Masaryk University
Faculty of Informatics

Complex Data Analysis by Mining and Searching Operations

Ph.D. Thesis Proposal

Jakub Peschel

Advisor: Prof. Ing. Pavel Zezula, CSc.

Brno, Fall 2019

Signature of Thesis Advisor

Contents

1 Introduction **1**

Introduction **1**
1.1 *Problem description* 2

2 State of the Art **5**
2.1 *Type Independent Analysis* 5
2.1.1 Pattern Mining 5
2.1.2 Similarity Based Analysis 13
2.2 *Analytical Systems and Libraries* 16
2.2.1 SPMF 17
2.2.2 ELKI 18
2.2.3 Weka 19

3 Aims of the Thesis **21**
3.1 *Research Plan* 23

4 Achieved Results **25**
4.1 *ADAMiSS* 25
4.1.1 Storage 25
4.1.2 Transaction File 26
4.1.3 Analytical Processing Unit 26
4.2 *Prototype of ADAMiSS* 28

5 Author's Publications **31**

Bibliography **33**

A Publications **39**

1 Introduction

In recent years, data and metadata are created in orders of petabytes or even exabytes. Every online system generates data, human activity is recorded, and even interactions between fully automated systems are captured. Some examples of data are as follows:

Financial transactions: Every day, billions of items are bought and sold. Most of these financial transactions are captured on multiple levels for purposes such as tax collection, economic growth trends or solvency estimation.

Social networks: Thanks to the enormous popularity of social networks, an immense amount of interaction between people is created on social networks every day. Not only messages sent through the social network and statuses posted on public space but other forms of interactions like emoji's, hashtags and likes are also created.

World Wide Web: The World Wide Web structure contain enormous amounts of web pages that are interlinked and contain a huge amount of information itself. The amount of pages registered to this day is in orders of hundreds of billions and growing.

Network interactions: In these days, not only human interactions exist. Machines and automated systems communicate between themselves. Traffic lights, automated assembly lines, elevators, air conditioners, and many others create a lot of valuable information about its working.

Real-life interactions: People do not only communicate through the social networks, but other means of communication and interaction are also captured in the form of telephone calls, images from public events, video recordings, police reports and many others.

The huge amounts of data are a direct result of advancements in technology and a decline in their creation can not be expected. The data records our everyday behaviour, and thus one can extract valuable

insights. Data mining serves as a tool to get these insights and to learn valuable information from data.

The data from different sources come in a lot of different forms. Raw, unstructured or sometimes semi-structured data contain all the information but are challenging to process. To address this problem, experts transform data into more suitable forms and create pipelines of processing that transforms data into a more usable form.

Often during this transformation, some parts of data are omitted. As an example, textual data transformed into a bag of words are stripped of their ordering; voice frequencies are transformed into text transcripts, images are transformed into vectors of numbers.

Process of mining is a process of iterative application of different mining techniques, that uncover part of the hidden information. Multiple techniques are often used to create a final analysis of data. Different techniques require a specific form of data, and thus a transformation of data is done multiple times. With each consecutive transformation, the probability of a loss of information grows.

This work focuses on two areas of data mining, similarity-based analysis and pattern mining. These techniques allow to work with high range of data formats and do not restrict its usage to some specific cases. The similarity-based analysis spans over a task like clustering, similarity searching and outlier detection. All of these tasks use the similarity of objects to obtain information. Pattern mining is a problem of finding a relationship between data objects in a collection of data. The main role in the process has a frequency and co-occurrence of the data objects.

1.1 Problem description

As drafted in the introduction, data mining is an iterative process where multiple techniques are applied to data. Tools providing these techniques often require a very varying type of input. This requirement has a negative impact on two aspects. Firstly, expertly trained for data mining task require to know a high range of tools and their respective input format. Such a person has to be highly specialised and trained for many years to provide results. However, many of the techniques

have fixed input and output, and the transformation can be done automatically in between.

The second problem is the growing risk of loss of information when numerous types of transformation are introduced into the process of analysis – especially when transformations are performed on-demand and human factor plays a high role in the process. The created error can then lead to completely misleading results.

Another aspect explored in this proposal is the potential of combinations of analytical techniques such as pattern mining and similarity-based analytical techniques. Best to our knowledge, a combination of multiple techniques from these areas are not researched, and primary focus is drawn to the development of the elemental block like clustering and pattern mining.

2 State of the Art

In this chapter, different techniques and systems are described. The main goal is to give an overview of existing approaches to complex analytical techniques. As it is shown, lot of described systems are very usable but lack the properties to be well suited for the task of combining universal analytical blocs into complex analytical tools without a high level of tweaking and transformation of data.

In the first section 2.1, data mining tasks relevant to this thesis are explained, and respective techniques are described. Section 2.2 contain a description of libraries containing a high range of possible techniques relevant to this thesis proposal and their possible drawbacks in regard to the chosen objective.

2.1 Type Independent Analysis

There exist a high number of analytical techniques; some of them strictly require a specific form of data. Although these techniques are very relevant for such type of analysis, they are not very universally applicable. Often transformation into the required form is not possible or is very time-consuming.

There also exist analytical techniques, that does not force specific form and are suitable for high range of types. These are call type independent analytical techniques in this thesis because they can be applied universally to heterogeneous data. Example of these techniques is pattern mining and similarity-based analysis. The only restriction introduced by these techniques is that data objects must be distinguishable.

2.1.1 Pattern Mining

Pattern mining is an area of data mining that search for patterns in data. The pattern is a structure inside the collection of data that is frequent and can be thus distinguished from random behaviour. There exist several areas of a pattern mining based on a structure of searched patterns. Here is a list of several areas:

- Frequent item-set mining

- Sequence mining
- Graph mining
- Association-rule mining

The problem of pattern mining was stated by Aggarwal et al. [1] as:

Definition 1. *Given a database $\mathcal{D}$ with transactions $T_1, \ldots, T_n$, determine all patterns P that are present in at least a fraction s of the transactions.*

where s is minimal support provided by a user. This definition can be seen in each variant of a pattern mining although other conditions can be added.

In classical pattern mining, support framework is often used. Support is a term for a frequency of occurrence in database and strategies to efficiently compute support is one of the possible optimizations of pattern mining algorithms.

Frequent Item-set Mining

Pattern mining was first introduced in the form of *frequent item-set mining* as a possible approach to the problem of market basket analysis. The problem consisted of an analysis of the items bought together in markets. Due to the enormous amount of baskets and items, the brute force approach suffers from combinatorial explosion. Problem was approached by the usage of the Apriori algorithm proposed by Agrawal et al. [2] in 1995.

In frequent item-set mining, transactions contain data in the form of sets. A pattern is a common subset occurring in these transactions. To select a common pattern, downward closure property is used.

Definition 2 (Downward closure property)**.** *Every subset of frequent item-set is also frequent.* [3]

This property is very convenient from an algorithmic point of view because it allows for a more concise representation of patterns and allows for an iterative approach.

In frequent item-set mining, there are three main families of algorithms:

- Apriori based
- Pattern growth based
- Vertical mining based

Apriori based algorithms work on the principle of eliminating infrequent subsets early on during evaluation of their frequency and thus reducing search space considerably. Example of the reduction can be seen in figure 2.2.

These algorithms follow a simple two steps process:

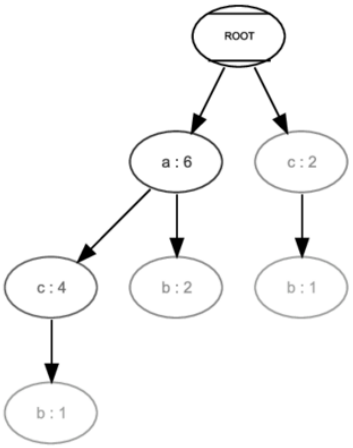
Generation step: During this step, candidate patterns of size k are generated from frequent patterns of size $k-1$.

Elimination step: In this step, support of each generated candidate pattern from the generation step is computed, and infrequent candidate patterns are removed from the result.

Pattern growth algorithms opposed to Apriori like algorithms do not generate candidate patterns, but instead, they build longer patterns from smaller ones. The most notable algorithm from this family is called FP-Growth[4] that transforms the initial database into a more concise structure called FP-Tree. The structure was developed as an effort to:

- group original data elements for mining and then analysing relevant data sets instead of generated candidate sets
- minimise scanning of entire database for the whole set of candidate patterns
- reduce the amount of needed main memory.

The main contribution to the idea of the FP-tree can be seen in TreeProjection method[5] and FP-Growth algorithm. TreeProjection allowed for the better ordering of the analysed items, based on the frequency of items and order of their processing based on their frequency. FP-Growth algorithm projects database into FP-tree structure, which better fits into memory, and allows to generate projected FP-trees recursively. Example of simple FP-tree can be seen in figure 2.1.



a, b, c, d
a, c
a, b, d
a, c
c,
b, c, d
a, b
a, c

Figure 2.1: Sample FP-tree generated from small database (right side)

Pattern growth family of algorithms includes FP-Growth, PPV[6], PrePost[7], FIN[8] and NegFIN[9]. These algorithms propose more optimised variants of FP-tree structure with modified node structure for better traversing of the tree.

The last approach, mentioned in this thesis proposal, is based on a vertical representation of the transaction database. In this approach, the database is transformed into an inverted list called tidList or tidSet. This form allows for better counting of the support of patterns. Support is computed by the intersection of lists of individual items.

As the main representative of this family of algorithms can be considered algorithm Eclat[10]. This is memory-optimised and candidate partitioned version of Apriori-like algorithm using the vertical representation of a database for support counting.

Sequence Mining

After the initial proposal of the Apriori algorithm and the problem of frequent item-set mining, a similar approach was also proposed for mining of frequent subsequences or problem of sequence mining. The problem was defined by Agrawal and Srikant [11] as:

Definition 3. *Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-*

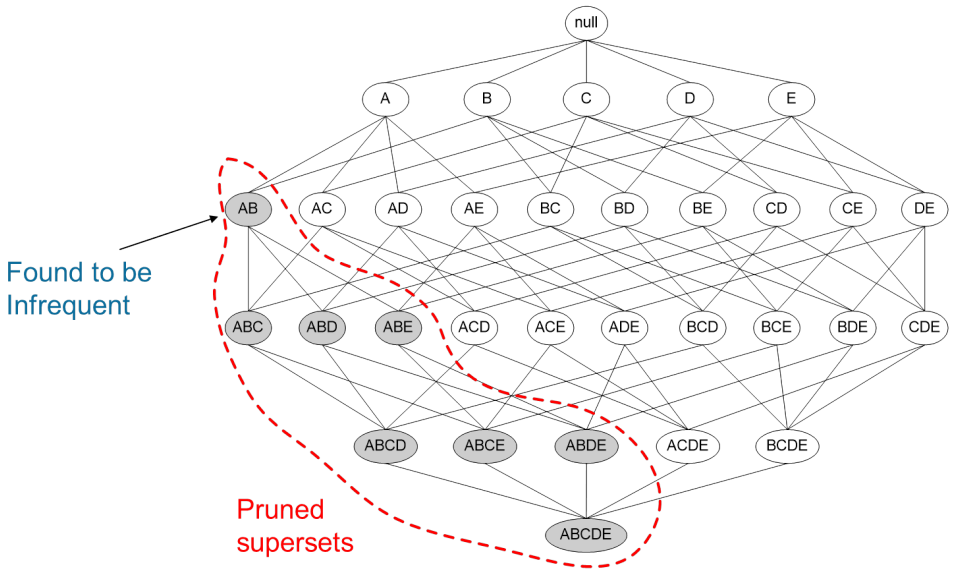


Figure 2.2: Complete exploration space for 5 items for problem of frequent item-set mining. For 5 elements the number of possible subsets it 2^5. Early elimination of infrequent subsets can prune space considerably.

specified min_support threshold, sequential pattern mining is to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.

Sequential pattern mining developed in a similar manner as frequent item-set mining. Two main families of algorithms developed Apriori based and pattern growth-based.

