leq f_i$ for $i \in \{2, ...n\}$.

(Definition) Let h be a frequency histogram. The **cumulated frequency histogram** $|h|$ of h is the set $\{(0,0),(f_j,g_j),(1,1)\}$ with f_j being a distinct frequency of h and $g_j = \sum_{h_j \leq f_j} h_i$.

Definition Let h be a frequency histogram. The **marginal frequency histogram** $\|hist_S\|$ of h is the set of pairs (f_j, F_j), where (f_j, g_j) is a pair of the cumulated frequency histogram $|h|$ and F_j is $\sum_{f_k > f_j}((f_k - f_{k-1}) \cdot g_{k-1}$

	ANGEL	BLUE	CHURCH	CUPOLA	TREE
1	False	False	False	False	True
2	False	True	False	False	True
3	False	False	True	False	True
4	False	True	True	True	True
5 (outlier o)	True	True	False	False	True
6	False	True	False	False	False
7	False	False	True	False	False
8	False	False	True	True	False
9	True	True	True	False	False
10	True	True	False	False	False
11	True	False	False	False	False

(a) Example database $DB_{example}$

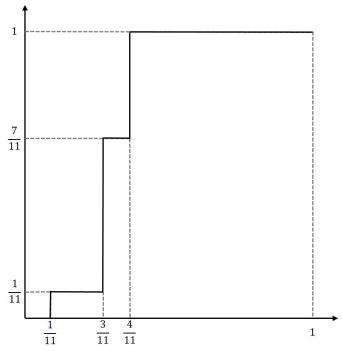
	Tuple	Freq
1	(ANGEL_True, TREE_True)	1/11
2	(ANGEL_True, TREE_False)	3/11
3	(ANGEL_False, TREE_False)	3/11
4	(ANGEL_False, TREE_True)	4/11

(b) Objects in $DB_{example}$ and their frequencies

Figure 5: Example database

j	f_j	g_j
1	0	0
2	1/11	1/11
3	3/11	7/11
4	4/11	11/11
5	1	1

(a) cumulated histogram



(b) curve of the cumulated histogram

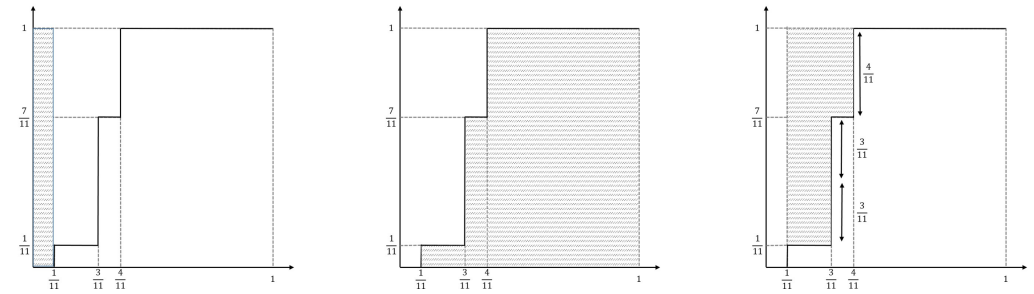
j	f_j	F_j
1	0	86/121
2	1/11	86/121
3	3/11	84/121
4	4/11	7/11
5	1	0

(c) marginal histogram

Figure 6: Histograms of the example database

Before defining the outlierness measure we are going to use later let us look at an example that shows how the given definitions are used. Figure 5a shows an example database $DB_{example}$. A value of *"True"* means that a tag was used to describe a certain image. Figure 5b shows the frequencies associated to $DB_{example}$ of the different combinations of the values of the 'ANGEL' and 'TREE' tags in decreasing order. $freq_{\{ANGEL,\ TREE\}}(o)$ is only $\frac{1}{11}$

which should intuitively provide for a relatively good explanation for the outlierness of object o. The frequency histogram $hist_S^{DB_{example}}$ is the ordered multiset $\{\frac{1}{11}, \frac{3}{11}, \frac{3}{11}, \frac{4}{11}\}$ which can be seen in table in figure 5b.
The same example is continued with figure 6a which shows the cumulated frequency histogram $|h|$ associated with $DB_{example}$ and $S = \{ANGEL, TREE\}$. The marginal frequency histogram is given in figure 6c. Finally figure 6b shows the curve of the histograms which can now be used to determine the desired outlierness value.



