**Final Project**

1. *Requirement*: Consider a data set of your choice: e.g. from Kaggle or from any other source (e.g. job or your master thesis).

For this task I have chosen the “All laptops with Specifications Dataset” (<https://www.kaggle.com/datasets/durgeshrao9993/laptop-specification-dataset>) containing several information about the hardware specification of various laptops on the market.

***Motivation***

Buying a new laptop can be an important financial and practical decision due to its many personal and job related use cases in our daily life. But the reality is that most people looking to buy a new laptop are not familiar enough with all the hardware details that can make a big difference between a successful acquisition that can last years, and a disappointment. As such, I think that by using FCA on this dataset about laptops we might get a stronger understanding about how various hardware components are related, which in turn should make the acquisition process easier.

1. *Requirement*: If necessary, apply some AI, KD and Data Mining algorithms in order to extract a first layer of knowledge. Describe the results you have obtained and the knowledge you have extracted.

The raw version of the dataset contains several columns, each with a large variety of values, therefore, it needs some cleaning. I have chosen some representative columns such as **company, CPU, RAM, Memory and GPU**. In my opinion, these hardware pieces along with the brand should be the main factors to take into consideration when choosing a laptop.

First of all, for each field, I choose the most representative values, discarding the ones that are not frequent enough.

The company field has the following values, along with their frequencies: Acer (23), Apple (5), Asus (33), Chuwi (2), Dell (29), Fujitsu (3), Google (2), HP (36), Huawei (1), LG (2), Lenovo (36), MSI (5), Mediacom (4), Microsoft (3), Razer (4), Samsung (6), Toshiba (9), Vero (2), Xiaomi (2).

The CPU field has the following values: amd (27), intel (180).

The RAM field has the following values (GB): 2 (9), 4 (47), 6 (21), 8 (64), 12 (16), 16 (37), 24 (3), 32 (9), 64 (1).

The memory type field has the following values: HDD (63), SSD (85), SSD + HDD (36), flash (23).

The GPU field has the following values: AMD (49), Intel (88), Nvidia (70).

In the end, the processed dataset has **207** objects.

1. *Requirement*: Use either the result of this analysis or any other many-valued context to build a ToscanaJ system. Describe in detail the scales you have built and the amount of knowledge you was able to extract here from. Be inventive and creative in the scale building, usually the knowledge gems are there, but you need to dig after them. For ToscanaJ you will need an older version of Java to make it work.

For this task I have used both **ELBA and ToscanaJ** in order to extract knowledge. ELBA was useful in designing the contexts and scales by using **attribute list, nominal and ordinal scales**. This process was eased by the capacity of ELBA to connect and read my custom dataset from an sql file which I have generated using an online csv to sql tool (<https://www.convertcsv.com/csv-to-sql.htm?fbclid=IwAR1QEPb9QfeDWG4APvfzit2MSq6agpxtfQe_6mn0eEUa87psX3_jL3rBUdg>). The output sql looks like this:

CREATE TABLE mytable(

id INTEGER NOT NULL PRIMARY KEY

,company VARCHAR(9) NOT NULL

,cpu VARCHAR(5) NOT NULL

,ram INTEGER NOT NULL

,memory VARCHAR(9) NOT NULL

,gpu VARCHAR(6) NOT NULL

);

INSERT INTO mytable(id,company,cpu,ram,memory,gpu) VALUES (0,'Apple','intel',8,'SSD','Intel');

INSERT INTO mytable(id,company,cpu,ram,memory,gpu) VALUES (1,'Apple','intel',8,'flash','Intel');

…

After loading the data into ELBA, I was able to generate conceptual schema (csx) files which are in turn used by ToscanaJ to visualize the concept lattices, with more details about where exactly the objects are located. In ToscanaJ I used the “**Show exact matches**” option from the view panel.

The next step is the knowledge extraction. In order to have a better understanding of the dataset, I have created several scales and diagrams as follows.

First of all, I wanted to get a better understanding of the dataset proportions with respect to each column. I started with RAM and I used the ordinal scale option:

A screenshot of a computer

Description automatically generated with medium confidence

Afterwards, I visualized it in ToscanaJ:

A picture containing line, screenshot

Description automatically generated

It seems clear that most laptops (68.12%) have less than 8GB of RAM which is nowadays enough for usual activities such as browsing the internet or using Microsoft Office, but not quite enough for more demanding activities such as playing games or training machine learning models. This is reflected by the low proportions of laptops which have RAM ranging from 8 to 16 GB (25.6%) and even lower proportion of laptops with a lot of RAM (6.28%). Considering what most people need from a laptop, these proportions make sense.

The next element from which I tried to extract knowledge is the CPU:

A picture containing diagram, screenshot, line, circle

Description automatically generated

From the diagram it is clear that the most popular option are the CPUs from Intel which is a well established producer in the market.