Apriori based algorithms incorporate algorithms AprioriAll[11], GSP[12], PSP[13], SPADE[14]and SPAM[15]. All of these algorithms use downward closure property to generate level-wise candidate patterns. These algorithms can be divided into two categories based on their support counting, as with horizontal data format and vertical data format. AprioriAll, GSP and PSP can be categorized into a group with a horizontal data format in which data are normally represented as a transaction database. SPADE and SPAM are algorithms from the second group, where transactions are transformed into tidLists.

In pattern growth family, the main focus is on avoiding candidate generation that consumes a lot of memory. The main idea is to repeatedly narrow the search space by dividing the database into smaller ones that are mined separately. Main representatives of this family of algorithms are FreeSpan[16] and PrefixSpan[17].

Graph Mining

Graph mining is the process of extraction of graph structure, that commonly occur either in a database of graphs or in one large graph. These patterns can both explain to us some behaviour like in case of chemicals and biochemicals [18, 19, 20] or can be used to create better indexing structures[21].

To explain a graph mining problem, lets define several terms. A graph G is an pair (V, E) where V is a set of nodes and E is a set of edges between nodes. A subgraph S of such the graph G would be a pair (V_S, E_S) where $V_S \subseteq V$ and $E_S \subseteq E$. Most of the graphs have additional information connected to their elements. Let assume a function $l_v : V \to L_V$ that allows us to label nodes and a function $l_e : E \to L_E$ that allows us to label edges. Let say two subgraphs S and S' are same if there exist a mapping function $f : V_S \to V_{S'}, \forall v \in V_S, l_v(v) = l'_v(f(v)), \forall e \in E, l_e(e) = l'_e(f(e))$. If such function exist,

subgraphs are called isomorphic and can be supposed for support counting as same.

Support of subgraph is then the number of isomorphic subgraphs in transactions that contain graphs, or in one large graph. However, the problem of finding isomorphic graphs was proved to be NP-complete[22]

As in the case of frequent item-set mining and sequence mining, also in graph mining algorithms can be divided into two groups: Apriori like and pattern growth-based.

Apriori like algorithms starts with small subgraphs and in a bottom-up manner generates new subgraphs by joining two similar but not same subgraphs, that were found frequent in the previous step. In Apriori like group, algorithms like AGM[23] and FSG[24] can be found.

In pattern growth based family of algorithms, new subgraphs are generated by adding edges into new possible position, so it does not perform expensive join operation. One of the possible issues is that same graph can be discovered from multiple subgraphs. gSpan[25] tries to reduce the number of similar graphs by introducing new nodes in the rightmost path. Other algorithms from pattern growth family are FFSM[26], Spin[27] and Gaston[28].

Association Rule Mining

Mining process can be divided into two different groups. First, there is descriptive, which tries to describe data in collection and summarise general properties of data. Whereas second, prescriptive tries to infer new knowledge about data collection and to use it for prediction from historical data into the future. Association mining falls into the second group.

The aim of association rule mining is to analyse the correlation between data and found patterns. These causalities can be then used for optimisation.

An example can be found in the classical market basket problem. Imagine that thanks to frequent item-set mining, a pattern that bread and butter are bought together. Thanks to this information, be can organise shop to prolong path through the shop by putting these two items on the other side of the shop. However, if we closely analyse in

which direction there is a stronger correlation, we can find out that in 80% cases, when butter was in the basket, bread was also there. Thanks to this discovery, we can put butter closer to the entrance to the shop and promote it in expectation of buyer buying also bread.

The information about butter leading to bread is called association rule. The rule is often in the form of two patterns and intuitively can be interpreted as: "If the first pattern occurs, then the second pattern occurs." Association rule has meaning only together with some measure, that will tell us the likelihood of this event. The most notorious measure used is the Support/Confidence framework.

Definition 4. *Association Rule Let A and B be two sets of items. The rule $A \Rightarrow B$ is said to be valid at support level s and confidence level c, if the following two conditions are satisfied:*

1. *The support of the item set A is at least s.*
2. *The confidence of $A \Rightarrow B$ is at least c.*

Process of generating association rules consist of two parts, finding frequent patterns and estimating their confidence. First part can be done by before mentioned techniques. For the second part, there exist several measures that can be used. List of measures follows:

Support/Confidence The basic measurement of the relevancy of the rule is called Support/Confidence[29]. This measurement is computed from support obtained during mining of the pattern frequency. The formula for computing confidence of the association rule follows:

$$Conf(X \Rightarrow Y) = \frac{supp(X \cup Y}{sup(X)}$$

Lift It was observed that confidence could be quite high even when items are unrelated. To take into account this weakness, a new measure called Lift[30] was proposed. This measure takes into account also the occurrence of the second pattern as can be seen in formula bellow.

$$Lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) * sup(Y)}$$

Cosine Another measure is the cosine measure[31]. This measure is null-invariant. This mean, that measure is not changing for the same number of occurrences of the patterns but the different size of the database.

$$Cosine(X \Rightarrow Y) = \frac{supp(X \cup Y)}{\sqrt{supp(X) * sup(Y)}}$$

2.1.2 Similarity Based Analysis

There exist several approaches to type-less analysis based on the similarity of objects. These are based on observation of structures withing objects. Distances like edit distance or Jaccard coefficient are used for comparison of these object or group of objects.

Clustering

One of the traditional data mining tasks is clustering. The goal of the task is to group data objects into clusters, so objects in one cluster are closer to each other than objects from different clusters. In the clustering, there are three main families of the algorithms:

- Hierarchical clustering
- Representative based clustering
- Density based clustering

Hierarchical Clustering Algorithms in this family iteratively build a hierarchical structure. There are two approaches to hierarchical clustering. First, bottom-up (agglomerative) approach tries to merge the most suitable clusters together and create bigger ones. Comparison between groups can be either based on the smallest distance between group members, biggest distance, or the one, that will minimize the change in the variance.

Top-down (divisive) approach starts with all the data in one cluster and in each step, it tries to partition existing clusters into smaller ones. For criteria of division can be specified by the user or another clustering algorithm can be used.

Representative Based Clustering Sometimes called K-means based is a group of algorithms, that use the partitioning representatives as a lead for building clusters. Clusters created this way do not have any hierarchy between them. Algorithms like K-means, K-medians and K-medoids belong to this family. For these algorithms user select parameter k, which correspond to the number of resulting clusters. Algorithms then try to find the best grouping by updating representative and assigning respective nodes to them.

Density Based Clustering In the previous two families, algorithms mostly used the distance between nodes as a measurement. In this family of algorithms, the main focus is to identify dense regions in data and create clusters by joining neighbouring dense clusters. The most known density-based algorithm is DBSCAN[32]. This algorithm marks points into core, border and noise points based on amount and type of neighbouring nodes. Based on neighbouring core and bordering nodes, clusters are then assembled.

Similarity Search

In similarity search, the main task is to obtain similar objects in database $\mathcal{D}$. In general, the search problem has been described by Zezula et al. [33]:

> Let $\mathcal{D}$ be a domain, d a distance measure on $\mathcal{D}$, and ($\mathcal{D}$,d) a metric space. Given a set X $\subseteq \mathcal{D}$ of n elements, preprocess or structure the data so that proximity queries are answered efficiently.

In similarity search, there are several types of queries, that can be used for searching data. Range query and nearest neighbour query are among the most commonly used. Another query types can be found in [33].

Range Query Range query is probably the most common type of similarity query. The query is specified by a query object $q \in \mathcal{D}$ and distance r that constraint the result. The task is to retrieve all objects that are in range, formally:

$$R_d(q, r) = \{o \in \mathcal{D}, d(o, q) \leq r\}$$

To interpret the result of the query, we can use a geographic example: *Give me all ATMs within a one-kilometre radius from my position.*

Nearest Neighbor Query Whenever a range query is used, the maximal range must be provided. Sometimes the user does not know the distances between objects. So other option is to specify not the range, but the number of objects retrieved. This query is often called kNN query, where k is the parameter of the query and meaning of it is the amount of nearest objects. KNN query formal definition follows:

$$kNN(q) = \{R \subseteq \mathcal{D}, |R| = k \wedge \forall k \in R, y \in \mathcal{D} - R : d(q,x) \leq d(q,y)\}$$

Interpretation of this query in the context of geography is: *Give me five closest restaurant to my position.*

Indexing Structures One of the goals of similarity searching is to answer queries efficiently. Intuitive strategies for query evaluation based on sequential scanning of the data is inefficient. Because of that, multiple indexing structures were developed to optimise the number of comparisons. The examples of this structures are M-Tree[34], M-index[35], D-index[36] and many others[33].

Similarity Join

One of the possible tasks in data mining is to clean or integrate heterogeneous data from multiple sources. This task is often approached by the usage of similarity join. Similarity join takes to sets of objects and assigns to objects of first set objects from the second set accordingly to their similarity. Similarity join can be formalised as:

$$SJ_d(X, Y, \mu) = \{(x,y) \in X \times Y : d(x,y) \leq \mu\}$$

Parameter μ is specified by the user, and intuitively can be described as how much objects must be similar.

There are several special cases, which can be seen in the literature. First, if $\mu = 0$, objects from different sets must be identical; this case is called *natural join*. The second case, when $X = Y$, then the case is called similarity self join.

Outlier Detection

There exist cases where we are not interested in normally behaving objects, but instead, we are looking for exceptions. Such objects that behave so differently than the rest of the data are called outliers. Aggarwal[3] described outliers as:

> "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."

Most of the approach tries to create a model of normal data and detect outliers by not fitting into this model. For the creation of these models, techniques from clustering, distance-based quantisation or dimensionality reduction can be used.

In this thesis proposal, the main focus will be on distance and density-based outlier detection.

Distance-based Outlier Detection This is the most intuitive and straightforward approach to outlier detection. The kNN query is often used and as an outlier is supposed every node, which k-th nearest neighbour distance is biggest. This technique has been used for detection of land mines from satellite ground images and to detect shorted turns (anomalies) in the DC field winding of large synchronous turbine-generators[37].

Density-based Outlier Detection In density-based outlier detection, a similar approach is used for the comparison of nodes. But instead of analysing distance to the k-th nearest neighbour, local outlier factor (LOF) is compared. The intuition behind using this technique is that outliers will be in a region with lower density; thus their LOF will be higher.[37]

2.2 Analytical Systems and Libraries

In the following section, there is a description of libraries and systems that provide techniques and algorithms for mentioned tasks. The main

focus is on the available techniques and their possible combination capabilities.

There exist a wide range of tools available for data mining. There were selected a few well known in literature:

- SPMF
- ELKI
- Weka

2.2.1 SPMF

Library SPMF is large collection of pattern mining techniques developed and maintained by Fournier-Viger et al.[38] This collection contain over 178 algorithms and focuses on areas:

- association rule mining
- frequent itemset mining
- sequential pattern
- sequential rule mining
- sequence prediction
- periodic pattern mining
- episode mining
- high-utility pattern mining
- time-series mining
- clustering
- classification

Although the library provides a large number of algorithms, some of the implementations are prototypical and are heavily limited. As an example, implementation of NegFin allows usage of only four

thousands of transactions for being processed which is a really low number for data mining and big data purposes.

The library defines textual input different for every task. The data must be input from a file and the library output it into a file. Due to this practice, it is really inconvenient for combining tasks into a more complex analysis.

Plus of this library comes with an integration of new algorithms, that other libraries are sometimes missing.

Library completely lacks the algorithms for tasks of similarity searching or similarity join.

2.2.2 ELKI

One of the possible tools that contain both pattern mining techniques and similarity-based analytical techniques is an environment for deveLoping Kdd-application supported by Index-structures (ELKI)[39]. This tool focuses primarily on research and development of algorithms for unsupervised methods in cluster analysis and outlier detection.

The main focus of this tool is to provide a better environment for the evaluation of the quality of new algorithms. To do so, ELKI divides process on the data mining part and data management part. This is because of the need for precise evaluation of processing time and other metrics of compared algorithms. ELKI contains a high number of methods and index structures for evaluating purposes.