(a) Area associated with $freq_S(o)$ (b) Area associated with $\|hist\|(freq_S(o))$ (c) Area associated with $out_S(o)$

Figure 7: Different areas associated with the curve of the cumulated frequency histogram

Definition The **outlierness** value $out_S(o)$ of the set of tags T in o with regards to dataset DB is given by:
$out_S(o) = 1 - \Big(freq_S(o) + \|hist_S\|(freq_S(o))\Big) \in [0, 1]$

The outlierness takes values between 0 and 1. If it is 0 the outlying property is considered to be usual. An outlierness value very close to 1 means that the tags are a good explanation for an objects abnormality. The closer $out_S(o)$ gets to 1 the more the property by a set of tags is outlying.
Let us continue out example and see how $out_S(o)$ can be visualized. Figure 7a shows the area above the curve of the histogram associated with $freq_S(o)$. It's value is $\frac{1}{11}$. The area representing the term $\|hist_S\|(freq_S(o)$ is highlighted in figure 7b. Its value is $\frac{1}{11} \cdot (\frac{3}{11} - \frac{1}{11}) + \frac{7}{11} \cdot (\frac{4}{11} - \frac{3}{11}) + 1 \cdot (1 - \frac{4}{11}) = \frac{86}{121}$. Figure 7c shows the area of $out_S(o)$ which is the area in-between the other two areas that have already been determined. The value $out_S(o)$ is then equal to $1 - (\frac{1}{11} + \frac{86}{121}) \approx 0.2$.
This definition of $out_S(o)$ has some both useful and intuitive properties:

- If $hist_S = \{\frac{c}{n}, ..., \frac{c}{n}\}$, then $out_S(o) = 1 - \Big((\frac{c}{n} + (1 - \frac{c}{n})\Big) = 0$. That means that in case of a uniform distribution of the tags $t_1, .F..t_n \in S$ the outlierness is 0.
- If $freq_S(o) = max(hist_S(o)) =: m$, then $out_s(o) = 1 - \Big(m + (1 - m)\Big) = 0$. So, if o assumes the value which occurs with the highest frequency in DB then $out_s(o)$ will also be 0. This is rather intuitive, too. A set of tags cannot justify an objects outlierness if the object has the most usual values with those tags.

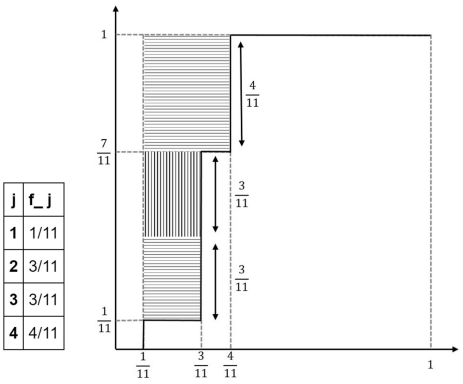


Figure 8: Example of outlierness computation by vertically adding up the highlighted areas

- The maximal value for $out_S(o)$ is reached if $hist_S = \{\frac{1}{n}, \frac{n-1}{n}\}$ and $freq_S(o) = \frac{1}{n}$. In that case $out_S(o) = 1 - \left(\frac{1}{n} + \|hist_S\|(\frac{1}{n})\right) = 1 - \left(\frac{1}{n} + \frac{1}{n} \cdot (\frac{n-1}{n} - \frac{1}{n}) + 1 \cdot (1 - \frac{n-1}{n})\right) = \frac{(n-1)(n-2)}{n^2}$. The bigger the database DB is, the closer $out_S(o)$ will get to 1: $\lim_{n\to\infty} out_S(o) = 1$. Intuitively this means that the best explanation is provided if object o assumes one value on the set of tags S and all the other objects assume one same but different value.