The next component might not be that important for basic usage of the latop. The GPU is usually important to those that need heavy algebraic computations. This is usually needed for computer games or machine learning. The diagram is the following:

A picture containing line, diagram, circle

Description automatically generated

Here Intel is also leading in popularity, probably because it is a cheaper option, suitable for basic usage of the laptop. Following closely, there is Nvidia with a reasonably high percentage of 33.82%. This is the kind of GPU needed for machine learning and game playing.

I also analyzed the memory component. This is important because it decides how fast applications are loading on the laptop:

A picture containing line, diagram, circle

Description automatically generated

We can see that there are mostly two types: SSD and HDD, with the latter being an older technology, but cheaper. Laptops also may have both an SSD and an HDD. Flash memory is similar to SSD but smaller in size, which is why it is not that popular.

A diagram of a network

Description automatically generated with low confidenceThe next component is not related to hardware, but it is the company producing the laptop. From this diagram I can tell the variety of options offered by each brand:

Now that I have covered the basic components, I want to see what knowledge I can extract such that I can better understand laptops suitable for machine learning.

A picture containing line, diagram

Description automatically generatedFirst of all, it needs to have a Nvidia GPU. In the next diagram I have illustrated some of the most popular brands and whether they have a Nvidia GPU or not:

It seems that there are no Apple laptops with Nvidia GPU, therefore they are not suitable for this. Also, Asus seems to lead with 6.36% of laptops with Nvidia GPU, which means it offers more variety with a Nvidia GPU.

Finally, I wanted to have the bigger picture. I want to understand all components together. This ideally will help in deciding which components are worth it, which is especially important when buying a laptop on a budget.

Therefore, I have chosen the following attributes: if it has SSD, Intel CPU and Nvidia GPU. Also, I differentiated between RAM larger than 8GB, 16GB and 32GB.

A picture containing diagram, line

Description automatically generated

Also, this is the context:

A screenshot of a computer

Description automatically generated with medium confidence

And here we have the same diagram, but showing the list of objects for each concept:

A picture containing diagram, plan, line, technical drawing

Description automatically generated

Finally, it seems that the laptop brands having the best components for machine learning are: Asus, Dell, Lenovo, MSI and Razer. But depending on the budget, it seems that there are options for all combinations of hardware components.

1. *Requirement*: Choose a set of attributes from any source you are comfortable with (master thesis, job, computer science, etc.) Perform attribute exploration on this set of attributes and collect the relevant knowledge from experts in that field. Explain your results and what you have learned here from.

For this task I have used the Concept Explorer tool: <https://conexp.sourceforge.net/index.html>. The aim is to start with a set of known rules and samples and then explore the unsolved questions. The results of this exploration are either new samples or new rules.

A screenshot of a computer

Description automatically generated with low confidenceAs before, I sampled a subset of the laptop dataset. I am more interested in laptops with Intel CPU, Nvidia GPU, SSD, and at least 8GB of RAM, therefore these are the attributes. For the objects, I have chosen 2 random samples from each brand. Using a python script, I created the following context:

For the next step, I have used the concept explorer tool to perform attribute exploration, by answering the questions raised by the tool. In order to decide whether or not the implication is true, I checked if the dataset contains such an example.

A screenshot of a computer error

Description automatically generated with low confidence

For the above question, I have found the counterexample and I introduced it as follows after pressing ‘no’:

A screenshot of a computer

Description automatically generated with medium confidence

For the next question I could not find a counterexample, as all laptops with an Nvidia GPU seemed to have Intel CPU and more than 8GB of ram, therefore I pressed yes.

A screenshot of a computer error

Description automatically generated with medium confidence

After this questions, the process finished:

A screenshot of a computer error

Description automatically generated with medium confidence

Finally, I found one new example, and the context after exploration looks like this:

A screenshot of a computer

Description automatically generated with low confidence

After attribute exploration, I also computed the set of implications, which confirms what I have found in the dataset, that all laptops with Nvidia GPU have an Intel CPU with at least 8GB of RAM:

A picture containing text, line, screenshot, software

Description automatically generated

I also computed the association rules, which give the probability that some implication is true:

A screenshot of a computer program

Description automatically generated with medium confidence

Finally, I have found out some interesting aspects about the dataset. For instance, most laptops (96%) that have a SSD, also have an Intel CPU, similarly 86% of laptops with SSD also have at least 8GB of RAM. Also, it seems that only 33% of laptops with Intel CPU and at least 8GB of RAM have an Nvidia GPU.

1. Choose a triadic data set and perform knowledge discovery as discussed at the course.

For this task I have chosen the Data Science Salaries 2023 dataset, available at: <https://www.kaggle.com/datasets/arnabchaki/data-science-salaries-2023>. I consider this relevant because it is important at any time during one’s career to know if the salary is right. Since this can be quite complicated, I consider that FCA might help in understanding the current market.