For pattern mining, ELKI only contains a limited amount of techniques. All the available algorithms are only for frequent item-set mining task or association rule mining task. Inside this tool are implemented these algorithms for frequent item-set mining: Apriori, FP-Growth, Eclat.

For association rule mining task, ELKI provides high range of metrics for evaluation of association rules. To generate them, ELKI does not use optimised algorithms; it instead generates them from frequent item-sets. ELKI allows specifying which algorithm is used for generating frequent item-sets.

From the area of the clustering, ELKI provides high range of algorithms from families such as DBSCAN, K-means and Hierarchical clustering.

This tool provides high range of techniques for outlier detection. ELKI provides different variants of ABOD algorithm for detection based on angles, algorithms based on clusters such as K-means Outlier Detection, Cluster-based local outlier factor (CBLOF) and Silhouette Coefficients based clustering. There are several algorithms based on distance computations based on K-Nearest Neighbours. The tool provides a plethora of other algorithms based on intrinsic dimension, local outlier factor, spatial based, subspace-based and also provide Support-Vector Machines for detection of outliers.

The tool does not support other tasks like similarity searching or similarity join and other tasks from pattern mining like sequence and graph mining. Also, because of different objective for this tool, it lacks the support for chaining multiple basic techniques together into more complex analytical functions.

2.2.3 Weka

Waikato Environment for Knowledge Analysis [40] is a system developed by the University of Waikato for the purpose of providing easy access to data mining and machine learning techniques. This system, written in Java consists of several libraries and easy to use interface.

Weka accepts input in the form of Attribute-Relation File Format (ARFF)[41]. This format consists of two parts, header and data. Former defines the type of input, whereas the latter contains data in the CSV-like format. Format restricts the type of input data heavily, because it allows only numeric, string date or nominal value.

Weka provides a list of packages containing different algorithms and techniques, and whole list can be seen at address: `https://weka.sourceforge.io/packageMetaData/`

Weka provides Apriori algorithm for frequent item-set mining and GSP for sequence mining. This tool does not provide any algorithms for graph mining.

Weka provides DBSCAN, several variants of K-means and several other techniques for clustering. This tool does not provide any algorithms for processing similarity queries.

As the main task for weka is to provide an environment for training classifiers, it provides much more tools for one-time classification, than for chaining classifiers, but such an approach is possible. Thou

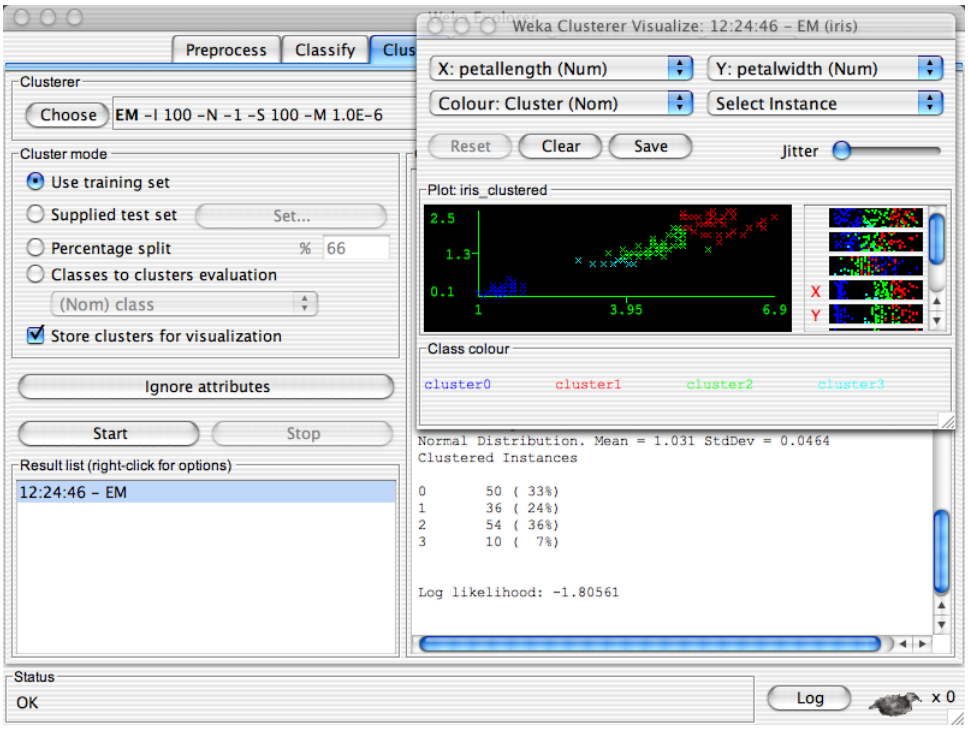


Figure 2.3: Weka user interface: cluster panel

output does not have a unified format due to a big number of different techniques, so the transformation of result back to ARFF format is necessary.

3 Aims of the Thesis

Data analysis is a complex process that typically requires execution of multiple specific analytical tasks. In previous chapters, we have described the actual state of the field and identified several problems in the current state-of-the-art. These can be summarised as follows:

1. Contemporary data are more complex and unstructured than in the past. They are also much more interlinked, and they must be analysed in the context of these links.

2. Current analytical tools are typically specialised, and a use of multiple of them is required for more complex analyses.

3. Input formats of the analytical tools typically differ and transformations of data between these formats are difficult or sometimes impossible.

These three problems cause that the data analysis requires a lot of knowledge and is very time-consuming. The solution to these problems provides sufficient space for further research of more complex analytical functions. The aims of this thesis are:

- to provide a new unified data structure that can be used by all analytical operations as input and output. This structure should allow the formulation of complex operations by chaining the elementary ones.

- to formalise the standard data analytical operations and their interactions in the form of algebra that allowing to describe the process accurately and to estimate the cost; thus allows to optimise the execution of user-defined queries.

- to develop an analytical query language that offers an easy way for an analyst to describe the desired process with the focus on the data analysis concepts instead of specific implementation details of specific analytical techniques.

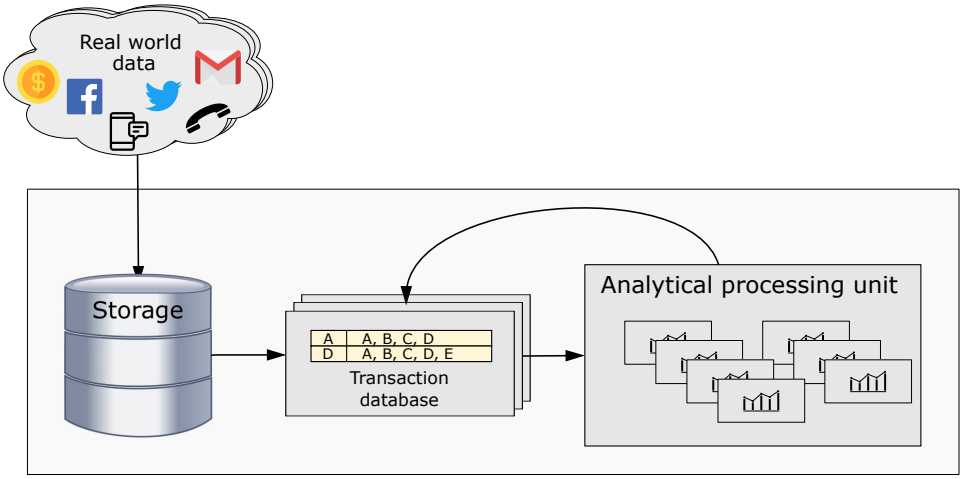


Figure 3.1: Model of system ADAMiSS for complex data analysis.

We perceive the collection of these aims as an interconnected whole elevating the process of data analysis and providing sufficient environment for before mentioned research of complex analytical functions. A brief overview of such an environment can be seen in Figure 3.1.

The diverse data can be stored in the form of a graph for the process of analysis in storage. Such storage can be realised by a suitable graph database. The graph representation was selected because it provides a sufficient capability to store data in the rawest form and capture their context. However, analytical operations over such representation are difficult and costly.

We want to utilise a new data structure, transaction file, that deals with the problem of a complicated form of the data in the storage. It is a flat structure consisting of a set of transactions, where each transaction consists of the transaction body and attributes of the transaction. Body of a transaction is a list of elements, that are application-defined objects. This structure provides flexibility for the analytical operations because data can be perceived as multi-sets, sets, ordered sets, vectors, individual items and many other forms. Because of this, the transaction file can be used as an input of multiple analytical operations. The transaction file also serves well as a format for analytical operations. This possibility simplifies the process of combining analytical operations into more complex functions.

It is necessary to transform data from storage into the transaction file. This transformation is dependent on the type of desired analysis, and it is not possible to do it automatically. Because of that, the responsibility of transforming correct data is on the user of such a solution.

As a part of the solution, we want to create an analytical module that is going to contain algorithms for several data analytical operations. This is going to provide simple access to necessary analytical operations in one system.

We want to propose a formalisation of data analytical tasks, which is contemplating the transaction file. Based on this formalisation, we are going to create a new analytical language that serves to describe analysis on the level of concepts. This language is inspired by SQL and database systems. Language is going to provide simple means to describe complex analytical pipelines, without the need for specifying algorithms or providing transformations of data in between. Such language is going to open the area of a combination of analytical functions significantly.

Similarly to database systems and the SQL language, we want to propose a module that is going to process user-defined queries in the analytical query language. The module is going to select the right algorithms based on the data provided in the transaction file to efficiently process user-defined queries. Possibility of optimisation of the queries as in SQL language is going to be researched.

3.1 Research Plan

The plan of the research is going to focus on the development of the key features of the system to create a system following the transformationless concept of analysis. Several steps have been done; these are described in chapter 4. The research steps follow:

2020 In the first half of the 2020 year, we are going to focus on formalisation of the data analysis tasks in regard to ADAMiSS architecture. Next step is to develop a basic structure of the analytical language. The prototype of the system is going to be extended to process this language. We plan to publish the formalisation of the tasks in the context of ADAMiSS and the definition of new

analytical language in the form of conference papers. There is a possibility of extending the description of analytical language into a journal paper.

2021 After the creation of the analytical language; the next step is to create a module for an optimisation of user queries. As described above, this module is going to select a processing plan and create pipelines in optimise way. This part should make the system flexible and usable. We plan to describe the proposed module as well as possible optimisation in the form of the conference paper with possible extension for journal paper.

2022 Last year will be focused primarily on the summarisation of the system and preparation of dissertation thesis.

The suitable publishing platforms for proposed research papers are conferences focused on data mining, data processing, similarity search and knowledge discovery.

4 Achieved Results

In this chapter, there is a description of the work, that has been already done. As a first step to creating a system for analysis, a sample system containing pattern mining and similarity searching techniques was proposed, and a prototype was developed.

4.1 ADAMiSS

In this section, there is a description of the architecture of system ADAMiSS. A brief schema can be seen in Figure 3.1. ADAMiSS is composed of several components and serves for storing and processing of data. The components are the following:

1. Storage
2. Transaction file
3. Analytical processing unit

4.1.1 Storage

Data are imported into ADAMiSS storage in raw form to minimise this loss. To allow such import, storage is in the form of a multigraph $S : (V, E)$, where V is a set of data objects and E is a multi-set of edges. Each data object $v \in V$ is pair $(ID, ATTRs)$ where ID serves as a simple representation of the object. Complex data of objects are stored as attributes in map of attributes $ATTRs$ that contains *key:value* pairs. Possible interactions between data are stored in edges $e \in E$. Each edge e is triplet of two data objects v_1 and v_2 and set of attributes $ATTRs$ to store additional information.

The multigraph structure with attributes as a representation of raw data was selected, because it seems, that most of the data can be stored in this structure without loss of the information. There exist other structures like hypergraph, but the usage of attributes can simulate such structures.