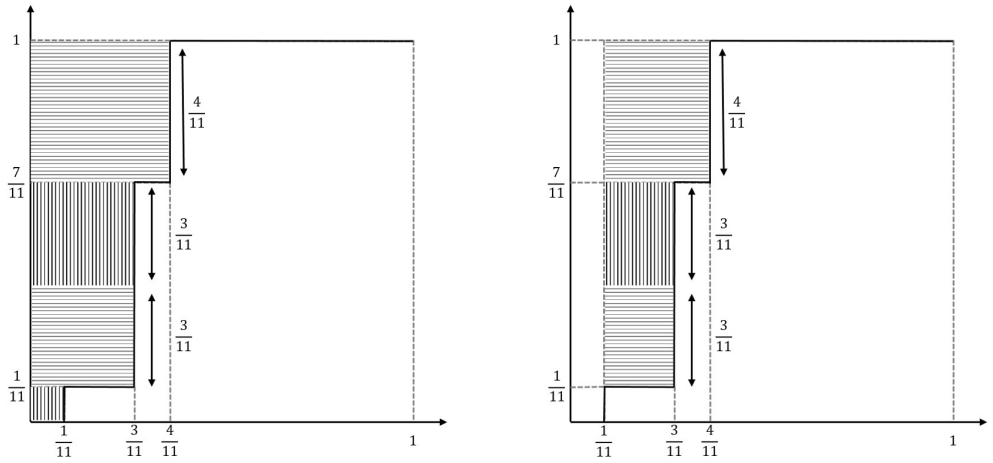
In order to effectively compute $out_S(o)$ there is an alternative way to determine the area that represents the outlierness of an object: The rectangles highlighted in Figure 8 which also represent $out_S(o)$ can be summed up directly. In the example $out_S(o)$ is composed of three rectangles. Thus $out_S(o) = \frac{3}{11} \cdot (\frac{3}{11} - \frac{1}{11}) + \frac{3}{11} \cdot (\frac{3}{11} - \frac{1}{11}) + \frac{4}{11} \cdot (\frac{4}{11} - \frac{1}{11}) \approx 0.2$.
In [AFP09] the authors justify this method of determining an object's outlierness by the following theorem:

(Theorem) Let o be an object with $freq_S(o) = f$ and $hist_S = \{f_1, ..., f_k\}$. Then $out_S(o) = \sum_{f_i > f}(f_i \cdot (f_i - f))$.
This way $out_S(o)$ can be calculated efficiently with the values from the histogram (of course the histogram still has to be calculated).

In [AFP09] the authors also point out that their definition of outlierness is closely related to the well-known *Gini index* (also *Gini coefficient*). The Gini index G measures the *heterogeneity* or *inequality* of a frequency distribution and is defined as:

Definition: Gini index $G = 1 - \sum_{i=1}^{n}(f_i)^2$ with f_i representing the frequencies from $hist_S$.

Figure 9a now shows the area associated with the term $\overline{G} = 1 - G = \sum_{i=1}^{n}(f_i)^2$ while figure 9b shows the area associated with $out_S(o)$. The interesting relationship between those two measure gets quantified by [AFP09] in the following way: *"$\overline{G}$ corresponds with the value of outlierness associated with any object not occurring in the database. Thus, the smaller the object frequency f is, the closer the outlierness value gets to $\overline{G}$".*



(a) Area associated with $\overline{G}$ (b) Area associated with $out_S(o)$

Figure 9: Relationship between the Gini index and $out_S(o)$

Since the frequencies occurring in the ARTigo dataset are usually very small (because most tags do not occur very frequently throughout the whole database) it is likely that using the Gini index instead of $out_S(o)$ would provide us with similar results. Nevertheless the outlierness measure introduced by [AFP09] seems to be a bit more convenient to me since it considers both the frequency of an object o and the heterogeneity of the frequencies of those objects that are more frequent than o.

5.2 The FindOutlyingProperties algorithm

Now that we have defined a measure of an object's outlying properties let us come back to the *OPD* problem. The first important thing to notice is that the search space consisting of all subsets of the set of tags T is exponentially large. Since $|DB| = 8616$ the search space contains 2^{8616} properties. This means that an effective pruning strategy is essential in order to compute the desired solution within reasonable time boundaries.
The next important aspect is that $out_S(o)$ is not monotone with respect to property inclusion.
Given two sets of tags S and S' with $S \subset S'$, it cannot be concluded whether $out_S(o)$ is smaller, larger or the same as $out_{S'}(o)$.
Fortunately an upper bound to the outlierness of any superset of S can be calculated. This is stated by the following theorem:

(Theorem) Let $h = hist_S^{DB}$ be the frequency histogram of DB, S a subset of tags, f the frequency of object o in DB with respect to S and n the number of objects in DB. Then
$out_S(o) \leq \left(\sum_i (f_i)^2\right) - \frac{(2f+1)n-2}{n^2} =: oub_S(o)$.

