university of groningen / faculty of science and engineering

Identifying Underlying Skills in Math Problems

A Data-Driven Approach

Author
Karlijn Rozestraten
S4174984

Internal Supervisor
Prof. Dr. N.A. Taatgen
Department of Artificial Intelligence
University of Groningen, The Netherlands

Co-Supervisor
Prof. Dr. D.H. van Rijn
Department of Experimental Psychology, and
Behavioral and Cognitive Neurosciences
University of Groningen, The Netherlands

Master's Thesis
Computational Cognitive Science
Department of Artificial Intelligence
University of Groningen, The Netherlands

December 11, 2021

Abstract

Over the years, various intelligent, personalised, data-driven and cognitive tutoring systems have been developed to aid in teaching and tracking of study progress of students. These systems can make use of a cognitive model that is designed for a task that a student is expected to learn. Cognitive models require predetermined skills that are usually based on what the designer of the model assumes are needed for the task. To better future development of cognitive tutors, this project aimed to develop an unsupervised machine learning algorithm that can determine underlying skills required for math problems from the accuracy scores of exam data. The developed knowledge graph algorithm calculates a partial ordering on a set questions in a data set with multiple mathematical topics. The algorithm determines relations between pairs of questions that later forms a hierarchy of the questions. A relation represents that a question at a higher level in the hierarchy contained a subset of skills that is required for a question at the lower level. The hierarchy of the questions is visualised in a directed acyclic graph. The algorithm is validated on a simulated data set of which the possible skill combinations known. It is then applied to multiple existing mathematical exam data sets. Although the algorithm is able to discover hierarchical relationships between questions, it is not designed to determine the required skills for solving the math problems from the data. The results of this project bring us a step closer to accomplishing an objective determination of skills required in math problem-solving tests. Nonetheless, more research has to be performed to get closer to identifying exact skills required for solving math problems from exam data.

Contents

1 Introduction

The growth and adoption of educational technology gained momentum since the start of the COVID-19 pandemic (Li & Lalani, 2020; Fleming, 2021). The pandemic forced many educational institutes to switch from on-site education to fully online teaching. Currently, more than 1.5 years after the beginning of these rushed efforts, multiple disadvantages of online teaching have been observed. One of such disadvantages is the lack the possibility of monitoring student activity and behavior. However, the most striking obstacles are those regarding knowledge assessment. The risk of fraud during online exams is high and many educational institutes have introduced regulations and software to enable supervision during exams. Yet another issue faced by both staff and students during online examinations is that of software and hardware malfunction. For example, in the past year, many exams had to cancelled in the Netherlands due to malfunction (Trouw, 2020; NOS, 2020).

Currently, the majority of the education and exams have resumed in their classic, on-site form. However, further lockdowns are likely to happen again in the future. Therefore, alternative and improved modes of knowledge assessment should be investigated to avoid making the same mistakes. In most educational areas, it is common to use testing and examination for the assessment of knowledge a student has acquired during or at the end of a course. In the ideal situation, there would exist a system that allows the students to study the materials for the course, and then track the progress. The teacher would then be able to track this progress which would make it unnecessary to distribute a test.

Over the years, various intelligent, personalised, data-driven and cognitive tutoring systems have been developed to aid in teaching and the tracking of study progress (Anderson, Corbett, Koedinger, & Pelletier, 1995; Van Rijn, van Maanen, & van Woudenberg, 2009; Klinkenberg, Straatemeier, & van der Maas, 2011). Some tutoring systems use a so called data-driven approach which implies that the system uses answers to questions in the system to further develop the system. For example, Math Garden (Klinkenberg et al., 2011) is a tutoring system that aims that the students will always have a correctness score of 75% to keep the students motivated. This is accomplished estimating the item difficulty scores as well as the performance scores of students. The estimation is based on the Elo rating systems developed for the rating of chess players (Elo, 1978) (as cited by Klinkenberg et al. (2011)). If a problem is often solved correctly by multiple students, the item difficulty score decreases and the student performance score increases. These scores are then used to assign a mixture of difficult and easy questions to the student.

Other developments of tutoring systems are cognitive tutoring systems that use cognitive models. To develop cognitive tutors, it is necessary to know what skills students need to perform the a task. Skills are sets of knowledge that are task-independent and can be reused for other tasks (Hoekstra, Martens, & Taatgen, 2020). Current cognitive tutors are developed based on the skills the designer assumes a student needs for a certain task (Anderson et al., 1995). The designer of a cognitive tutor has to determine and build these skills into the cognitive model (Ritter, Anderson, Koedinger, & Corbett, 2007), which is time-consuming. In addition, skill determination is difficult because there may be underlying skills required for a task that the designer is not aware of. It is essential to build the right skills into a cognitive model to ensure that the cognitive tutor solves the task as similarly as possible to humans. If this is accomplished, the cognitive tutor can provide better, personalised feedback to the student.

To ensure all required skills are included in the cognitive tutor, it would be beneficial to develop a

method for extracting skills necessary for solving a problem from exam answers. This approach differs from previously developed cognitive tutors (Anderson et al., 1995; Anderson, 1983; Ritter et al., 2007; Anderson, 2009) and has not been explored in previous research efforts. The key difference between the classical approach and the suggested approach is the method used for skill extraction. The classical approach the designer of a cognitive tutor subjectively extracts skills, while this research proposes an alternative objective approach of skill extraction. This study aims to uncover underlying skills required for answering exam questions, in particular math problems. The identification was done by applying an unsupervised machine learning algorithm on accuracy scores from math problem exams. An unsupervised machine learning algorithm will be used due to lack of information about the underlying skills. The reason why only mathematical exam data was used is that the skills required for solving math problems are more tangible than those used for problem solving in theoretical subjects. For example, solving the math problem: $2 + 1 = 3$, requires a student to master the arithmetic skills of addition and counting (Dendane & Math, 2009).

To tackle the problem of objective skill extraction, this thesis will address the following research question:

"How can skills required for solving mathematical problems be determined by a machine learning algorithm which only uses the accuracy scores from the mathematical exam data?"

To answer this research question, a machine learning algorithm will be developed. It is unknown what type of data is required for the algorithm to uncover skills from it. Therefore, a simulated data set will be created that can be adjusted into the desired format. Next, the algorithm can be applied to the simulated data set. The algorithm can be adjusted based on the results. After these steps, it will be known what type of data is required for the algorithm. Then the algorithm can be applied to real data sets and the results can be analysed.