An important property of this storage is, that system is capable of capturing heterogeneous structures, and manage them without bigger

nuisance. Let phone communication be an example; then individual phones are imported as nodes in the network. Data as an owner of the phone and model of the phone are stored in respective attributes. Calls between phones are captured as edges and attributes as the duration of call and voice messages are captured. Then other data such as financial transactions or e-mail communication can also be added in a similar way. A newly created heterogeneous network of data then can be processed by other parts of the system based on user demands.

4.1.2 Transaction File

Because storage structure is too complex and providing analysis over it is in many cases expensive, ADAMiSS system contains *transaction file* structure. It is set of transactions $t : (Data, ATTRs)$. $Data$ is list of transaction items $(i_1, \ldots, i_n)$, which are application-defined objects. The only restriction on these items is that the check of identity must be defined. $ATTRs$ is set of *key:value* pairs as in case of storage objects. Attributes in transactions are used for storing the result of analysis and intermediate results.

The main advantage of this flat structure is that a lot of different views on the $Data$ can be used. It can be perceived either as ordered list, as multi-set, or as a set. This possibility of different views allows to process the same data by different operations; thus, it serves as a universal input into several analytical operations.

4.1.3 Analytical Processing Unit

The vital part of the whole ADAMiSS system is *Analytical processing unit*. It is a modular collection of analytical operations that works over the transaction file and as a result, provide transaction file. To this collection, every function that meets $F : ([\mathcal{D}, ...], params) \rightarrow \mathcal{D}$ can be added.

As a part of the analytical processing unit, there is a simple formalisation of the tasks in the context of the proposed architecture. The list of task with brief description and formalisation follows:

Frequent Item-set Mining is a task of the discovery of frequent sets of co-occurring items. A user-defined threshold determines the frequency of a pattern. The task can be defined as a function $IM : (\mathcal{D}, \theta) \rightarrow \{(p, \{"support" : s_p\})\}|p \subseteq t.data; t \in \mathcal{D}; s_p > \theta$ where p is a set of items, s_p is support computed over the original transaction file.

Sequence Mining serves for searching for frequent subsequences in the sequential data. Data in a transaction of transaction file are taken as a list, and common order of data items are analysed. The task can be described as a function $SM : (\mathcal{D}, \theta) \rightarrow \{(p, \{"support" : s_p\})\}|p \subseteq t.data; t \in \mathcal{D}; s_p < \theta$, where s_p is support computed over original transaction file.

Association Rule Mining is a process of deriving association rules based on a Support/Confidence framework. The association rule is triplet $(p_0, p_1, conf(p_0, p_1))$ where p_0 and p_1 are patterns, and a function $conf(p_0, p_1)$ describes the probability of how often based on the occurrence of the pattern p_0 the pattern p_1 will be present. Support/-Confidence framework can be replaced by other measures evaluating before mentioned probability, like Lift or Kulczynski measure [42]. Operation of mining such rules is a function $AM : (\mathcal{D}, p_q, \theta) \rightarrow \{(p_i, \{"confidence" : conf(p_q, p_i)\})\}|\exists t_j : t_j \in \mathcal{D}, p_i \subseteq t_j.data;.$

Range Search is an operator that searches for all the transactions that are similar to the selected query. The user defines the threshold for similarity. In the range search, an arbitrary distance function must be selected. The operation can be seen as a function $R_{dist} : (\mathcal{D}, q, r) \rightarrow \{(t_i, \{"distance" : d\})\}|T_i \in \mathcal{D} : dist(q, body(T_i)) < r\}$.

K-NN Search serves for searching K nearest, most similar transactions in the sense of distance evaluation. Suitable distance function *dist* must be selected. Formally, K-NN search is a function: $kNN_{dist} : (\mathcal{D}, q, k) \rightarrow \{(t_i, \{"distance" : dist(q, t_i.data\})\}|t_i \in \mathcal{D}; \forall t_j \in \mathcal{D} : dist(q, t_i.data) < dist(q, t_j.data)\}$. Query object q and amount of search neighbours k must be provided by user.

Similarity Self-join It is an operator that compares transactions between themselves and pairs them based on their similarity distance. The similarity distance is measured by the distance function *dist* provided by the user. The user also provides the similarity threshold. Similarity join is defined as $SJ_{dist} : (\mathcal{D}_1, \mathcal{D}_2, \theta) \rightarrow \{((t_1, \ldots, t_n), d)\}; t_1, \ldots, t_n \in \mathcal{D}_2; \forall t_q \in \mathcal{D}_1; dist(t_q, t_m) = d; d < \theta$.

Outlier Detection is a process of finding transactions, that differs from others in the transaction file. To evaluate the difference, the similarity of the transactions is used. Suitable distance function *dist* must be selected. Function $OD_{dist} : (\mathcal{D}, r) \rightarrow \{(t_i, \{"min_distance" : d\})\} | t_i \in \mathcal{D}; d = mindist(q, t_i) : \forall t_j in \mathcal{D}; d > r\}$ returns objects, which differs to most similar object at least r; thus they are outliers.

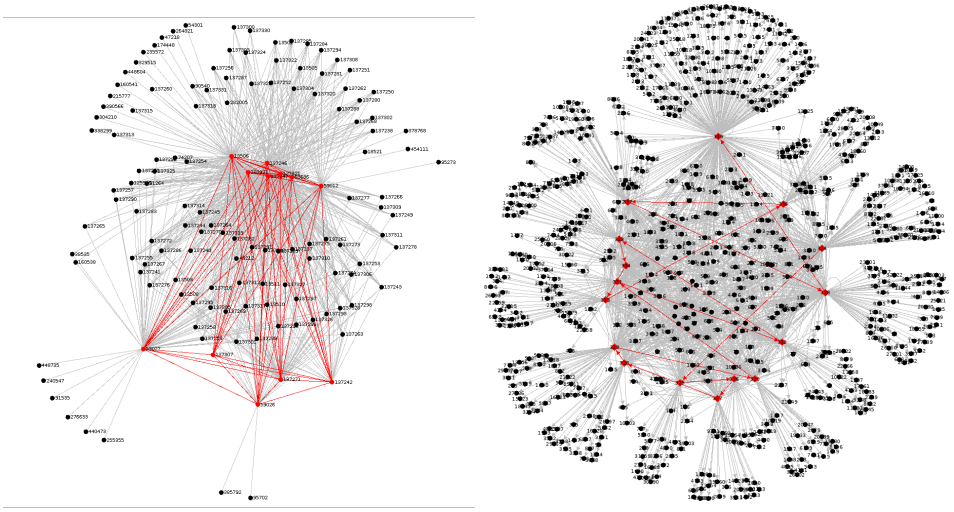
This list of operation shows the flexibility of the transaction file to provide a suitable representation of data to analyse. Due to the possibility of analytical operation to return results in the form of transaction file, it is possible to chain operation one after the other and thus to iteratively create more complex operations. The proposed architecture is able to store and analytically process data in an iterative manner.

4.2 Prototype of ADAMiSS

We developed a prototype application to show the usability of ADAMiSS. Two datasets were used, Twitter Higg's Boson dataset [43] and Kosarak dataset [44].

All experiments were performed on a server with 4 cores Intel Xeon CPUs and 16GB RAM.

We apply frequent item-set mining operator to dataset Twitter Higg's boson. We use a slightly modified FP-Growth algorithm for mining communities instead of frequent sets. This dataset contains interactions between Twitter users with a topic connected to the discovery of Higg's boson at CERN. The dataset contains interactions such as mentioning, retweeting and replying. We transform each account and all neighbouring accounts as transactions. Transaction file consists of 304691 transactions/accounts, and the average length/degree of transaction/node was 3.96. The small degree of nodes is because many



(a) Community of size 12 at thr. 11 (red) (b) Paths of length 16 at thr. 1024 (red)

Figure 4.1: Example of identified patterns in different datasets.

accounts have only one connection to the star hub, i.e. node with many nodes connected just to the hub node.

The goal is to obtain communities of twitter accounts that share information about the discovery of Higg's boson between themselves. We use a strict definition of community as a fully interconnected subgraph, i.e. clique. There was selected a threshold of size 11, which is quite a low number for pattern mining in such a database. Reason for small threshold is that social networks are quite shallow graphs, so there is not expected to have bigger cliques containing a higher amount of unique accounts. This assumption showed up as true because the largest cliques found are of size 12. To show the result, we picked one random community of size 12 and visualised it with neighbouring nodes in figure 4.1.

We discovered 7 communities of size 12 and 94 communities of size 11. Average overlap of all these communities is 80.7%. Communities were further analysed if they contain a common subset, but none such subset was found.

For sequence mining, ADAMiSS contains modified GSP algorithm that does not allow gaps. We used Kosarak dataset for an experiment. This dataset contains 990002 click-stream data from the logs of an

online news portal. Click-stream is a sequence of browsed web pages by one user on the web page domain. We use sequences of click-stream as paths of users. The threshold was set to 1024. A result contains 322 paths and just 5 paths that contain more than 4 nodes. The longest path contains 16 nodes. Sub-paths of found paths were not taken into account. The longest path is visualised with the corresponding neighbourhood of nodes on the path in figure 4.1.

We use one selected community found by dataset Twitter Higg's boson for a test of the similarity searching. One of the communities of size 12 was randomly picked, and for each node, we search for similar nodes. The similarity of nodes is defined as the similarity of neighbouring sets. Jaccard coefficient was selected as a distance function. ADAMiSS uses Messif framework developed by Batko et al [45] for efficient processing.

For range queries, the threshold was selected to 0.2 that mean 80% of the set is shared. Four nodes have their most similar items inside the community. One node (137328) has six nodes most similar to itself outside of the community. Other nodes did not have sufficiently similar nodes.

We selected 10 as k for k-nn queries, and an average amount of most similar nodes from the community is 8.33 nodes. One node (137328) has all ten most similar nodes outside of the selected community.

Underlying algorithm used for evaluation of both range query and k-nn query was M-index [35].

5 Author's Publications

Conference paper "ADAMiSS: Advanced Data Analysis, Mining and Search, System"[46] was presented at conference SISAP'19 in Newark, New Jersey, USA. This paper describes the main idea of analytical system ADAMiSS with motivation and showcase usability by results of the analysis on the implemented prototype.

- PESCHEL, Jakub; ZEZULA, Pavel. ADAMiSS: Advanced Data Analysis, Mining and Search, System. In: International Conference on Similarity Search and Applications . 2019, pp. 351–355.

Full text of the conference paper can be found in Appendix A. The contribution of Jakub Peschel to this conference paper is estimated at 85%.

Conference paper "Techniques for complex analysis of contemporary data" describes the architecture of ADAMiSS in more detail together with the formalisation of standard data analytical tasks in the context of this system. The formalisation of the tasks shows the possibility of chaining of methods and allows to create analytical language as described in chapter 3. This conference paper was submitted to the conference, but its acceptance was not decided yet.

Full text of the conference paper can be found in Appendix A. The contribution of Jakub Peschel to this conference paper is estimated at 70%.

Bibliography

1. AGGARWAL, Charu C; HAN, Jiawei. *Frequent Pattern Mining*. Springer, 2014.
2. AGRAWAL, Rakesh; SRIKANT, Ramakrishnan, et al. Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*. 1994, vol. 1215, pp. 487–499.
3. AGGARWAL, Charu C. *Data mining: the textbook*. Springer, 2015.
4. HAN, Jiawei; PEI, Jian; YIN, Yiwen. Mining frequent patterns without candidate generation. In: *ACM sigmod record*. 2000, vol. 29, pp. 1–12. No. 2.
5. AGARWAL, Ramesh C; AGGARWAL, Charu C; PRASAD, VVV. *Depth first generation of large itemsets for association rules*. IBM Thomas J. Watson Research Division, 1999.
6. DENG, Zhihong; WANG, Zhonghui. A new fast vertical method for mining frequent patterns. *International Journal of Computational Intelligence Systems*. 2010, vol. 3, no. 6, pp. 733–744.
7. DENG, ZhiHong; WANG, ZhongHui; JIANG, JiaJian. A new algorithm for fast mining frequent itemsets using N-lists. *Science China Information Sciences*. 2012, vol. 55, no. 9, pp. 2008–2030.
8. DENG, Zhi-Hong; LV, Sheng-Long. Fast mining frequent itemsets using Nodesets. *Expert Systems with Applications*. 2014, vol. 41, no. 10, pp. 4505–4512.
9. ARYABARZAN, Nader; MINAEI-BIDGOLI, Behrouz; TESHNEHLAB, Mohammad. negFIN: An efficient algorithm for fast mining frequent itemsets. *Expert Systems with Applications*. 2018, vol. 105, pp. 129–143.
10. ZAKI, Mohammed Javeed. Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*. 2000, vol. 12, no. 3, pp. 372–390.
11. AGRAWAL, Rakesh; SRIKANT, Ramakrishnan, et al. Mining sequential patterns. In: *icde*. 1995, vol. 95, pp. 3–14.