This theorem implies that the outlierness of object o is maximized with regards to any

superset S' of S, if S' preserves $hist_S(o)$, with the exception of making o different from any other object. In [AFP09] the authors provide a proof of the theorem which I am not going to repeat here since it is quite lengthy and knowledge of the correctness is enough to utilize its practical implications.
Let us now consider how the basic findOutlyingProperties algorithm which solves the *OPD* problem looks like.

```
1  def findOutlyingProperties(DB, T, o, θ, S, T_k):
2      out = outlierness(DB, S, o)
3      if out ≥ θ:
4          updateTopOuliers(T_k, S, out)
5      θ* = minOutlierness(T_k)
6      upperBound = outliernessUB(DB, S, o)
7      if upperBound > max(θ*, θ):
8          while T != []:
9              t = T[0]
10             T = T − {t}
11             findOutlyingProperties(DB, T, o, θ, S ∪ {t}, T_k)
```

The findOutlyingProperties algorithm

The findOutlyingProperties algorithm computes the top k outlying properties.
The input parameters for the algorithm are:

- DB: the input database
- T: the set of tags
- o: the object for which the top k outlying properties are determined
- θ: the outlierness threshold representing the minimal outlierness value we are interested in
- S: the subset $S \subseteq T$; it is initialised to [ ] and tags are added to it by the recursive calls
- T_k: the top k outlying properties which represent the solution of the *OPD* problem

Let us examine how the algorithm works. With the first call T_k is set to []. Then the outlierness $out_S(o)$ is calculated. If $out_S(o)$ is larger or equal to the outlierness threshold θ and if $out_S(o)$ is larger than the smallest value in T_k then S is added to T_k and the smallest value is removed from T_k. Otherwise T_k is not changed. In the case that T_k contains less than k items S is added to T_k if $out_S(o) \geq T_k$ and no items are removed from T_k.
The smallest value of T_k is returned by the minOutlierness(T_k) function. If T_k contains less than k items minOutlierness returns -1.
The outliernessUB(DB, S, o) function exploits the theorem for calculating an upper bound to the outlierness value for any superset of S. If upperBound is larger than the maximum of θ and θ^* then the findOutlyingProperties algorithm is called recursively with more items from T added to S. Otherwise the search space is pruned at that point.
I want to mention that it is important to notice that calling the $findOutlyingProperties(DB, T, o, \theta, S \cup \{t\}, T_k)$ function shows call-by-value-like behavior with respect to parameter T.

Practically the algorithm will only work well if the pruning process works well.

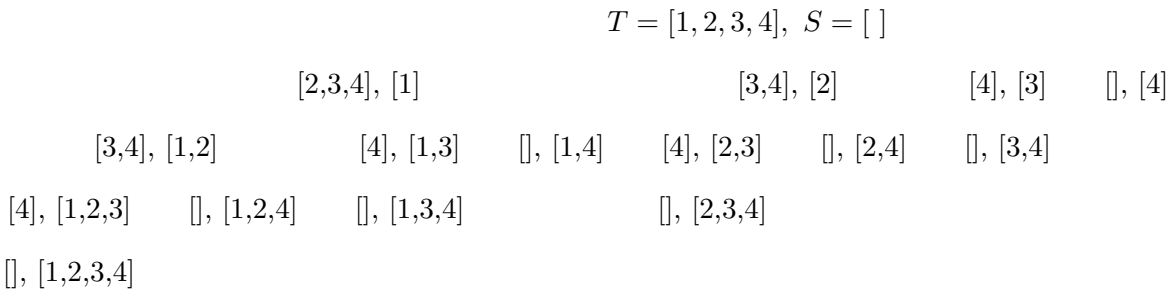


Figure 10: Example of algorithm execution for $T = [1, 2, 3, 4]$ without pruning.

Figure 10 shows an example of how the findOutlyingProperties algorithm would go through the complete search space for $T = [1, 2, 3, 4]$ without pruning. It requires $2^{|T|} = 2^4$ steps to cover the complete search space with all subsets of T. The algorithm works completely depth first. If the upper bounds for S were $\leq max(\theta, \theta^*)$ the tree would be cut at that point.
Note that the earlier the pruning happens, the more effective it will be. Also pruning on the left side of the algorithm's execution tree will be more effective since the tree is not symmetrical. Another idea to improve the pruning process is to grow θ^* as quick as possible. I will further discuss these topics in section 5.3.