2 Theoretical Framework

2.1 Cognitive Tutors

The category of cognitive tutors that will be highlighted in this project is defined by Anderson et al. (1995) as a '*type of computer-based instructional technology*'. The authors further state that a cognitive tutor in this category should be designed with a reference to a cognitive model that is designed for a specific task a student is expected to learn. A cognitive model is a computational model that is able to solve problems that are given to a student in a similar way to how a student is expected to solve these problems. The early stage of cognitive tutor development was based on the ACT* theory (Anderson, 1983), in which a skill was assumed to consist of hierarchical units of goal-related knowledge that was decomposed into sub-goals that could be reused. Later versions of ACT-R, the extension to the ACT* theory, abandoned this assumption. In these applications the sub-goal mechanism was omitted and not replaced by another assumption. Instead of creating sub-goals for a goal, separate goals were created (Anderson & Corbett, 1995; Ritter et al., 2007).

An experiment performed by Anderson (2009) illustrates how cognitive models struggle to simulate humans in the application of a combination of skills while completing a task. In this experiment participants were asked to solve several math problems. Cognitive models were developed that aimed to simulate humans in solving these problems. The results of the experiment showed that the approach applied by the participants for a particular problem in the experiment could not be simulated by the cognitive model. The math problem that was addressed is the 'pyramid problem' e.g., 5 $ 2. In such problems, digit noted on the left side of the dollar sign is the 'base' (B) and digit noted on the right side of the dollar sign represents the 'height' (H). To solve this problem participants had to start from B and perform an addition of H digits where each digit was decreased by 1 from B. Below three pyramid problems are presented to illustrate how these problems should be solved:

$$5\ \$\ 2 = 5+4+3 = 12$$
$$10\ \$\ 4 = 10+9+8+7+6 = 40$$
$$5\ \$\ 7 = 5+4+3+2+1+0+-1+-2 = 12$$

The authors created two models that attempted to solve the pyramid problems. The participants and the models were both able to solve the easier problems with low numbers (< 20). The difference between participants and the models becomes apparent at more complex versions of the problem, such as:

$$1000\ \$\ 2000$$

In the experiment, the students were all able to solve the first seven low number pyramid problems with a similar iterative addition approach. Theoretically, the same approach could be used for the '1000 $ 2000' problem. However, once faced with an addition of large volumes of numbers, students were observed to apply alternative approaches. Some of the participants mentioned that there should be an easier way to solve this. Others compared the problem to problems with lower numbers but a similar distance:

$$2\ \$\ 4 = 2+1+0+-1+-2 = 0$$
$$1\ \$\ 2 = 1+0+-1 = 0$$
$$1000\ \$\ 2000 = 0$$

Both cognitive models were able to solve the problem, but the models themselves were not ideal representations of human processing. The first model was able to solve the '1000 $ 2000' problem in approximately the same time as the median time of the participants. However, the model was based on handcrafted information in the declarative memory which means that the model rather used information from the declarative memory to solve the problem instead of by applying a skill. The second model solved the problem in a much longer time compared to the participants and the first model. This is because the model performs the iterative addition for each problem, meaning that the model calculated the '1000 $ 2000' problem by performing the additive iteration 2000 times.

The authors tried to improve the second model by letting it recognize problems like the '1000 $ 2000' problem by checking if the problem belongs to the parent problem '2 $ 4'. Although this improvement might have helped the cognitive model to solve the problem faster, it still did not apply the same reasoning as human subjects. Contrary to the models, human subjects use a set of skills and recognize in arithmetic problems with large numbers that there could be an easier approach to solve the problem than what is thought of first.

The pyramid problem highlights an important issue that cognitive tutors faced. Namely, cognitive models often focus on very specific tasks. Therefore, often times they are not a good representation of the entire human processing. Studies like the one from Anderson (2009) aided in the understanding of how humans acquire and apply specific skills. However, cognitive model do not capture the nature of human skill development well. For example, humans exhibit mathematical abilities as early as in infancy. These abilities are a basis for what later becomes more complex in mathematical skills (Anderson, 2009). Therefore, a method that allows objectively uncovering skills from mathematical exam data could be extremely useful in the development of cognitive tutors to better simulate human reasoning.

2.2 Digital Personalised Learning Environments and Total Ordering

In the introduction section, the digital personalised learning environment Math Garden (Klinkenberg et al., 2011) was briefly mentioned. As explained, this data-driven tutoring system determines the item difficulty score with a method that is based on the Elo rating system (Elo, 1978). To determine the difficulty score, the answers of all students to all the questions are compared to determine the difficulty. In mathematics, specifically in the order theory, this comparison is the calculation of a total ordering (Russell, 1901). An advantage of calculating a total ordering to questions compared to the development of a cognitive tutor is that a total ordering is independent of the designer because the item difficulty is determined automatically based on the performance of the students. Furthermore, in contrast to cognitive tutors, the skills required to answer the questions do not have to be known.

On the other hand, a total ordering as applied in Math Garden also has some disadvantages. First of all, a total ordering does not allow to extract underlying skills or to deduce knowledge from it. The ordering is based on the difficulty of a question rather than knowledge or skill required to answer it. A second disadvantage is that a total ordering requires similar questions so they are all comparable (Fuchs,

1963). Because of this requirement, problems based on different mathematical topics can not be grouped together in tutoring systems that calculate a total ordering.

The addressed issue regarding the development of cognitive tutors in the previous section, show that it would be beneficial to improve the method. In an ideal situation, cognitive tutors could be developed using predefined skills extracted from a personalised digital learning environment like Math Garden, where multiple different mathematical topics can be accessed.

2.3 Partial Ordering and Directed Acyclic Graphs

An alternative approach to calculating a total ordering, that can alleviate all the issues mentioned in the previous section, is the calculation of a so called partial ordering (Russell, 1901). In this project, an algorithm will be developed that does allow the comparison of assignments from different mathematical topics. In order theory, this approach is referred to as a partial ordering. By calculating a partial ordering on assignments from different mathematical topics, it is expected that groups of assignments can be discovered, each representing a topic. Additionally, within each such group, the assignments are ordered based on their difficulty.

The hierarchy as determined by calculation a partial ordering, a partially ordered set, can be visualised by a directed acyclic graph (Christofides, 1975; Thulasiraman & Swamy, 1992). A directed acyclic graph represents the objects within the set as well as the relationship between them. An example of a directed acyclic graph is illustrated in Figure 1. A graph (G) consists of a nonempty set of nodes (V) that are connected by a set of edges (E), where set V and E are defined as a collection of unique objects (Wilson, 1979). In a directed acyclic graph, the set of edges can be represented in the form of pairs of nodes. For example, edge $\{0,1\}$ as illustrated in Figure 1 demonstrates an edge from node 0 (tail) to node 1 (head).