12. SRIKANT, Ramakrishnan; AGRAWAL, Rakesh. Mining sequential patterns: Generalizations and performance improvements. In: *International Conference on Extending Database Technology*. 1996, pp. 1–17.

13. MASSEGLIA, Florent; CATHALA, Fabienne; PONCELET, Pascal. The PSP approach for mining sequential patterns. In: *European Symposium on Principles of Data Mining and Knowledge Discovery*. 1998, pp. 176–184.

14. ZAKI, Mohammed J. SPADE: An efficient algorithm for mining frequent sequences. *Machine learning*. 2001, vol. 42, no. 1-2, pp. 31–60.

15. AYRES, Jay; FLANNICK, Jason; GEHRKE, Johannes; YIU, Tomi. Sequential pattern mining using a bitmap representation. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 429–435.

16. HAN, Jiawei; PEI, Jian; MORTAZAVI-ASL, Behzad; CHEN, Qiming; DAYAL, Umeshwar; HSU, Mei-Chun. FreeSpan: frequent pattern-projected sequential pattern mining. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 355–359.

17. PEI, Jian; HAN, Jiawei; MORTAZAVI-ASL, Behzad; PINTO, Helen; CHEN, Qiming; DAYAL, Umeshwar; HSU, Mei-Chun. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: *Proceedings 17th international conference on data engineering*. 2001, pp. 215–224.

18. BORGELT, Christian; BERTHOLD, Michael R. Mining molecular fragments: Finding relevant substructures of molecules. In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* 2002, pp. 51–58.

19. DESHPANDE, Mukund; KURAMOCHI, Michihiro; WALE, Nikil; KARYPIS, George. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*. 2005, vol. 17, no. 8, pp. 1036–1050.

20. HUAN, Jun; WANG, Wei; BANDYOPADHYAY, Deepak; SNOEYINK, Jack; PRINS, Jan; TROPSHA, Alexander. Mining protein family specific residue packing patterns from protein structure graphs. In: *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*. 2004, pp. 308–315.

21. YAN, Xifeng; YU, Philip S; HAN, Jiawei. Graph indexing: a frequent structure-based approach. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 2004, pp. 335–346.

22. GAREY, Michael R; JOHNSON, David S. A Guide to the Theory of NP-Completeness. *Computers and intractability*. 1979, pp. 641–650.

23. INOKUCHI, Akihiro; WASHIO, Takashi; MOTODA, Hiroshi. An apriori-based algorithm for mining frequent substructures from graph data. In: *European conference on principles of data mining and knowledge discovery*. 2000, pp. 13–23.

24. KURAMOCHI, Michihiro; KARYPIS, George. Frequent subgraph discovery. In: *Proceedings 2001 IEEE International Conference on Data Mining*. 2001, pp. 313–320.

25. YAN, Xifeng; HAN, Jiawei. gspan: Graph-based substructure pattern mining. In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* 2002, pp. 721–724.

26. HUAN, Jun; WANG, Wei; PRINS, Jan. Efficient mining of frequent subgraphs in the presence of isomorphism. In: *Third IEEE International Conference on Data Mining*. 2003, pp. 549–552.

27. HUAN, Jun; WANG, Wei; PRINS, Jan; YANG, Jiong. Spin: mining maximal frequent subgraphs from graph databases. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 581–586.

28. NIJSSEN, Siegfried; KOK, Joost N. A quickstart in frequent structure mining can make a difference. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 647–652.

29. AGGARWAL, Charu C; YU, Philip S. A new framework for itemset generation. In: *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1998, pp. 18–24.

30. BRIN, Sergey; MOTWANI, Rajeev; ULLMAN, Jeffrey D; TSUR, Shalom. Dynamic itemset counting and implication rules for market basket data. *Acm Sigmod Record*. 1997, vol. 26, no. 2, pp. 255–264.

31. TAN, Pang-Ning; KUMAR, Vipin; SRIVASTAVA, Jaideep. Selecting the right objective measure for association analysis. *Information Systems*. 2004, vol. 29, no. 4, pp. 293–313.

32. ESTER, Martin; KRIEGEL, Hans-Peter; SANDER, Jörg; XU, Xiaowei, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. 1996, vol. 96, pp. 226–231. No. 34.

33. ZEZULA, Pavel; AMATO, Giuseppe; DOHNAL, Vlastislav; BATKO, Michal. *Similarity search: the metric space approach*. Springer Science & Business Media, 2006.

34. CIACCIA, Paolo; PATELLA, Marco; ZEZULA, Pavel. M-tree: An E cient access method for similarity search in metric spaces. In: *Proceedings of the 23rd VLDB conference, Athens, Greece*. 1997, pp. 426–435.

35. NOVAK, David; BATKO, Michal; ZEZULA, Pavel. Metric Index: An efficient and scalable solution for precise and approximate similarity search. *Information Systems*. 2011, vol. 36, no. 4, pp. 721–733.

36. DOHNAL, Vlastislav; GENNARO, Claudio; SAVINO, Pasquale; ZEZULA, Pavel. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications*. 2003, vol. 21, no. 1, pp. 9–33.

37. NIU, Zhixian; SHI, Shuping; SUN, Jingyu; HE, Xiu. A survey of outlier detection methodologies and their applications. In: *International Conference on Artificial Intelligence and Computational Intelligence*. 2011, pp. 380–387.

38. FOURNIER-VIGER, Philippe; GOMARIZ, Antonio; GUENICHE, Ted; SOLTANI, Azadeh; WU, Cheng-Wei; TSENG, Vincent S. SPMF: A Java Open-Source Pattern Mining Library. *Journal of Machine Learning Research*. 2014, vol. 15, pp. 3569–3573. Available also from: `http://jmlr.org/papers/v15/fournierviger14a.html`.

39. SCHUBERT, Erich; ZIMEK, Arthur. ELKI: A large open-source library for data analysis - ELKI Release 0.7.5 "Heidelberg". *CoRR*. 2019, vol. abs/1902.03616. Available from arXiv: `1902.03616`.

40. HALL, Mark; FRANK, Eibe; HOLMES, Geoffrey; PFAHRINGER, Bernhard; REUTEMANN, Peter; WITTEN, Ian H. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*. 2009, vol. 11, no. 1, pp. 10–18.

41. PAYNTER, Gordon; TRIGG, Len; FRANK, E; KIRKBY, R. Attribute-relation file format (ARFF). *Online*] *http://www. cs. waikato. ac. nz/ml/weka/arff. html*. 2008.

42. WU, Tianyi; CHEN, Yuguo; HAN, Jiawei. Re-examination of interestingness measures in pattern mining: a unified framework. *Data Mining and Knowledge Discovery*. 2010, vol. 21, no. 3, pp. 371–397.

43. DE DOMENICO, Manlio; LIMA, Antonio; MOUGEL, Paul; MUSOLESI, Mirco. The anatomy of a scientific rumor. *Scientific reports*. 2013, vol. 3, pp. 2980.

44. BODON, Ferenc. A fast APRIORI implementation. In: *FIMI*. 2003, vol. 3, p. 63.

45. BATKO, Michal; NOVAK, David; ZEZULA, Pavel. MESSIF: Metric similarity search implementation framework. In: *Digital Libraries: Research and Development*. Springer, 2007, pp. 1–10.

46. PESCHEL, Jakub; ZEZULA, Pavel. ADAMiSS: Advanced Data Analysis, Mining and Search, System. In: *International Conference on Similarity Search and Applications*. 2019, pp. 351–355.

A Publications

ADAMiSS: Advanced Data Analysis, Mining and Search, System

Jakub Peschel( ) and Pavel Zezula

Masaryk University, Brno, Czech Republic
{jpeschel,pzezula}@mail.muni.cz

Abstract. The complexity of contemporary data warrants a need for better analysing tools in investigative areas. Human processing of data is no longer a viable option. We present an architecture of a novel universal system for analysis of graph-structured data, where data-mining and similarity-search operators can be used to discover or search for unknown information. We also present results that were obtained by our prototype implementation on two real-world data collections: the Twitter Higg's boson dataset and the Kosarak dataset.

Keywords: Data analysis · Pattern mining · Similarity search · Advanced data types

1 Introduction

In the investigative fields, such as forensic analysis, there is often a need for analysis of contemporary data gathered from different sources. The main problem that arises is that experts do not know what they are looking for until it is shown to them and then, based on the observation of newly discovered data, there is a need for searching techniques. There exist many techniques for these sub-tasks, such as SNAP [8], SPMF [6], MESSIF [3]. While there are various more specialised tools and approaches that can give even more detailed answer, the forensic analysts rather use a single generic system than multiple ones.

The critical aspect of contemporary data is that it has connections to others. Whether it is a response in the form of a tweet, a picture showing some event or record of purchase, a social network can be created from such information.

Search and mining are two main areas that we are interested in. The main focus on these areas is further narrowed into query by example search in case of searching and into pattern mining for the mining task.

Supervised by P. Zezula.

This work has been supported by the Ministry of the Interior of the Czech Republic under the "Security Research for the Needs of the State Program 2015–2020," through the Project No. VI20172020096, "Complex Analysis and Visualization of Large-scale Heterogeneous Data."

© Springer Nature Switzerland AG 2019
G. Amato et al. (Eds.): SISAP 2019, LNCS 11807, pp. 351–355, 2019.
https://doi.org/10.1007/978-3-030-32047-8_31

In query by example search, one of the main aspects used for searching is similarity between objects [10]. For this task, there exist various types of queries, but we focus on the two widely used ones: k-nearest neighbours query and range query. These two approaches allow for a high range of use-cases.

Pattern mining is a technique of discovery of frequent behaviour in the data. The main emphasis is put on periodicity or frequency. In connection with social networks, the main goals are mining communities and frequent sub-paths. Main techniques are older Apriori algorithm developed by Agrawal *et al.* [1], FP-growth developed by Han *et al.* [7] and GSP developed by Srikant *et al.* [9].

Goal: To address the mentioned problems, we want to propose a system that satisfies these requirements:

- general multipurpose system
- unified data structure
- modularity of system
- efficient processing
- large scalability

We call this system Advanced Data Analysis by Mining and Searching System, shortened as ADAMiSS.

The system for advanced analysis that is presented allows opening research space for the development of new techniques from combined ones.

2 Architecture

In forensic analysis, an expert looks at a high volume of data and tries to uncover new evidence. There is a need to obtain new insight into various data to gain hidden information.

To tackle the problem, we propose an architecture consisting of four components containing data and transformations between them visualised in Fig. 1. Transformations are a series of simple operations that reveal hidden information when combined in the right way. The ordering of operations is dependent on user requirement.

We selected multigraph as the appropriate data structure since a lot of real-world data can be viewed as entities introduced by links. Every element of multigraph is allowed to have attributes, to capture all the information needed for analysis.

ADAMiSS allows working with a broad range of data inputs. Examples of such inputs are interaction networks from Facebook or Twitter, e-mail message network, phone call network, bank transaction network, computer network and many others. The only prerequisite is that transformation into graph must exist.