5.3 Experiments

This section covers the insights I gained from implementing the findOutlyingProperties algorithm for the ARTigo dataset.
A rather trivial observation is that most tags are not used to describe most images. Therefore the most frequent value for a single tag is *"False"*. This implies that $out_{\{t\}}(o)$ is zero if $o[\{t\}]$ is *"False"* (which means that t was not used to describe image o). Also $out_{\{t_1,...t_k\}}(o)$ is zero if the projection $o[\{t_1, ..., t_k\}] = (o[t_1], ..., o[t_k])$ equals (*"False",..., "False")*. In other words, a set of tags which do not describe an image does not provide an explanation for the image's outlierness.
A more qualitative observation is that the outlierness scores are often rather high (that means close to 1) if they are not zero. To quantify this a bit I can say that outlier scores larger than 0.9 are not uncommon for a single tag. That is because the frequencies of most tags are relatively small. This is especially true for single tags and if outlying properties within subspaces are not considered. However, the comparison between multiple high outlierness scores is well possible, even if they are close to 1.
Let us now come to the two main problems I encountered when trying to apply the findOutlyingProperties algorithm to the ARTigo dataset. Both problems are related to each other and I was not able to find a completely satisfying solution to both of them.

The overfitting problem

Given an image o a good strategy in finding a set of tags S with a high outlierness score $out_S(o)$ is the following: One starts with a tag t_1 with $o[\{t_1\}] = "True"$ with a relatively small frequency $freq_{\{t_1\}}(o)$. Because of its small frequency $\{t_1\}$ will already have a good outlierness score $out_{\{t_1\}}(o)$. This score can often be improved in the following way:
Let $\{c_1, ...c_k\}$ be the maximal set of objects with $c_1[\{t_1\}] = ... = c_k[\{t_1\}] = "True"$. Now find a tag t_2 with $o[\{t_2\}] = "False"$ and $c_i[\{t_2\}] = "True"$ for some $c_i \in \{c_1, ..., c_k\}$, such that $out_{\{t_1,t_2\}}(o) > out_{\{t_1\}}(o)$.
Note that such t_2 does not necessarily exist, but the experiments have shown that often it does.
This step can then be repeated: Find t_3 with $o[\{t_3\}] = "False"$ and $c_i[\{t_3\}] = "True"$ for some c_i that have not been "eliminated" in the previous step, such that $out_{\{t_1,t_2,t_3\}}(o) > out_{\{t_1,t_2\}}(o)$.
Finally we get a large set of tags $S = \{t_1, ..., t_l\}$ with $o[\{t_1, ..., t_l\}] = ("True", "False", ..."False")$ which has a really high outlierness score.
The problem is that this kind of solution is of relatively little value for the user and solutions of this kind can hardly be interpreted in a meaningful way.
This effect of the solution being too much adjusted to the exact problem can be described as overfitting. Thus I will refer to it as the *overfitting problem.*

The upper bounds problem

Another unpleasent effect happens when the upper bound for a set of tags, which each do not belong to an image's tagging, is calculated.
So let $o[\{t_1\}]$ be *"False"*. As mentioned above the outlierness score $out_{\{t_1\}}(o)$ will be zero then. Nevertheless t_1 can still be used to justify the outlierness of image o. This could for instance be the case in a scenario similar to the one mentioned in relation with the *overfitting problem.* The tag t_1 could then be used to distinguish object o from other objects with a similar tagging.
This idea is expressed by the observation that tags with the characteristic of t_1 usually have really high upper bounds. The same holds true for $o[\{t_1, ..., t_k\}] = (o[t_1], ..., o[t_k]) =$ *("False",..., "False").*
The resulting problem is that the values returned by the outliernessUB() function are so large, that the whole pruning process does not work anymore for all tags t_i with $o[\{t_i\}] = "False"$. As a result the findOutlyingProperties algorithm does not terminate within any reasonable time limits.
In a less formal way one could describe this problem by concluding that there are so many tags t_i with $o[\{t_i\}] = "False"$, which potentially could all justify an object's outlierness, that it is not possible to calculate the outlierness scores of all of them.