The partial ordering that the developed algorithm will calculate on a set of question, should determine whether a question contains a subset of skills required for answering another question in the set. This relation between questions can be visualised in a directed acyclic graph as illustrated in Figure 1. The

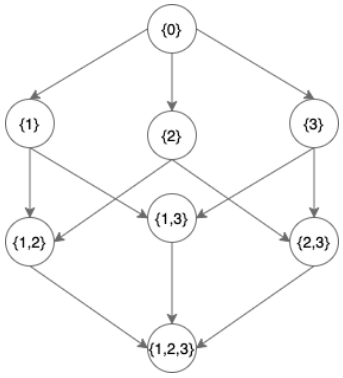


Figure 1: Example of directed acyclic graph (G). The circles represent the node-set (V): {{0}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}, which can represent skill sets. The lines with arrows at the end, represent the directed edge-set (E): {{0-1}, {0-2}, {0-3}, {1-1,2}}, {1-1,3}, {2-1,2}, {2-2,3}, {1,2-1,2,3}, {1,3-1,2,3}, {2,3-1,2,3}}.

question that contains a subset of skills required for answering another question, is the node from where the edge is drawn (tail). The question that requires the subset of skills receives the edge (head). The challenge of the algorithm is that the partial ordering has to be calculated without knowing the required skills for answering the set of questions.

2.4 Clustering Methods

The algorithm that will be developed should be applicable to a data set containing questions and answers by students to these questions. However, it would be beneficial to cluster the data set before applying the algorithm, preferably on both the questions and answers. The reason why it would be valuable to cluster the questions is because similar questions that share the same topic will be grouped together. These groups of questions could then represent a skill or skill set.

The suggestion to cluster the answers of the students into groups aids in uncovering minor differences between small groups students. Without clustering, these between-subjects differences would not be as visible and therefore could not be used to tailor the learning experience to individual student. For example, when a group of four students answer almost identically to all of the questions in a data set of 1000 students the behaviour of these four students would not be visible without clustering. With applying clustering to the students, groups of students will be created wherein the students with similar behaviour are grouped together. This could result in e.g., 50 groups of students of which one of these groups is that group of four students.

Two different types of clustering methods that could be applied are K-Means clustering (Lloyd, 1957, 1982; MacQueen, 1967) and agglomerative hierarchical clustering (McQuitty, 1957, 1961). Both algorithms are unsupervised machine learning clustering algorithms (Hinton & Sejnowski, 1999) that cluster unlabelled nor categorised data points in a data set. From all clustering methods, the K-Means clustering algorithm (Lloyd, 1957, 1982; MacQueen, 1967) is the simplest and most commonly used algorithm (Rokach & Maimon, 2005). According to the authors, the algorithm groups data points into a prior determined number of clusters (K). A data point is assigned to the cluster of which the cluster centre is nearest to the data point. The authors further explain that the initial set of cluster centres are chosen at random or with a heuristic procedure, then the data points are assigned to the nearest cluster centres. The algorithm iterates N times (predetermined), after each iteration the cluster centres are recalculated as the mean from the assigned data points to the cluster. Next, a new iteration starts where the data point is again assigned to the nearest cluster centre. The distance of a data point between a cluster centre is determined by a distance metric like the Euclidean distance or Manhattan distance.

Although the K-Means algorithm has a relative quick running time and is not computer-intensive, there are some disadvantages to using the algorithm in this project. First of all, the number of clusters has to be determined before applying the algorithm. A second disadvantage is that the algorithm randomly picks K-points or K-rows from the data to start the clustering iteration. This approach could result in starting points where the points are close to each other which results in unequally divided cluster centres. Furthermore, reproduction of the same results is difficult because the starting points are randomly selected at the start of the iteration (Boehmke & Greenwell, 2019). A final disadvantage of the K-Means algorithm is that it cannot deal with missing values in the data.

The other method, the agglomerative hierarchical clustering algorithm (McQuitty, 1957, 1961) is a

bottom-up algorithm that at first takes each data point as a cluster of its own (Rokach & Maimon, 2005). This method differs from K-Means because the number of clusters does not have to be determined first. Next, each cluster (data point) is merged into a new cluster with the closest cluster. This process is repeated until there is one cluster left, this represents a dendrogram that shows the hierarchy at which similarity level clusters were merged in the process. The dendrogram can be cut at the desired similarity level or a maximum number of clusters. The authors (Rokach & Maimon, 2005) further state that the closest cluster is determined based on the selected linkage criteria. An example of such criteria is Ward's criterion (Ward Jr, 1963). In this method the closest cluster is determined by comparing the sum of squared distances between clusters.

Most of the disadvantages such as the determination of the number of clusters beforehand and incorrect starting points of the clustering listed for the K-Means algorithm do not apply for the agglomerative hierarchical clustering algorithm. Due to the bottom-up approach, the clustering can always be reproduced similarly and prevents the risk that the algorithm starts with a set of data points that are too close to each other. However, there are some disadvantages of using this method as well. Firstly, agglomerative hierarchical clustering is not able to deal with missing values. Secondly, because of the bottom-up approach the running time of the algorithm can be long and compute- intensive.

3 Research Goal and Algorithm Requirements

This project aimed to uncover underlying skills required for math problems from accuracy scores of exams with an unsupervised machine learning algorithm. If fulfilled, the resulting algorithm could be later applied in the development of cognitive tutoring systems. The unsupervised machine learning algorithm could replace the currently prevailing approach of determining the set of required skills prior to tutor development (Anderson et al., 1995; Anderson, 2009).

By objectively uncovering skills from the data, the expectation is that more (underlying) skills are discovered. To explore this assumption, the following research question is proposed: *"How can skills required for solving mathematical problems be determined by a machine learning algorithm which only uses the accuracy scores from the mathematical exam data?"* The expectation is that the algorithm will discover a relation between questions for which a similar skill is required but there is a difference in difficulty. For example, such a relation between question A and question B represents that the set of skills required for answering question A is a subset of skills required for answering question B.

In order to successfully fulfil the task as presented in the research question, the used algorithm has to meet five requirements:

R1: The algorithm has to be able to find skill sets in a simulated data set where the skill set is known.

R2: The algorithm has to be able to show a hierarchy of the skill sets in a simulated data set where the skill set is known.

R3: The algorithm has to be developed in a way that it can be applied to any data set.

R4: The algorithm has to be able to show a hierarchy of the questions from existing mathematical exam data sets.

R5: The algorithm has to be able to uncover skills from existing mathematical exam data sets.

4 Knowledge Graph Algorithm

In this section, the simulated data set will be introduced and described. Following that, the aforementioned data set will be used to explain the proposed algorithm step by step.

4.1 Simulated Data Set

To apply an existing algorithm or to develop an algorithm, a data set was needed of which it was known how many skills were present. Therefore, an experiment with students has been simulated. The experiment has been performed in Python version 3.7.6 (Van Rossum & Drake, 2009) with the libraries NumPy (version 1.19.2) (Harris et al., 2020) and Pandas (version 1.1.2) (McKinney, 2010).

In the simulated experiment, 1000 'students' answered 100 'questions'. These questions could be answered correctly if the student mastered the skill of the question. The number of skills in this experiment was set to three. The chance for a student to master a skill was 50% per skill. The number of skills a student could master ranged between zero and all three skills. The skills for the questions were organised similarly: for every skill, there was a 50% chance for a question to be present, where the number of skills for a question ranged between zero and all three skills.