As such, I chose a subset from the dataset to contain only the job title, position (junior, mid, senior and director), the company size, if remote working is an option, and the salary split in 3 categories: low, medium, high.

The dataset is also available here: <https://fca-tools-bundle.com/view-context/64809a94ef7188e866e399ac> under the name “Data Science Salaries 2023”.

After some more processing, I generated the context.csv file which I then uploaded on the website. The idea is to obtain jobs titles as objects, company size and salary, and remote status as attributes and job level from junior to senior as the conditions. In this way we can analyze jobs at various levels of experience.

For instance, from the following we can tell that junior ML scientists have low salaries in small companies but can work remotely.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

1. There are various significant applications of FCA on real life data and projects. Write a report about one selected application.
2. Introduction

Software engineering involves the development, maintenance, and evolution of complex software systems. Effective knowledge representation and analysis play a crucial role in managing software artifacts and improving software quality. Formal Concept Analysis (FCA) offers a valuable methodology for organizing, analyzing, and reasoning about software-related knowledge and relationships.

1. FCA and Software Engineering

FCA is a mathematical framework that has gained significant attention and utility in the field of software engineering. As software systems become increasingly complex and interconnected, the need for effective knowledge representation and analysis has become crucial. FCA offers a systematic approach to organizing and analyzing software-related knowledge, enabling software engineers to make informed decisions and improve software quality at any point in the software lifecycle.

Software engineering encompasses various activities such as software design, development, testing, and maintenance. These activities involve managing and understanding complex software artifacts, including classes, methods, modules, and their relationships. FCA provides a formal methodology for representing these artifacts and their interdependencies, allowing software engineers to gain valuable insights into the structure, behavior, and quality of software systems.

One of the key benefits of FCA in software engineering is its ability to facilitate knowledge representation and conceptual hierarchies. By constructing a formal context, consisting of objects (software components) and attributes (properties or characteristics), FCA allows for the organization of software artifacts into taxonomies or ontologies. These hierarchical structures provide a clear representation of the software system, enabling software engineers to navigate, understand, and communicate about the system's architecture, dependencies, and functionalities.

Software maintenance and evolution are critical aspects of the software engineering lifecycle. FCA plays a crucial role in these activities by helping to identify dependencies between software components and assess the impact of changes. By analyzing the relationships between formal concepts, software engineers can efficiently perform tasks such as bug fixing, module reusability, and system enhancement. FCA-based analysis facilitates effective software maintenance, ensuring that changes and updates are applied in a structured manner without introducing unintended side effects.

In the realm of software testing, FCA offers valuable insights and approaches. Test cases, requirements, and software components can be analyzed using FCA to identify dependencies and coverage gaps. By identifying formal concepts within the formal context, software engineers can create comprehensive test suites that cover different concepts, ensuring thorough testing coverage and reducing redundancy. FCA-based analysis aids in improving the effectiveness and efficiency of software testing, ultimately leading to higher software quality and reliability.

Collaboration and communication among software engineering teams are essential for successful project outcomes. FCA serves as a common language and framework for discussing and understanding software artifacts and their relationships. By representing software artifacts using formal concepts, FCA facilitates better communication and knowledge sharing among team members. It enables a shared understanding of the software system, promotes effective collaboration, and enhances decision-making processes, leading to improved software development outcomes.

1. Example: Analyzing application use cases

In the initial phase, the software engineers construct a formal context for the use cases using FCA. The objects in this context represent different software components, such as modules, classes, or functionalities, while the attributes represent various required use cases. Each object is associated with its corresponding attributes based on the defined criteria.

For instance, a more detailed example is described in the paper by Hesse and Tilley [], where the use cases of a software application for a wine trading company are analyzed using FCA. Firstly, an analyst distills the requirements for the use cases written in natural language and a formal context is produced:

A picture containing text, screenshot, parallel, receipt

Description automatically generated

Further on, based on this context, a concept lattice is generated:

A picture containing diagram, line, origami

Description automatically generated

Having this concept lattice, the process of identifying which classes or modules are correlated with which actual use cases. The immediate consequence is that it is much clearer to decide the development steps. This way the process of scheduling the development activities is simplified. For instance, the objects on the lower part of the diagram such as the Product or detailed order item are important for more use cases, therefore a possible schedule could be to develop the software components bottom up.

Also, by going on various concepts, it is easier to split the tasks between different teams which can increase efficiency in many aspects from communication to actual technical development.

1. Conclusion

In conclusion, FCA has emerged as a valuable tool for knowledge representation and analysis in software engineering. Its ability to organize software artifacts, identify dependencies, and facilitate software designing, developing maintenance, testing, and collaboration makes it a powerful asset for software development teams. By leveraging FCA, software engineers can gain valuable insights, improve software quality, achieve faster development processes, and enhance collaboration, ultimately leading to the delivery of high-quality software systems.