Avoiding the overfitting and upper bounds problem

As mentioned above I was not able to solve the *overfitting* and *upper bounds* problem satisfyingly. Nevertheless it is possible to avoid both problems by not taking tags into account that are not part of an image's tagging. This is a rather big concession. It means that the implications drawn from the output of the findOutlyingProperties algorithm will be restricted to something along the lines of:
"The outlierness of object o is best explained by the fact, that it was described by the set of tags $\{t_1, ..., t_k\}$*."*
Conclusions like the following one will no longer be possible:
"The most outlying property of object o is, that it was tagged with $\{t_1, ..., t_l\}$*, but not with* $\{t_{l+1}, ..., t_k\}$*."*
A benefit of taking only tags with $o[\{t\}] =$ *"True"* into account is that the size of the search space is considerably reduced. The other and most important benefit is, that the pruning process now works reasonably well.

Parametrization and further improvements of the algorithm

Let us have a closer look at how the pruning process works and how it can be improved. Given a set of tags S the idea is to first calculate the upperBound to the outlierness value of all supersets of S. Then we check if upperBound is larger than the maximum of $\{\theta, \theta^*\}$. In the first iterations of the algorithm θ^* will be set to -1. Only after we have detected a number of k outlying properties with an outlierness value larger than threshold θ the maximum between θ and θ^* will be set to θ^*. This means that the pruning process will be more effective in the later iterations of the algorithm. A method to speed up the growing process of θ^* is to consider the outlying properties of the most promising tags first. These are the tags with the lowest frequencies regarding their overall appearance in the database. This means that we should presort the set of tags T in increasing order of their frequencies before calling the findOutlyingProperties algorithm. In [AFP09] the authors propose a different way of sorting the attributes by reordering them after each iteration based on the previous results. In my opinion this option should generally be prefered, but in our case the simple preordering should always provide the best chance of growing θ^* as quickly as possible.
Another way to improve the pruning process is to consider whether a potential improvement of the outlierness value is *significant*. For example - given an object o, a set of tags S, an outlierness value $out_S(o) = 0.95$ and an upper bound of 0.9505 to the outlierness of any superset of S - we could state that a *significant* improvement of the result within that branch of the search space is not longer possible and prune all supersets of S. This idea can be implemented by subtracting a certain value δ from $max(\theta, \theta^*)$ before deciding whether to prune or not. Continuing the previous example, for $\delta = 0,001$, the term $max(\theta, \theta^*) - \delta = 0.9505 - 0.001$ would be evaluated to 0.945. This would imply that the potential improvement is not large enough and since $0.945 < 0.95$ the algorithm would not continue. A similar possibility is to check if the upper bound is significant enough on a percentage basis instead of demanding an absolute improvement.
A somewhat drastic method to avoid getting solutions that contain too many tags is to define a maximal number of iterations that the algorithm can go through. However this did

not proof to be necessary. Firstly the top outlying properties returned by the findOutlying-Properties algorithm mostly include sets with only a few tags. For instance among the top 20 outlying properties it is quite rare to find solutions with more than 4 tags. Secondly this method also does not improve the performance of the algorithm in a noticeable way which also means that the pruning process works well from a certain point on.
Let us now briefly consider the runtime of the algorithm which is closely related to the values chosen for the parameters k and θ.
An obvious observation is that the smaller k and the larger θ is, the faster the algorithm will finish. Nevertheless I think that both parameters will usually be chosen accordingly to the user's interest.
My experiments showed that by implementing the methods mentioned above and choosing $k \lesssim 20$ the algorithm always finished in less than 20 seconds using a Intel(R) Core(TM) i7-6500U @ 2.59 GHz CPU, regardless of how θ is chosen.
It is always possible to significantly speed up the runtime by choosing θ slightly lower than the minimum of the outlierness values of the top k outlying properties. However this is not convenient at all since we do not know it advance how large the resulting outlierness values will be. After some testing I think that choosing $\theta \approx 0.5$ and $k \approx 20$ is a reasonable default setting to find the most interesting solutions relatively quickly. I have to admit though that I am sure that the performance of the algorithm can be significantly improved by providing a better implementation than I did. I will come back to this topic in section 7.