After assigning the skills, an experiment was simulated in which the students performed the questions based on their mastered skills. This implies that a student correctly answered a question if the student masters the skill(s) that are required for the question. The simulation returns a NumPy matrix consisting of the accuracy score per student per question. The students are presented per row, the questions are presented per column. The accuracy score is presented as 0 or 1 where 0 stands for incorrect and 1 stands for correct (Table 1). The NumPy matrix was then transformed to a Pandas dataframe. After rearranging the indices of the columns and rows (starting from 1 instead of 0), the dataframe was saved to a CSV file.

4.2 The Algorithm

After creating a simulated data set, the developed algorithm for uncovering skills from the data which is based on the order theory, is examined. As mentioned in the section 2.4, the expectation is that the algorithm can be applied to non-clustered data but will realise better results whenever the data is clustered

	Q1	**Q2**	**Q3**	**Q4**	**Q5**	**...**	**Q96**	**Q97**	**Q98**	**Q99**	**Q100**
S1	1	0	1	0	1	...	0	1	0	1	1
S2	0	0	1	1	1	...	0	1	0	1	1
S3	1	0	1	0	1	...	0	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...
S999	1	0	1	1	1	...	0	1	0	1	1
S1000	1	0	1	0	1	...	0	1	0	1	1

Table 1: Example of matrix design for the simulated data set. *Sx* represents a student and *Qx* represents a question.

first, preferably on both questions and the answers from students. Both methods will be examined in this section. The contents of the algorithm are discussed in section 4.2.1. Next, results of the application of the algorithm to the simulated data set will be discussed. The final three sections discuss the application of the combination of the developed algorithm with a clustering method and the validation of the algorithm.

4.2.1 The Knowledge Graph Algorithm

The core idea of the graph algorithm is to visualise the hierarchy of questions determined by a partial ordering. The visualisation should show which question was more difficult than the other and which question could be seen as prior knowledge to another question. To clarify, only the students that answered i.e., question A correct, were able to answer question B correct as well. The hierarchy will be visualised in a directed acyclic graph as mentioned in section 2. The nodes in the graph represent questions and the direct edges between the nodes visualise that a question is prior knowledge to another question. The question that is prior knowledge (easier) has an outgoing edge to the question that requires the prior knowledge (more difficult) to be answered correctly.

The knowledge graph algorithm is built in collaboration with N.A. Taatgen. The algorithm consists of three functions, each function will be explained thoroughly. The algorithm is developed in Python.

1. `Node_greater_than(a, b, strictness=0, correct_fraction=0.2)`
2. `Build_full_graph(m, strictness=0)`
3. `Prune_graph(G)`

In the first function `node_greater_than()`, a partial ordering is calculated on the questions. The function compares pairs of questions and their answers to determine how the question performed compared to the other question. This information is later used to create the edges between the nodes (questions) in the knowlege-graph. The function compares the nodes from the graph to determine which one is greater than the other. A node is a vector of scores (e.g., a question with all the answers of the students or a student with answers to all the questions). The general idea of the function is that *node a* is greater than *node b* if the majority of the answers in node a (*ai*) are greater than or equal to the answers in node b (*bi*). To explain the function in more detail, parts of the code as illustrated in Listing 1 will be discussed.

The function `node_greater_than()` takes four input parameters: `a, b, strictness` and `correct_fraction`. The input parameters `a` and `b` are vectors of answers that will be compared as explained above. Before explaining the other two input parameters `strictness` and `correct_fraction`, the operation of the function will be outlined. The function iterates over the answers of vector `a` and compares each answer (`a[i]`) to the answer with the similar index in vector `b` (`b[i]`) to determine which answer scored better. Whenever `b[i]` scored better than `a[i]`, 1 is added to the variable *worse*. Whenever `a[i]` scored better than `b[i]`, 1 is added to the variable *better*. Note that the difference between the scores must be at least 0.2 (20%), otherwise the scores are too similar. Finally, in case both answers were equal (less than 0.2 difference) but higher than 0.5, 1 is added to the variable *equal_correct*. The function performs nothing when both answers are equal (less than 0.2 difference) and lower than 0.5, this value implies that both answers are either wrong or of a low score and both answers did not score worse than the other. The difference of 20% was later added to the algorithm because some data sets contained missing values that

```python
def node_greater_than(a, b, strictness=0, correct_fraction=0.2):
    """This function compares two vectors of scores, a and b, and returns a boolean valua to indiciate if vector a is greater than vector b.
    input:
    a: a vector (column or row from a matrix) with values between 0 and 1
    b: a vector (column or row from a matrix) with values vetween 0 and 1, of equal length as a
    strictness: integer, default=0. The degree of strictness: higher strictness results vectors to be greater if the value of 'worse' is low and value of 'better' is high
    correct_faction: float, default = 0.2. Corrects for values of 0.5 if scores are e.g. means instead of binaray values
    """
    # for keeping track of times vector a scores worse than vector b
    worse = 0
    # for keeping track of times vector a scores better than vector b
    better = 0
    # for keeping track of times vector a scores equal to vector b
    equal_correct = 0

    for i in range(len(a)):
        # check if answer a[i] in vector a and answer b[i] from vector b
        # are not a NaN (missing) value
        if a[i] != np.nan and b[i] != np.nan:
            # determine which answer scored higher at least 20% higher
            if b[i] > a[i] + 0.2:
                # score of a[i] + 20% is lower than score of b
                # a[i] score is therefore worse
                worse += 1
            elif b[i] + 0.2 < a[i]:
                better += 1
                # score of a[i] is higher than score of b[i] + 20%
                # a[i] score is therefore better
            elif a[i] > 0.5:
                equal_correct += 1
                # score of a[i] is equal to score of b[i], both are 1
        # if both scores are 0 or the scores (when lower than 0.5) differ less than 20%, nothing happens.
        # both scores are incorrect

    # return boolean value to indicate if a is greater than b and add degree of strictness
    if strictness == 0:
        return (worse == 0) & (better > 0) & (equal_correct >= better * correct_fraction)
    else:
        return (better >= strictness * worse) & (equal_correct >= better * correct_fraction)
```

Listing 1: Python code for `node_greater_than()`. function

were substituted by the mean (section 5.2.2). The substituted decimal values ranged between 0 and 1. To determine if vector `a` scored better than `b`, the value of *worse* must be 0, the value of *better* must be higher than 0 and the value of *equal_correct* must be equal or higher than the value of *better* multiplied by the `correct_fraction`. Line 37 in Listing 1 illustrates the equation. If all three conditions are met, the function returns the boolean value `True` which implies that vector `a` is greater than vector `b`. Whenever one or more conditions are not met, the function will return the boolean value `False`, this implies that vector `a` is not greater than vector `b`.