Because every method needs a specific approach or view on data, ADAMiSS introduces the concept of transaction database. The transaction database is a

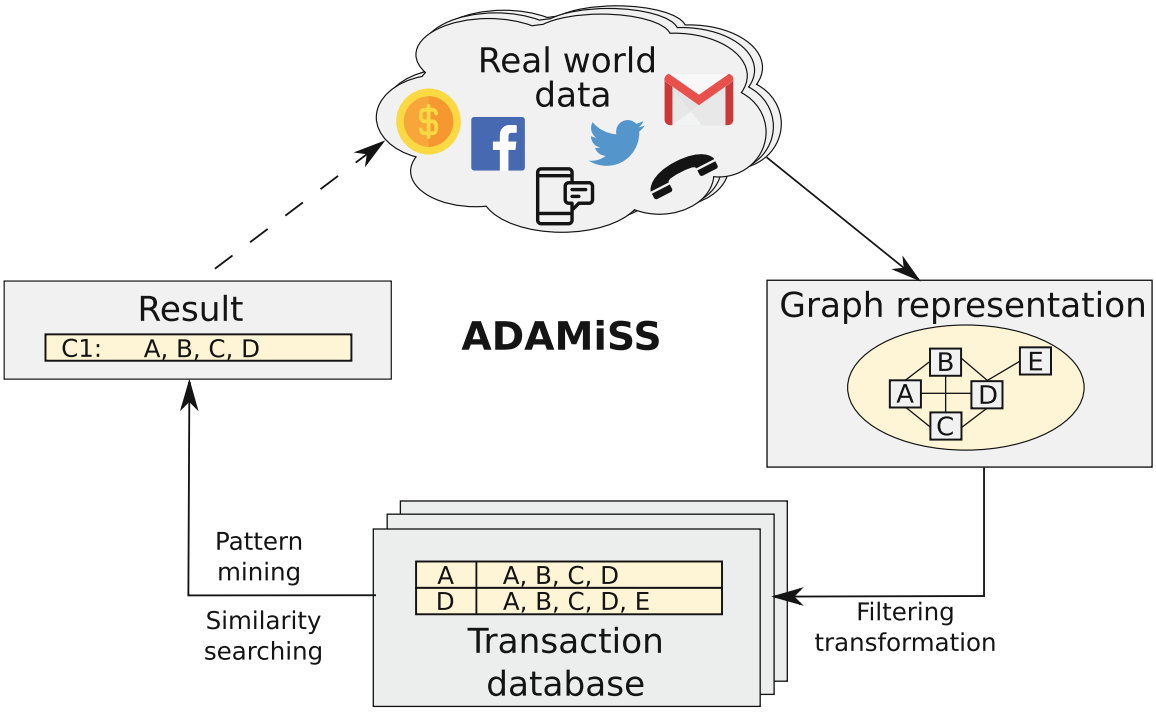


Fig. 1. Overview of an architecture of ADAMiSS

structure used by all the analytic operators. It consists of transactions and is used as a preprocessed data source for analytic operators. By the transformation into this flat structure, lots of the information from graph representation is omitted. Thus, it is necessary to select the right transformation for the desired operation.

Filtering transformation serves as a preprocess of data for analytic operators. During filtering, noisy data is removed, and only important structures are presented. An example of such preprocess is the removal of nodes without connection or taking into account only edges that appeared at least several times. By filtering, transformation can occur and a group node can be introduced by merging several nodes into one. A result of this transformation is a prepared transaction database.

To analyse the data, ADAMiSS contains analytic operators. These operators are basic functions that can be applied to the transaction database to obtain new knowledge. Due to the well-defined structure of transaction database, there is a possibility of implementing a large number of methods. Example operations that are available in the system:

Frequent item-set mining is a method that discovers information by providing frequent patterns of co-occurring items. The frequency of a pattern is determined by a user-defined threshold. In graph analysis, it can be used to find frequent neighbours. With a slight modification, it can be used to mine communities.

Sequence mining serves for searching for frequent paths in a graph. Data in a transaction of transaction database are then taken as a sequence, and ADAMiSS then searches for continuous subsequences. In sequence mining, it is often allowed

to have holes in patterns. Both approaches with holes and without holes can be applied to the transaction database.

Range search is an operator that searches for all the items that are similar to the selected query. The threshold for similarity is defined by the user. In the range search, an arbitrary similarity function must be selected. Such a function must satisfy the metric space properties. As an example, similarity can be measured on the sets of neighbours or at selected attribute values.

K-NN search is an operator very similar to the range search, but it retrieves k-most similar items to the ADAMiSS. In the same manner as in the range search, the suitable similarity function must be selected.

The proposed architecture seems to be very useful for an iterative approach to analysis. The main process is to import data selected for analysis, filter them by applying suitable filters and analyse them via analytic operators. The results of the analysis are then again imported for further analysis by searching or another round of pattern mining. This iterative approach is a process of creating an advanced analysis.

3 Prototype Results

Two datasets were used, Twitter Higg's Boson dataset [5] and Kosarak dataset [4], to showcase prototype application.

We applied a frequent item-set mining operator to dataset Twitter Higg's boson. This dataset contains 304691 interactions on Twitter with a topic connected to the discovery of Higg's boson at CERN. A threshold of size 11 was selected, because social networks are quite shallow graphs. TA total of 7 communities of size 12 and 94 communities of size 11 were discovered. Average overlap of all these communities is 80.7%.

For path mining, sequence mining was applied to dataset Kosarak, containing 990000 click-stream data from the logs of an online news portal. The threshold was set to 1024, to obtain quite short frequent paths. There was found 322 paths and only 5 paths containing more than 4 nodes. The longest path contained 16 nodes.

One of the communities of size 12 was picked, and the Jaccard coefficient was computed on its nodes. For efficient processing, ADAMiSS uses MESSIF library developed by Batko et al. [3].

For range queries, the threshold was selected to 0.2. Four nodes (60686, 137247, 137321, 137246) had their most similar items inside the community.

For k-nn queries, 10 was selected as k, and an average amount of most similar nodes from the community for each node was 8.33 node.

4 Conclusion

In this paper we introduced a new system ADAMiSS. This system is used for obtaining new information from large volumes of data that can be transferred

into a suitable graph representation. The system is based on unifying graph representation and flat transaction database used for two analytic approaches, pattern mining and similarity searching.

The system is highly modular; new methods can be developed and added. In the near future, we plan to add inference of association rules and to improve pattern mining methods by implementing new method NegFIN [2]. We would also like to improve our definition of community and to develop a suitable method for the discovery of such a structure. We also see a lot of space for research of interconnecting similarity search and pattern mining.

A prototype of the system was used for the analysis of two datasets, Twitter Higg's Boson dataset and Kosarak dataset. The system processed data and provided information about communities, frequent paths through the network and similarities between nodes.

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)
2. Aryabarzan, N., Minaei-Bidgoli, B., Teshnehlab, M.: negFIN: an efficient algorithm for fast mining frequent itemsets. Expert Syst. Appl. **105**, 129–143 (2018)
3. Batko, M., Novak, D., Zezula, P.: MESSIF: metric similarity search implementation framework. In: Thanos, C., Borri, F., Candela, L. (eds.) DELOS 2007. LNCS, vol. 4877, pp. 1–10. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77088-6_1
4. Bodon, F.: A fast apriori implementation. In: FIMI, vol. 3, p. 63 (2003)
5. De Domenico, M., Lima, A., Mougel, P., Musolesi, M.: The anatomy of a scientific rumor. Sci. Rep. **3**, 2980 (2013)
6. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM Sigmod Record, vol. 29, pp. 1–12. ACM (2000)
8. Leskovec, J., et al.: Stanford network analysis project (2010). http://snap.stanford.edu
9. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0014140
10. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, vol. 32. Springer, Boston (2006). https://doi.org/10.1007/0-387-29151-2

Techniques for complex analysis of contemporary data

Jakub Peschel
Masaryk University, Czech Republic
E-mail: jpeschel@mail.muni.cz

Michal Batko
Masaryk University, Czech Republic
E-mail: batko@mail.muni.cz

Pavel Zezula
Masaryk University, Czech Republic
E-mail: pzezula@mail.muni.cz

Abstract. Contemporary data objects are typically complex, semi-structured, or unstructured at all. Besides, objects are also related to form a network. In such situation, data analysis requires not only the traditional attribute-based access but also access based on similarity as well as data mining operations. Though tools for such operations do exist, they usually specialise in operation and are available for specialized data structures supported by specific computer system environments. In contrary, advance analyses are obtained by application of several elementary access operations which in turn requires expert knowledge in multiple areas. In this paper, we propose a unification platform for various data analytical operators specified as a general-purpose analytical system ADAMiSS. An extensible data-mining and similarity-based set of operators over a common versatile data structure allow recursive application of heterogeneous operations, thus allowing definition of complex analytical processes, necessary to solve the contemporary analytical tasks. As a proof-of-concept, we present results that were obtained by our prototype implementation on two real-world data collections: the Twitter Higg's boson and the Kosarak datasets.

1. Introduction

The challenging feature of contemporary data is twofold. Firstly, the data objects are usually big, complex, and contain semi-structured or completely unstructured data. A simple example can be an object person, consisting of not only attributes, but also containing texts, images, and videos. Second, as these objects are usually related to other objects, links representing these relationships can also contain a lot of additional information. For a married couple, it might be the date of marriage, but also the time of their last phone contact or even content of their E-mail conversation. A lot of such data is produced by social networks, telephone, or other communication networks, but there might also be many other resources as well.

In order to exploit the information buried in such big and complex databases, similarity and link analysis data processing techniques are vitally important. As a simple motivating example, consider the telephone network. Given a specific telephone number, an analyst might be interested in other numbers which call similar sets of contacts in a pre-specified time range. Here a simple similarity search on sets can be applied to solve the problem. A more difficult task is to find groups of people who call each other, thus possibly form a community with special common interests. A discovery of inferential rules that would predict possible calls with certain reasonable probability might also be of high importance. Here the frequent item analysis can be used to solve the problems, but there are other technologies as well. In practice, several such actions need to be applied to solve a specific data analysis problem.

Obviously, a lot of technology – for similarity searching as well as the link analysis – have been developed for each such specific task, but each of the respective systems has its specific constraints and assumptions; thus their combination is technically very challenging and time-consuming. The basic idea of our proposal is to create a unifying platform, a framework, which would allow storage of such complex data and support an extensible set of similarity and link analytic operations for especially filtered data profiles, easily combined in an algebraic fashion. Specifically, we see the storage as a network of complex nodes and multiple semantic links between the nodes. Specific subsets, e.g. all phone calls in the last week, are extracted by specific filters into flat transaction files of a variable number of items, which can be seen as sets, multi-sets, or sequences – e.g. traces of consecutive phone calls. Transaction files are inputs for operations, the outputs of which can contribute to inputs of other operation, the results can also be inserted back to the storage.

In the following, we discuss the objectives and challenges to solve in Section 2. Similar systems and their differences are described in Section 3. Then we define the structure and operations of our system in Section 4. In order to illustrate the capabilities of the system, Section 5, outlines four use-cases in light of the goal, data, and processes. The functionality of the current prototype is demonstrated in Section 6 using a real-life Twitter dataset. Future trends are discussed in the Conclusions.

2. Preliminaries

The process of analysis consists of incremental steps, where different techniques are applied to data to obtain hidden information. The problem comes, when for simple task a lot of different highly specialised tools must be used. Each of these specialised tools contain their specific format, which differs from the others. Expert analyst then has to focus on ability to manipulate data into these formats instead of on the semantics of the analysis and the analysis itself.

One of the possible approaches to this problem is to create a unifying system, that will allow for specification of data analytical tasks without a need to specify used algorithms or to define transformation between individual tasks. Based on this idea, system ADAMiSS [16] was proposed.

We perceive the contribution of this paper in these points:

(i) In this paper is described an architecture of a system ADAMiSS, that will allow for easy combination of data analytical tasks into more complex analytical functions.