Local outlying properties

The task of detecting outlying properties within a (*local*) context specified by the user is simple. Given a database *DB,* an object o and a set of tags $\{t_1, ..., t_k\}$ the subspace associated to $\{t_1, ..., t_k\}$ consists of the objects c_i with $o[\{t_1\}] = c_i[\{t_1\}]$, ...,o[$\{t_k\}$] = $c_i[\{t_k\}]$. Finding all objects that fulfill this property can be done by dropping all objects from DB that don't fulfill the desired tagging. Afterwards the findOutlyingProperties algorithm is going to detect the top k outlying properties which is even easier now due to the reduced size of DB.
The task of finding local outlying properties in arbitrary subspaces is also considered in [AFP09], but I did not do not cover that topic in this work.

6 Tools

- *ELKI* (*Environment for Developing KDD-Applications Supported by Index-Structures*) [SKE+15] is an open source data mining software written in Java. It is developed at the Ludwig Maximilian University of Munich. *ELKI* includes many data mining algorithms and a capable visualization module. The framework proved to be of great value for this work - especially during the whole process of outlier detection.
- The Python programming language in combination with the NumPy, pandas and matplotlib libraries provided a very user-friendly and valuable tool for implementing and testing the outlier detecting methods I considered in this work.

7 Conclusion and Future Work

[AFP09] introduces an interesting new measure for the ability of a set of attributes to explain the outlierness of a certain object.
In [AFP09] the authors also propose an algorithm that solves the *OPD* problem. It turned out that the algorithm has the major constraint of practically not being able to cover the complete search space of the ARTigo dataset.
If the search space is reduced to sets of tags that were used to describe a certain image, then the algorithm solves the *OPD* problem within reasonable time limits.
Regarding the performance of the algorithm, its parametrization is not a substantial factor which means that default parameters can be used.
A logical way to improve upon the results of this work is to provide a better implementation. This should include a more sophisticated representation of the data and a better utilization of index structures. Also using a more performance-oriented programming language than python might be reasonable.
Also I do belief that a different approach might be able to overcome the limitations of the approach I considered in this work. A major improvement would be to find ways to cover the complete search space of all tags occurring in the database. Another preferable option would be to consider the number of times the tags were used, instead of only considering whether they were used with an image or not. This essentially would be a transition from categorical to numerical data.
Another interesting topic that was not considered in this work is the detection of local outlying properties which could also be done in the context of the ARTigo dataset.

References

[AFP09] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Detecting outlying properties of exceptional objects. *ACM Trans. Database Syst.*, 34(1), 2009.

[AFP13] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Discovering characterizations of the behavior of anomalous subpopulations. *IEEE Trans. Knowl. Data Eng.*, 25(6):1280–1292, 2013.

[AFPM13] Fabrizio Angiulli, Fabio Fassetti, Luigi Palopoli, and Giuseppe Manco. Outlying property detection with numerical attributes. *CoRR*, abs/1306.3558, 2013.

[ART16] ARTigo, 2016. [Online;].

[BBL78] Vic Barnett, Vic Barnett, and Toby Lewis. Outliers in statistical data. Technical report, 1978.

[BG05] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer series in statistics. Springer, 2005.

[BKNS00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 93–104, 2000.

[KN98] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 392–403, 1998.

[KN99] Edwin M. Knorr and Raymond T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 211–222, 1999.

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press, 2014.

[LvA11] Edith Law and Luis von Ahn. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[Ngu16] Tien Duc Nguyen. Explorative cluster analysis or artworks for the ARTigo Analytics Center. Bachelorthesis, Ludwig Maximilian University Munich, 2016.

[RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 427–438, 2000.

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

[SKE+15] Erich Schubert, Alexander Koos, Tobias Emrich, Andreas Züfle, Klaus Arthur Schmid, and Arthur Zimek. A framework for clustering uncertain data. *PVLDB*,

8(12):1976–1979, 2015.

[SZK14] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Min. Knowl. Discov.*, 28(1):190–237, 2014.

[VCR+16] Nguyen Xuan Vinh, Jeffrey Chan, Simone Romano, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. Discovering outlying aspects in large datasets. *Data Min. Knowl. Discov.*, 30(6):1520–1555, 2016.

[WQZ+03] Li Wei, Weining Qian, Aoying Zhou, Wen Jin, and Jeffrey Xu Yu. HOT: hypergraph-based outlier test for categorical data. In *Advances in Knowledge Discovery and Data Mining, 7th Pacific-Asia Conference, PAKDD 2003, Seoul, Korea, April 30 - May 2, 2003, Proceedings*, pages 399–410, 2003.