Now that the operation of the function is explained, the remaining input parameters will be discussed. The input parameter `strictness` (as integer) is the degree of strictness that is applied to the operation of the algorithm. The degree of strictness implies that the higher the strictness, the more weight is put the *worse* variable in the function. This is illustrated in line 39 of Listing 1. The earlier mentioned equation from line 37, is slightly adjusted. If the value `strictness` is not 0, the value of *better* must be higher than or equal to the value of *worse* multiplied by the value of `strictness`. The reason for adding this feature is to prevent the resulting knowledge graph from being overfitted by only being based on the accuracy of questions. With a low value for `strictness`, there is a high probability for connection between nodes by luck. Therefore, a strictness ranging between 5 and 15 is recommended although this is not always feasible. In cases where it is not feasible, it is recommended to use the highest value for strictness where still most nodes are connected and the returned graph from the test and train set (section 4.5) are most similar to each other.

The final input parameter is `correct_faction`, this parameter is by default set to `0.2`. The correct_fraction is used in lines 37 and 39 in Listing 1 to determine if node `a` is greater than node `b`. The amount of *equal_correct* must be at least higher than 20% of the value of variable *better*. This stricter equation is added to prevent that nodes where *equal_correct* is equal to or higher than 0 will return as a *greater than* node. Whenever the *equal_correct* value is bigger than 20% than the *better* value and the other equations meet in line 37 or 39 of Listing 1, it can be stated that node `a` is greater than node `b`.

The function `node_greater_than()` is not directly applied by the user. The user will apply the function `build_full_graph()`, this function calls the `node_greater_than()` function. The input parameters of `build_full_graph(m, strictness=0)` are `m` and `strictness`. The parameter `strictness` is explained above. The input parameter `m`, must be a NumPy matrix with student accuracy scores (ranging from 0 to 1) that is displayed with either the students as rows and questions as columns or the other way around. The latter is dependent on creating a knowledge graph of the questions or students.

When `build_full_graph()` (Listing 2) is called, first an empty graph is created by using the iGraph library (Csardi & Nepusz, 2006) (version 0.8.3). Next, another function from the iGraph library is called for adding nodes (the number of columns from the matrix) to the empty graph. The nodes represent either the questions or the students, dependent on what is represented on the columns. The next step is to compare the columns (vectors of answers) for determining the relationships between the columns as illustrated in lines 11 to 20 of Listing 2. To determine which vector 'is greater than' the other, both vectors are put into the `node_greater_than()` function in both ways e.g., `node_greater_than(a, b)` and `node_greater_than(b, a)`. When the output returns `True`, a directed relation (edge) is added between the two vectors (nodes) from the vector that performed better. The assumption is that in case a question is answered correctly by a higher population compared to the other question, the question was

```
1 def build_full_graph(m, strictness=0):
```

```
    """This function constructs a connected directed network graph from a matrix.
    Each column is a node. The directed edges are determined with the
    node_greater_than function.
    input
    m: a matrix of students answers (students * questions)
    strictness:integer, default=0. The degree of strictness: higher strictness
    results vectors to be greater if the value of 'worse' is low and value of '
    better' is high"""
    # create an empty graph
    G = ig.Graph(directed=True)
    # add all nodes: number of columns in the matrix
    G.add_vertices(list(range(np.size(m,1))))
    # compare the columns (i and j) of the matrix to determine which one scored
    better than the other
    for i in range(np.size(m,1)-1):
        for j in range(i+1, np.size(m,1)):
    # if node_greater_than returns True for the coparison of i and j, i scored
    better
            if node_greater_than(m[:,i], m[:,j], strictness=strictness):
                # a directed edge is drawn from node i to j
                G.add_edge(i,j)
     # if node_greater_than returns True for the coparison of j and i, j scored
    better
            elif node_greater_than(m[:,j], m[:,i], strictness=strictness):
                # a directed edge is drawn from node j to i
                G.add_edge(j,i)
    return(G)
```

Listing 2: Python code for `build_full_graph()` function.

```
def prune_graph(G):
    """ This function prunes (removes) unnecessary edges from the directed graph.
    input:   G: a directed graph"""
    # create an empty list for the to added node tuples to remove these extra edges
    remove_list = []
    for i in range(len(g.vs())):
        for j in range(len(g.vs())):
            # check if j has an incoming edge from i
            if j in g.neighbors(i, 'OUT'):
                for k in range(len(g.vs())):
    # check if k has an incoming edge from i and if j has an incoming edge from k
    # if that is the case, remove the edge between i and j because k forms a layer
    in between
                    if k in g.neighbors(i, 'OUT') and k in g.neighbors(j, 'IN'):
                        remove_list.append((i,j))
                        break
    # Delete the edges between the nodes from the list of tuples
    g.delete_edges(remove_list)
    return(G)
```

Listing 3: Python code for `prune_graph()` function.

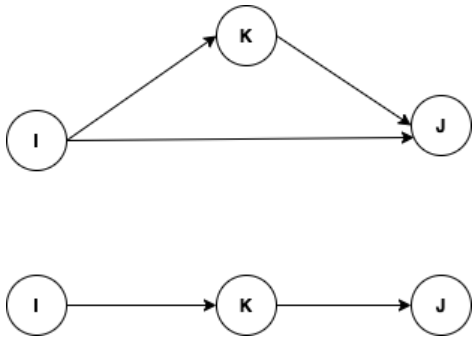


Figure 2: Basic operation of function `prune_graph()`.

easier and could be prior knowledge or a skill that is required for answering the other question. After adding all the edges, the complete graph is returned by the function.

The function `build_full_graph())` compares all vectors to each other which results in redundant edges in the returned graph. The function `prune_graph(G)` (Listing 3) that takes the returned graph from `build_full_graph()` as input, reduces the redundant edges. The aim of the function is to check for shared neighbours (transitive relations) for each node `i` and `j` in the graph whenever there is a directed edge from node `i` to `j`. When nodes `i` and `j` share a neighbour (`k`), the direction of the edges between the neighbour and the nodes is compared. As illustrated in lines 11 to 17 (Listing 3), if there is a directed edge from node `i` to node `k` and a directed edge from node `k` to node `j`, the edge between nodes `i` and `j` is removed which is illustrated in Figure 2. Figures 3 and 4 illustrate the effect of pruning a graph. Both graphs are results from the simulated data that is clustered before applying the algorithm. As Figure 3 illustrates, the unnecessary edges are removed which make it easier to interpret the knowledge graph.