(ii) There is proposed basic formalisation of the system and basic data analytical tasks in context of the described architecture.

(iii) This paper provide few sample use-cases that show versatility of proposed system.

3. Related work

In the area of data analysis, there has been created a lot of different methods for effective analysing of data ranging from methods analysing statistical properties of whole datasets to

methods focusing on the analysis of local properties of the single individual data object in the dataset. Because area of the data analysis is enormous and it is not possible to comprehend everything, main focus of this system at the moment is in analysis based on similarity, like similarity search methods, clustering, similarity joining, and an analysis based on pattern mining, like frequent itemset mining, sequence mining, association rule mining and others. These areas have been chosen because they are able to work with various data types without the need for feature extraction; thus allow bigger versatility.

For each of mentioned task, there exist a high range of efficient methods. For example, frequent item-set mining task was first proposed by Agrawal et al. [1] for analysis of items bought together in markets, sometimes called market basket analysis. The proposed Apriori method for solving this task was optimised several times [2] and other methods for this task such as FP-Growth [12], Eclat [20], PPV [9], or latest NegFIN [3] were proposed.

In the area of similarity analysis, there were developed multiple techniques and indexes for efficient processing of similarity search queries. Example of such indexes are M-tree [7], M-index [15], and many others [21].

To cope with a high number of use-cases, where some techniques are better than others, libraries containing a high number of specialised techniques were created. Example of such in the area of pattern mining is library SPMF [10] which contain before mentioned algorithms. The problem is that such a library often has a different goal than to allow building complex analysis from tools, so techniques cannot be easily put one after another to create more complex behaviour. Another example of such a library is ELKI [17] which primary purpose is to provide an environment for benchmarking of clustering techniques, thus does not provide support for chaining of analytical tools.

Most similar to desired functionality are systems that we can divide into two groups. Analytical languages like R [13], that allow for performing a big range of analytical operation, but their learning curve is steep and they require a lot of focus on data manipulation. And analytical systems such as Weka [11] and Matlab [14] that are most similar to the desired result. Both mentioned languages and systems also have a restriction on the type of data, such as they process either very limited data types like numerical and textual or either require data to be transformed into vectors or matrices. This restriction is in contemporary data very limiting, because it omits relations between data objects and it is not sufficient for more complex data types newly created.

4. Architecture

In this section, an overview of the architecture of ADAMiSS is described. A brief schema can be seen in Figure 1. ADAMiSS is composed of several components and serves for storing and processing of data. The components are the following:

(i) Storage
(ii) Transaction database
(iii) Analytical processing unit

Storage

One of the main issues of the systems for data analysis is that they expect data to be imported in specific form. This require enormous data manipulation before analysis itself. To alleviate this, data are imported into ADAMiSS storage in raw form. To capture such data, storage is in the form of a multigraph $S : (V, E)$, where V is a set of data objects and E is a multi-set of edges. Each data object $v \in V$ is pair $(ID, ATTRs)$ where ID serves as a simple representation of the object. Complex data of objects are stored as attributes in map of attributes $ATTRs$ that

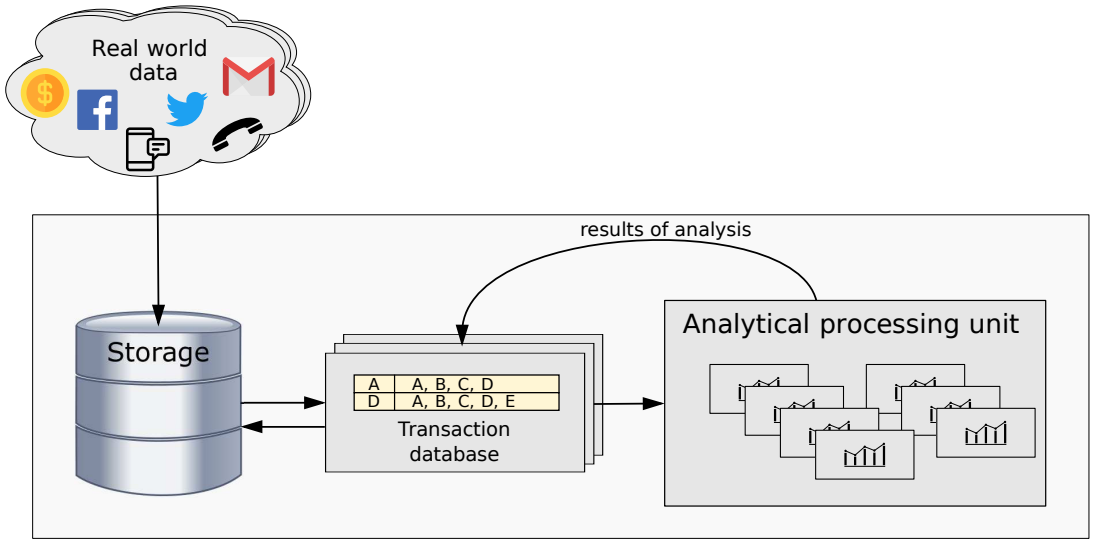


Figure 1. Overview of an architecture of ADAMiSS.

contains *key:value* pairs. Possible interactions between data are stored in edges $e \in E$. Each edge e is triplet of two data objects v_1 and v_2 and set of attributes $ATTRs$ to store additional information.

The multigraph structure with attributes as a representation of raw data was selected, because most of the data can be stored in this structure without need for transformation. There exist other structures like hypergraph[6], but the usage of attributes can simulate such structures.

An important property of this storage is, that system is capable of capturing heterogeneous structures, and process them without bigger nuisance. Let phone communication be an example; then individual phones are imported as nodes in the network. Data such as an owner of the phone and a type of phone card are stored in respective attributes. Calls between phones are captured as edges and attributes such as the duration of call and voice messages are captured. Also other data such as financial transactions or e-mail communication can be added in a similar way. A newly created heterogeneous network of data can be processed by other parts of the system based on user demands.

Transaction database

Because storage structure is too complex and providing analysis over it is in many cases expensive, ADAMiSS system contains *transaction database* structure. It is set of transactions $t : (Data, ATTRs)$. *Data* is list of transaction items $(i_1, \ldots, i_n)$, which are application-defined objects. The only restriction on these items is that the check of identity must be defined. $ATTRs$ is set of *key:value* pairs as in case of storage objects. Attributes in transactions are used for storing the result of analysis and intermediate results.

The main advantage of this flat structure is that a lot of different views on the *Data* can be used. It can be perceived either as ordered list, as multi-set, or as a set. This possibility of different views allows to process the same data by different operations; thus, it serves as a universal input into analytical operations.

Analytical processing unit

The vital part of the whole ADAMiSS system is *Analytical processing unit*. It is a collection of analytical operations that works over the transaction database and as a result, provide

transaction database. To this collection, every function that meets $F : ([TD, ...], params) \rightarrow TD$ can be added.

To follow the primary goal of analysing diverse data types, most of the methods available in ADAMiSS need to be flexible with the data structure. Both similarity-based techniques and pattern mining techniques offer the possibility of working with universal data. As an example, a list of available tasks with their respective description and usage follows:

Frequent item-set mining is a task of the discovery of frequent sets of co-occurring items. A user-defined threshold determines the frequency of a pattern. The task can be defined as a function $IM : (TD, \theta) \rightarrow \{(p, \{"support" : s_p\})\}|p \subseteq t.data; t \in TD; s_p > \theta$ where p is a set of items, s_p is support computed over the original transaction database.

Sequence mining serves for searching for frequent subsequences in the sequential data. Data in a transactions are taken as a list, and common order of data items is analysed. The task can be described as a function $SM : (TD, \theta) \rightarrow \{(p, \{"support" : s_p\})\}|p \subseteq t.data; t \in TD; s_p < \theta$, where s_p is support computed over original transaction database.

Association rule mining is a process of deriving association rules based on a Support/Confidence framework. The association rule is triplet $(p_0, p_1, conf(p_0, p_1))$ where p_0 and p_1 are patterns, and a function $conf(p_0, p_1)$ describes the probability of how often based on the occurrence of the pattern p_0 the pattern p_1 will be present. Support/Confidence framework can be replaced by other measures evaluating before mentioned probability, like Lift or Kulczynski measure [19]. Operation of mining such rules is a function $AM : (TD, p_q, \theta) \rightarrow \{(p_i, \{"confidence" : conf(p_q, p_i)\})\}|\exists t_j : t_j \in TD, p_i \subseteq t_j.data;$.

Range search is an operator that searches for all the transactions that are similar to the selected query. The user defines the threshold for similarity. In the range search, an arbitrary distance function must be selected. The operation can be seen as a function $RS_{dist} : (TD, q, r) \rightarrow \{(t_i, \{"distance" : d\})\}|T_i \in TD : dist(q, body(T_i)) < r\}$.

K-NN search serves for searching K nearest, most similar transactions in the sense of distance evaluation. Suitable distance function *dist* must be selected. Formally, K-NN search is a function: $KNNS_{dist} : (TD, q, k) \rightarrow \{(t_i, \{"distance" : dist(q, t_i.data\})\}|t_i \in TD; \forall t_j \in TD : dist(q, t_i.data) < dist(q, t_j.data)\}$. Query object q and amount of search neighbours k must be provided by user.

Similarity self-join is an operator that compares transactions between themselves and pairs them based on their similarity distance. The similarity distance is measured by the distance function *dist* provided by the user. The user also provides the similarity threshold. Similarity join is defined as $SJ_{dist} : (TD_1, TD_2, \theta) \rightarrow \{((t_1, \ldots, t_n), d)\}; t_1, \ldots, t_n \in TD_2; \forall t_q \in TD_1; dist(t_q, t_m) = d; d < \theta$.

Outlier detection is a process of finding transactions, that differs from others in the transaction database. To evaluate the difference, the similarity of the transactions is used. Suitable distance function *dist* must be selected. Function $OD_{dist} : (TD, r) \rightarrow \{(t_i, \{"min_distance" : d\})\}|t_i \in TD; d = mindist(q, t_i) : \forall t_j in TD; d > r\}$ returns objects, which differs to most similar object at least r; thus they are outliers.

This list of operation shows the flexibility of the transaction database to provide a suitable representation of data to analyse. Due to the possibility of analytical operation to return results

in the form of transaction database, it is possible to chain operation one after the other and thus to iteratively create more complex operations. The proposed architecture is able to store and analytically process data in an iterative manner.

5. Use-Cases

We designed four use-cases of how this system can be used. All of them are motivated by forensic analysis.

Grouping of people in monitored area

The task of security in the museum is to prevent exhibits from being stolen. Let us assume that a group of criminals try to steal the exhibit. They need to properly asses the security measures beforehand. This need leads to multiple visits to the secured place. Identification of such groups can be very beneficial for a further increase in security measures and for establishing early danger detection methods.

Task: Find frequently occurring groups of people in the monitored area. Example of the monitored area is a museum, bank, squares and others.

Data: Video records from a surveillance camera.

Process:

(i) To import the data, the records are imported as an entity in a graph representation. Information like location or time is stored in the attributes.

(ii) During the filtering, each frame is taken as a single image; entities are derived by techniques like face recognition. An important part is to properly determine each person's identifier to recognise them between images. Entities from each frame are then used as transaction items and a set of entities in a frame is a body of a transaction.

(iii) Frequent item-set mining is applied to the transaction database created in step 2. Reasonable threshold must be set because there is expected to be a repetition of people due to the frequency of video record.

Results: Set of groups of people that are often occurring in the monitored area. These sets can be used for further analysis. Often, there is an expectation of none of such bigger group frequently occurring in the monitored area.

Analysis of financial transactions

There exist groups of financially depending bank accounts, between which transactions are frequently created. These groups create chains frequently observable in transactions. A bank is interested in knowledge of which accounts are connected by these chains, to evaluate the amount of money available to subjects properly.

Task: Discovery of frequent chains of accounts.