4.3 Application of Knowledge Graph Algorithm to Simulated Data Set

The knowledge graph algorithm is applied by calling the developed functions, as described in the previous section. Listing 4 illustrates the lines of codes that have been used to create the knowledge graph. In line

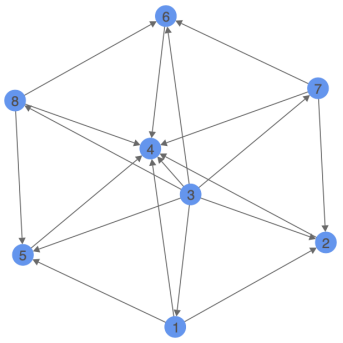


Figure 3: Knowledge graph of clustered simulated data set without use of `prune_graph()` function.

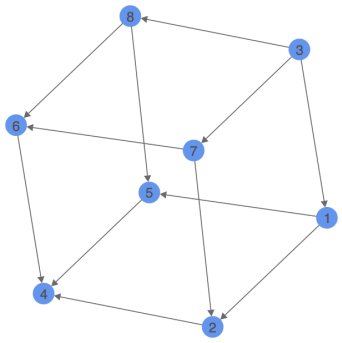


Figure 4: Knowledge graph of clustered simulated data set with use of `prune_graph()` function.

```
g = build_full_graph(datamatrix, strictness=0)
g1 = prune_graph(g)
g.vs["label"] = list(range(1,len(datamatrix[0])+1))
plt= ig.plot(g1, vertex_label = range(1, g.vcount()+1),
             edge_width=1.2, edge_arrow_size=.6, edge_arrow_width=1.5,
             edge_length=1.3, edge_color='#696969',
             vertex_frame_color='#6495ED', vertex_color='#6495ED',
             vertex_label_color='#524F4F', vertex_size=27,
             vertex_label_size=16, vertex_label_dist=-0.4,
             layout='kk')
plt
```

Listing 4: Application of knowledge graph algorithm.

1 of Listing 4, the function `build_full_graph()` is called, where datamatrix is the matrix of the simulated data set. The `strictness` is set to 0. The simulated data set is a data set without noise because it does not contain missing values or miss-clicks by participants. Therefore, the expectation was that no value was needed for `strictness`. In line 2 of Listing 4, function `prune_graph()` is called to reduce unnecessary edges in the graph. Next, in line 3 the labels of the nodes are added, the labels are the question numbers. Finally, the graph is plotted and styled with the code represented in lines 4-11 of Listing 4. The values applied for plotting the sizes of the edges and vertices differ per data set.

In the simulated data set, a total of three skills could be present for a student (ranging from 0 to 3). This implies that eight combinations for a set of skills are possible for a student: 2^3. In Figure 5, the hierarchy of the eight combinations is visualised. The nodes with an outgoing edge represent prior knowledge to the node that receives the edge. The expectation is that the algorithm is able to create a graph in the shape of a cube like Figure 5.

In Figure 6 the result of the knowledge graph algorithm applied to the simulated data is presented. Although the unnecessary edges are removed, there is still a high number of edges because 100 questions are plotted. Due to this high amount of edges, it is hard to recognize a shape in the graph. However, the

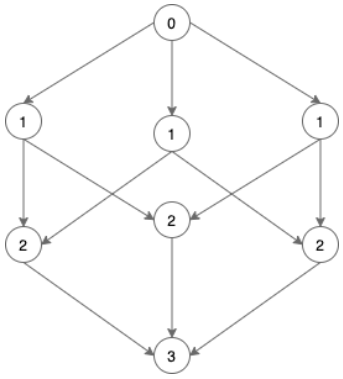


Figure 5: Hierarchy of eight possible combinations of skills visualised in a directed acyclic graph. The graph is similar to the graph as illustrated in section 2, Figure 1. A node represents a skill combination. The digits in the nodes represent the number of present skills in each combination. The edges visualise the relation between the combinations.

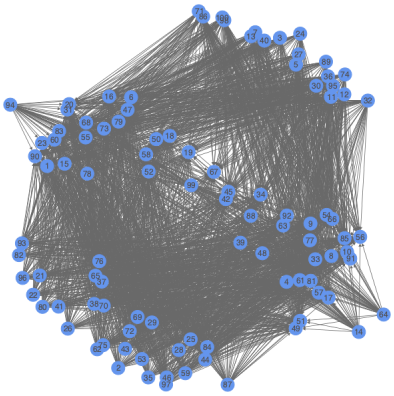


Figure 6: First result of knowledge graph algorithm applied to simulated data set. In the graph, each node represents a question and the directed edges visualise which question is prior knowledge to the question at the arrow of the edge.

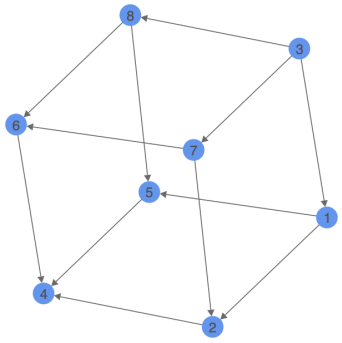


Figure 7: Knowledge graph of simulated data set after applying agglomerative hierarchical clustering with eight clusters. In the graph, each node represents a cluster of students with a unique skill set. The directed edges visualise which cluster of students contain fewer skills compared to the cluster of students at the end (arrow) of the edge.

plot functionality of the library iGraph (Csardi & Nepusz, 2006) attempts to plot the most similar nodes close together. By further analysing Figure 6, the hierarchy seems to be plotted the same way as in Figure 5. To declutter the visualisation, the earlier suggested approach of clustering the data first, will be applied in the next section.

4.3.1 Knowledge Graph Algorithm Combined with Clustering Methods

Earlier, in section 2.4 a suggestion was made to cluster the data before applying the knowledge graph algorithm. The examined suggestion in this section is to cluster the students of the simulated data set. A total of eight combinations of skills were possible for a student to have in the simulated data (3 skills, either present or not: 2^3). Therefore, the simulated data set will be clustered into eight clusters of students where each cluster represents one of the skill sets.

The agglomerative hierarchical clustering algorithm has been selected to perform the clustering. This method turned out to be more suitable for this project compared to the K-means algorithm. The main reason for that is, that the initial number of clusters does not have to be determined before the application of the algorithm. For K-Means, the determination of the number of clusters beforehand, is required. Although for the number of clusters (possible skill combinations) is known for the simulated data set, the possible skill combinations will not be known for real data set. Furthermore, in K-Means the random selection of cluster means to start the cluster iteration can result in clusters that are biased and too close to each other. This approach does not allow reproduction as well because the selected cluster means may be different at the start of a new iteration. Agglomerative hierarchical clustering is a bottom-up approach where each data point will be a cluster of itself, this allows reproduction and reduces the chance for bias (Boehmke & Greenwell, 2019). However, both clustering methods are unable to deal with missing

values, therefore whenever missing values are present in a data set, they have to be substituted before the clustering method is applied.

Agglomerative hierarchical clustering has been applied to the simulated data set by use of the implementation from the `sklearn` library in Python: `sklearn.cluster.AgglomerativeClustering`. For the function `AgglomerativeClustering()`, only the parameter `n_clusters` was adjusted, for the other parameters the default value was used. The latter implies that the parameter `affinity` was set to the distance metric `Euclidean` and the parameter `linkage` was set to `Ward`. In order to apply Ward's linkage, affinity has to be set to euclidean. With these settings, the closest cluster is determined by the lowest sum of squares of the Euclidean distance. Ward's linking criterion was selected based on the results of the `agnes` function from the `cluster` package (Maechler, Rousseeuw, Struyf, Hubert, & Hornik, 2021) in R which returned the highest *agglomerative coefficient* for this criterion.

Next, the knowledge graph algorithm will be applied to the clustered data. The algorithm will create a graph with eight nodes, the clusters of students. The expectation is that the figure of the visualisation will look like a cube as Figure 5, as explained earlier. To apply the knowledge graph algorithm, the average score of a cluster had to be calculated. The attribute `labels_` from the class `sklearn.cluster.AgglomerativeClustering` has been used to retrieve the cluster number for each student. For each cluster, the average vector (score to the questions) based on the student vectors in the cluster, could then be calculated. This resulted in a NumPy matrix of eight rows (clusters of students) and 100 columns (questions). Before applying the knowledge graph algorithm, the matrix was transposed first to represent the clusters of students as columns. The result of the knowledge graph algorithm is presented in Figure 7. This application returns a graph that looks like a cube, this implies that our expectation was correct and we can assume that the knowledge graph algorithm works. It can also be stated that requirement 1 and 2 as mentioned in section 3, are met by the algorithm. However, the algorithm is able to find the skill sets in the simulated data with (R1) because of the applied clustering method. Furthermore, the algorithm shows the hierarchy of the skill sets in Figure 3, which meets requirement 2 (R2).

In the above describes analysis only the students have been clustered. Since the data set does not contain noise and can be mirrored, the above assumptions also apply for the clustering of the questions.

4.4 Clustering the Data: Determine Optimal Amount of Clusters

The previous section showed that clustering the data before applying the knowledge graph algorithm, aided in uncovering the present skills in the simulated data. Therefore, this step is included in the process of applying the knowledge graph algorithm. A difference for real data compared to the simulated is that the number of possible skill combinations was known in the simulated data set. The skill sets are unknown in the data sets to which the knowledge graph algorithm is applied because the algorithm aims to uncover skills. Even data sets of which skills are known can be analysed with the knowledge graph algorithm because more underlying unknown skills could be present. Therefore, the number of clusters can not be determined based on the number of skill sets because the amount of skill sets is unknown. As mentioned earlier, the amount of clusters does not have to be determined before applying agglomerative hierarchical clustering. However, it remains to be determined at what number of clusters the returned dendrogram should be cut off.

In this section, different methods and algorithms have been analysed that determine the optimal amount of clusters for agglomerative hierarchical clustering for a data set. The best performing method had to meet some criteria. The first criterion is that the method should return a high amount of clusters to keep as many differences between students as possible. With this approach, minor differences between students are better expressed and contribute to a better overview of the data. Another criterion is that the method should be efficient because the to be analysed data sets can be large. The clustered data results in a faster application of the knowledge graph algorithm as it reduces the computational intensity.

First, existing methods for determining the optimal amount of clusters have been examined. For agglomerative hierarchical clustering, the package `pvclust` (Suzuki & Shimodaira, 2006) in R version 4.0.0 (2020-04-24) (R Core Team, 2017) could be used the determine the optimal cut of the dendrogram. The method calculates via multi-scale bootstrap resampling the p-values of the clusters in the hierarchical dendrogram. The p-value is a value between 0 and 1 to indicate the strength of the particular cluster in the data. Based on the best p-values of the clusters in the dendrogram, the optimal number of clusters could be determined. The `pvclust` method did not seem to be applicable in this project. Due to the large amount of data attributes in some data sets, the method was rather time-consuming and computationally intensive. For the smaller data sets, the p-values were not high enough to determine the optimal split. The `pvclust` package turned out not to be suitable for this project because the criterion efficiency is not met.

Another approach for agglomerative hierarchical clustering is to just plot the dendrogram and try to determine at what point the smaller clusters become the most similar to each other. Multiple libraries in Python or R can aid in finding the smallest distance. These methods yielded a low number of clusters (max 10) for the different data sets used in this project. This method is also not suitable for this project because one of the criteria, a high amount of clusters, is not met.

There are more methods used for determining the optimal number of clusters, which are meant for cluster methods in general, but suitable for agglomerative hierarchical clustering. Kassambara (2018) highlights some of these methods: the *elbow* method (Thorndike, 1953) and the *average silhouette* method (Kaufman & Rousseeuw, 2009). These methods were applied in R to the simulated data set. The reason for applying these methods in R was because Kassambara (2018) explained the methods along with an existing R package `factoextra` (Kassambara & Mundt, 2020) that allowed to quickly check if these methods were suitable for this project. From the package, the `fviz_nbclust()` function is applied for the calculation of the methods. The results of all these methods were not higher than 3 for the simulated data set. Although these methods did not yield a large amount of clusters, the approach of the methods will briefly be discussed.

4.4.1 Elbow Method

The aim of the *elbow method* (Thorndike, 1953) is to find the 'elbow' or 'knee' of a curve in a plotted graph. Kassambara (2018) states that in the graph the number of clusters is presented on the x-axis against the amount of explained variance on the y-axis. The explained variance in this method was calculated by the total within-cluster sum of squares (WSS) for each number of clusters. The method selects the number of clusters where the WSS is minimised and does not change much when adding an extra number of clusters. This is represented in the elbow, after the 'elbow' the line of the WSS becomes flat and adding

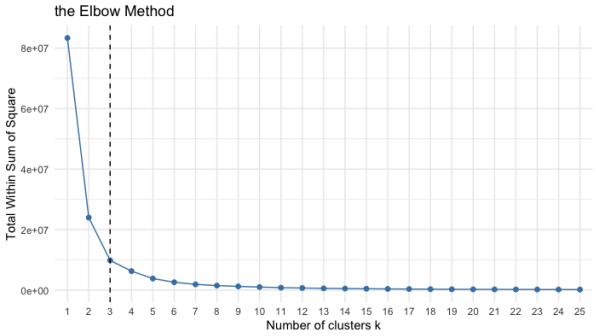


Figure 8: Result of application of function `fviz_nbclust(data, hcut, method = "wss", k.max = 25)` where data is the simulated data and `method="wss"` stands for the Elbow method. On the y-axis, the total within-cluster sum of squares is presented and on the x-axis the number of clusters of the model is presented. According to this method, a model with 3 clusters is the best performing model.