Data: A list of financial transactions

Process:

(i) Each transaction is imported, and a node is created for every new endpoint. Additional information from the transaction as an account owner, time of the transaction, amount of transferred money, etc. are stored in attributes for the purpose of filtering.

(ii) Paths are derived based on user-specified condition, such as time proximity, or frequency between incoming and outgoing transaction. An example of such condition is that the new transaction must appear within 24 hours with a similar amount of money. There is created a transaction in the transaction database for each such path.

(iii) There is applied the sequence mining operator to the created transaction database. The threshold of frequency is set by the user. Both approaches with holes and without holes can be selected.

Results: Set of subsequences, that frequently occur in the data.

Search for infected computers
Some computer viruses cause infected computers to communicate over proxy servers in the computer networks to capture communication over the internet. When these proxy servers are discovered, technician search for infected computers. Because these computers heavily communicate over found proxy servers, a technician can search for computers with similar communication behaviour and analyse them further.

Task: Find computers with a high volume of communication partners consisting of found proxy servers. Such computers are similar to the abstract node that has proxy servers as neighbours.

Data: Communication logs of the network.

Process:

(i) The first step is the import of data into an ADAMiSS graph structure. Computers are imported as nodes and communication between computers as edges.

(ii) Once graph representation is created, transactions are created from nodes by collecting neighbours of the processed node as data and selecting the processed node's id as a transaction ID. Transactions for server nodes are omitted because these are not important for the selected task.

(iii) Once the transaction database is created, similarity searching operator is used with a selected transaction as a query object. The similarity coefficient is computed between transactions and query transaction.

Result: Pairs of computer and its respective similarity coefficient to the query.

Identification of multiple accounts of one user
Online games often contain rules forbidding the creation of multiple accounts. This rule is often used to limit players in amount of game resources. Often users breaking this rule interact with same groups of players across all accounts. Identification of similar accounts can lead to the identification of players breaking this rule and punishment of such behaviour.

Task: Identify accounts with the same user. We assume that the same user has similar behaviour when using different accounts.

Data: In-game account interactions.

Process:

(i) During the first step, data are imported into ADAMiSS system. Accounts are imported as nodes, interactions are added as edges.

(ii) Once graph representation is created, the system filters out unwanted nodes by specifying filters. To create a transaction database set of neighbouring nodes is taken as a data of the transaction.

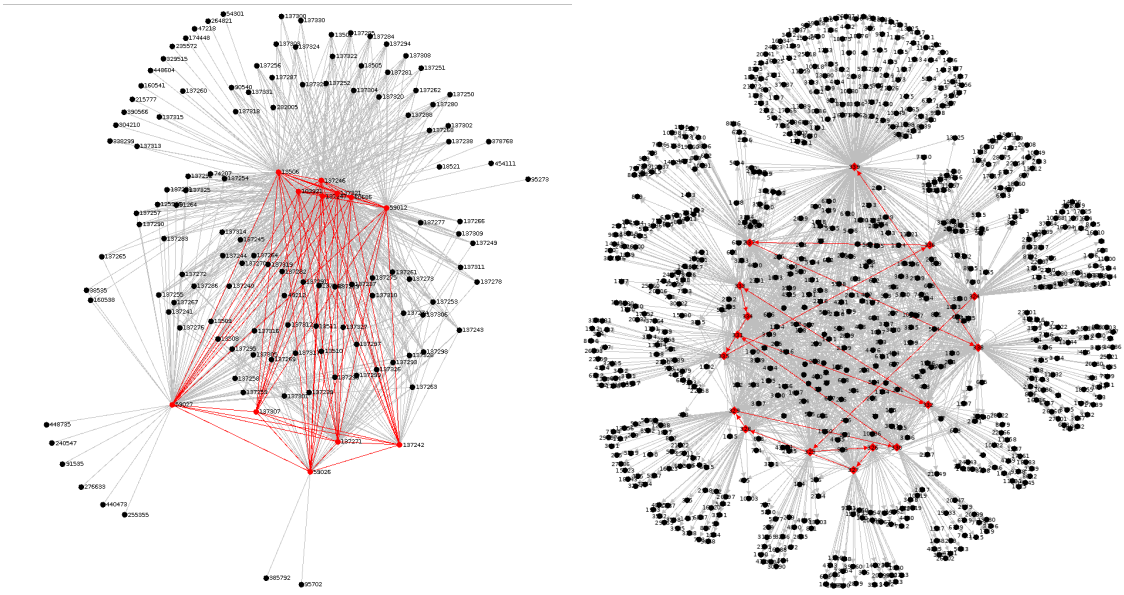
(iii) Once we have a transaction database, the self-join methods described in section 4 can be applied.

Result: Pairs of accounts with a similarity coefficient higher than the selected similarity threshold.

6. Experimental results

We developed a prototype application to show the usability of ADAMiSS. Two datasets were used, Twitter Higg's Boson dataset [8] and Kosarak dataset [5].

All experiments were performed on a server with 4 cores Intel Xeon CPUs and 16GB RAM.



(a) Community of size 12 at thr. 11 (red)

(b) Paths of length 16 at thr. 1024 (red)

Figure 2. Example of identified patterns in different datasets.

We apply frequent item-set mining operator to dataset Twitter Higg's boson. We use a slightly modified FP-Growth algorithm for mining communities instead of frequent sets. This dataset contains interactions between Twitter users with a topic connected to the discovery of Higg's boson at CERN. The dataset contains interactions such as mentioning, retweeting and replying. We transform each account and all neighbouring accounts as transactions. Transaction database consists of 304691 transactions/accounts, and the average length/degree of transaction/node was 3.96. The small degree of nodes is because many accounts have only one connection to the star hub, i.e. node with many nodes connected just to the hub node.

Task	Dataset	# Nodes	# Edges	Time
Community mining	Twitter Higg's boson	304 691	1 209 095	13m 12s
Sequence mining	Kosarak	41 270	8 019 015	52s
Sim. search with range query	Twitter Higg's boson	304 691	1 209 095	1s
Sim. search with k-nn query	Twitter Higg's boson	304 691	1 209 095	1s

Table 1. Overview of experiments performed on prototype implementation.

The goal is to obtain communities of twitter accounts that share information about the discovery of Higg's boson between themselves. We use a strict definition of community as a fully interconnected subgraph, i.e. clique. There was selected a threshold of size 11, which is quite a low number for pattern mining in such a database. Reason for small threshold is that social networks are quite shallow graphs, so there is not expected to have bigger cliques containing a higher amount of unique accounts. This assumption showed up as true because the largest cliques found are of size 12. To show the result, we picked one random community of size 12 and visualised it with neighbouring nodes in figure 2.

We discovered 7 communities of size 12 and 94 communities of size 11. Average overlap of all these communities is 80.7%. Communities were further analysed if they contain a common subset, but none such subset was found.

For sequence mining, ADAMiSS contains modified GSP algorithm [18] that does not allow gaps. We used Kosarak dataset for an experiment. This dataset contains 990002 click-stream data from the logs of an online news portal. Click-stream is a sequence of browsed web pages by one user on the web page domain. We use sequences of click-stream as paths of users. The threshold was set to 1024. A result contains 322 paths and just 5 paths that contain more than 4 nodes. The longest path contains 16 nodes. Sub-paths of found paths were not taken into account. The longest path is visualised with the corresponding neighbourhood of nodes on the path in figure 2.

We use one selected community found by dataset Twitter Higg's boson for a test of the similarity searching. One of the communities of size 12 was randomly picked, and for each node, we search for similar nodes. The similarity of nodes is defined as the similarity of neighbouring sets. Jaccard coefficient was selected as a distance function. ADAMiSS uses Messif framework developed by Batko et al [4] for efficient processing.

For range queries, the threshold was selected to 0.2 that mean 80% of the set is shared. Four nodes have their most similar items inside the community. One node (137328) has six nodes most similar to itself outside of the community. Other nodes did not have sufficiently similar nodes.

We selected 10 as k for k-nn queries, and an average amount of most similar nodes from the community is 8.33 nodes. One node (137328) has all ten most similar nodes outside of the selected community.

Underlying algorithm used for evaluation of both range query and k-nn query was M-index [15].

7. Conclusion

In the light of high volumes of contemporary data and their unstructured nature, automatic processes for better searching and analysis are required. The amount and the complexity of tools cause needs for expensive experts in the area and extensive processing of the data.

In this paper, we introduced an idea of analysis based on the unified structure, that allows reducing the need for transformation of the data in between individual steps of analysis. We described a system called ADAMiSS that stores and process data in a unified structure called a

transaction database. This system allows to reduce the amount of necessary transformation and allow to create more complex analytical constructs based on techniques of similarity analysis and pattern mining applicable universally to a high range of data.

A prototype of the ADAMiSS was developed, and it was applied for the analysis of two datasets, Twitter Higg's Boson dataset and Kosarak dataset. The system showed as usable for selected tasks of obtaining new information. The system processed data and provided information about communities, frequent paths through the network and similarities between nodes. The same approach, as shown in the experimental part, can be used for described use-cases in section 5, such as analysis of financial transactions and for analysis of public groups in social networks.

This system opens space for further analysis of more complex analytical constructs created from the blocks of operation. In the future, there is a need for further formalisation of the process of analysis into a query language based on this concept.

8. Bibliography

[1] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Acm sigmod record. vol. 22, pp. 207–216. ACM (1993)

[2] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I., et al.: Fast discovery of association rules. Advances in knowledge discovery and data mining **12**(1), 307–328 (1996)

[3] Aryabarzan, N., Minaei-Bidgoli, B., Teshnehlab, M.: negfin: An efficient algorithm for fast mining frequent itemsets. Expert Systems with Applications **105**, 129–143 (2018)

[4] Batko, M., Novak, D., Zezula, P.: Messif: Metric similarity search implementation framework. In: Digital Libraries: Research and Development, pp. 1–10. Springer (2007)

[5] Bodon, F.: A fast apriori implementation. In: FIMI. vol. 3, p. 63 (2003)

[6] Bretto, A.: Hypergraph theory. An introduction. Mathematical Engineering. Cham: Springer (2013)

[7] Ciaccia, P., Patella, M., Zezula, P.: M-tree: An e cient access method for similarity search in metric spaces. In: Proceedings of the 23rd VLDB conference, Athens, Greece. pp. 426–435. Citeseer (1997)

[8] De Domenico, M., Lima, A., Mougel, P., Musolesi, M.: The anatomy of a scientific rumor. Scientific reports **3**, 2980 (2013)

[9] Deng, Z., Wang, Z.: A new fast vertical method for mining frequent patterns. International Journal of Computational Intelligence Systems **3**(6), 733–744 (2010)

[10] Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 36–40. Springer (2016)

[11] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter **11**(1), 10–18 (2009)

[12] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM sigmod record. vol. 29, pp. 1–12. ACM (2000)

[13] Hornik, K.: R FAQ (2018), `https://CRAN.R-project.org/doc/FAQ/R-FAQ.html`

[14] MathWorks, I.: Matlab and statistics toolbox release 2012b. Natick (Massachusetts, United States) (2012)

[15] Novak, D., Batko, M., Zezula, P.: Metric index: An efficient and scalable solution for precise and approximate similarity search. Information Systems **36**(4), 721–733 (2011)

[16] Peschel, J., Zezula, P.: ADAMiSS: Advanced data analysis, mining and search, system. In: International Conference on Similarity Search and Applications. pp. 351–355. Springer (2019)

[17] Schubert, E., Zimek, A.: ELKI: A large open-source library for data analysis - ELKI release 0.7.5 "heidelberg". CoRR **abs/1902.03616** (2019), `http://arxiv.org/abs/1902.03616`

[18] Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: International Conference on Extending Database Technology. pp. 1–17. Springer (1996)

[19] Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework. Data Mining and Knowledge Discovery **21**(3), 371–397 (2010)

[20] Zaki, M.J.: Scalable algorithms for association mining. IEEE transactions on knowledge and data engineering **12**(3), 372–390 (2000)

[21] Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity search: the metric space approach, vol. 32. Springer Science & Business Media (2006)