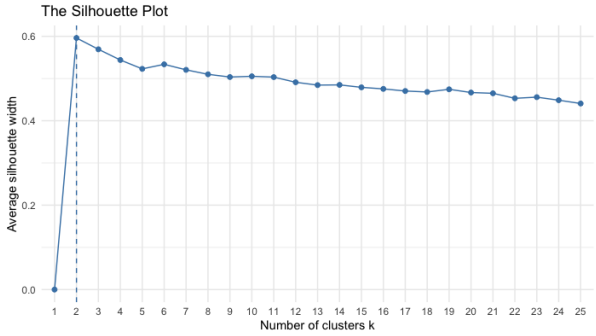


Figure 9: Result of application of function `fviz_nbclust(data, hcut, method = "silhouette", k.max = 25)` where data is the simulated data and `method="silhouette"` stands for the average silhouette method. On the y-axis, the average silhouette width is presented and on the x-axis the number of clusters of the model is presented. According to this method, a model with 2 clusters is the best performing model.

extra clusters does not affect the value much, the information gain becomes low.

Figure 8 illustrates the results of the elbow method applied to the simulated data set that is created in R with function `fviz_nbclust(data, hcut, method = "wss", k.max = 25)` from the package `factoextra` (Kassambara & Mundt, 2020). The parameter `hcut` means that the calculation is applied for hierarchical clustering and the parameter `method = 'WSS'` is called to determine the optimal number of clusters with the elbow method. As the graph in Figure 8 illustrates, the amount of 3 clusters was found as the optimal number of clusters. A larger amount of clusters is desired, for the simulated data set the expectation is to find eight as the optimal number of clusters (section 4.3).

4.4.2 Average Silhouette Method

Another method suggested by Kassambara (2018) which has been applied is the *average silhouette method* (Kaufman & Rousseeuw, 2009). The author states that in this method, the quality of the clustering is measured by determining how well every object lies in its clusters. Based on the average silhouette method, the optimal number of clusters is where the value of the average silhouette is maximised.

The results of the average silhouette method have to be determined again by using the `fviz_nbclust()` function from the package `factoextra` but with different parameters. Only the `method = 'silhouette'` differed compared to the previous method. In Figure 9 the results of the method applied to the simulated data set are illustrated. Again a low number of clusters is found to be the optimal number where the average silhouette is maximised: 2.

Both methods described above were also applied to the *TIMSS* math workbook 1 (train set) data set

(section 5.2.2). The suggested optimal number of clusters were 3 (Elbow) and 2 (Silhouette) but the data set contained approximately 3500 rows which makes it unlikely to only cluster the data is such a small amount of clusters, especially because for this project a large amount of clusters is desired to retain most differences between the students. Therefore, the quest to find an existing method for finding the optimal number of clusters but for a large amount of clusters continued.

4.4.3 AIC Method

Like in statistics, the *Akaike Information Criterion* (AIC) (Akaike, 1974) can also be used for determining the optimal number of clusters. The AIC value is in general used to determine the quality of a model and how well it fits the data in order to perform model selection (Aho, Derryberry, & Peterson, 2014). The authors explain how the AIC equation can be translated to determine the AIC value for a K-Means clustering model (Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, 2008). This explanation is again translated to a package in R: `ClusterR` (Mouselimis, 2020). Within this package, the function `Optimal_Clusters_KMeans()` is enclosed which allows determining the optimal number of clusters with the AIC as the criterion.

The function was applied to the simulated data and the *TIMSS* math workbook 1 (train set) to investigate if this method results in a higher optimal number of clusters. The function automatically plots the AIC values. Figure 10 illustrates the results for the simulated data set from the function `Optimal_Clusters_ KMeans(data=data, max_clusters=25, criterion="AIC")`. The maximum number of clusters was set to 25. A similar approach for determining the best model in statistics by use of the AIC value is applied: the model with the lowest AIC is in general the best performing model (Akaike, 1974). However, as can be seen from Figure 10, the AIC values flatten out and there is not a particular minimum. Therefore, in this project, the minimum distance between two models is calculated and the first model of the two is selected as the optimal number of clusters. This approach yielded a higher number of clusters for the two data sets. The `Optimal_Clusters_KMeans()` function was therefore translated to be applicable for agglomerative hierarchical clustering because that is the clustering method that has been selected for this project. The translation is written in Python.

The translated function is called `AIC_HC()`, the entire code for the function is enclosed in Appendix A. The function takes as input parameters: a data set (in NumPy format) that has to be clustered and the maximum number of clusters to calculate the AIC value. The function applies the `AgglomerativeClustering()` function from the `sklearn` library for each cluster amount in the range of the number of clusters. For each cluster amount, the within-cluster sum of squares (WCSS) is determined. The WCSS is used to calculate the AIC value. The AIC value is determined with an equation that is translated from the K-Means equation (Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, 2008). Next, a dictionary is created with all AIC values of the cluster amount and the differences between the values is determined. The differences are determined by a difference matrix. In the difference matrix, the `AIC_HC()` function checks for the first positive value. This implies that there is a positive difference from one model to the next and the first minimum is found. The index of the first model will be determined and selected as the optimal number of clusters. To clarify the selection, Figure 11 illustrates the point where a positive difference occurs between two models and also the (first) minimum point is found. This point is at a model consisting of eight clusters, which meets the expectation for the simulated data set as

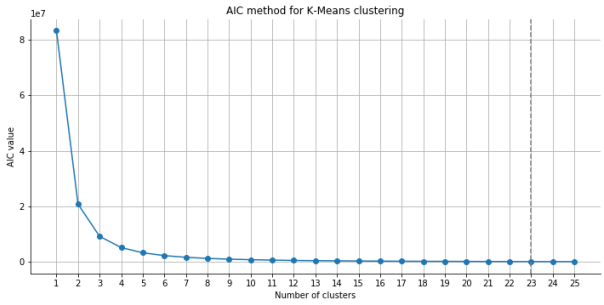


Figure 10: Result from the application function `Optimal_Clusters_KMeans(data=data, max_clusters=25, criterion="AIC")` where `data` is the simulated data and the criterion is the AIC value. On the y-axis, the AIC value is represented and on the x-axis the number of clusters of the model is presented. The minimum distance was found at 23 clusters, which makes it the best performing model according to this method.

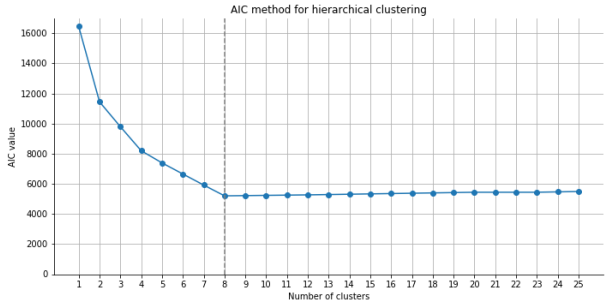


Figure 11: Result from the application of `AIC_HC(data, 25` where `data` is the simulated data and the number the maximum amount of clusters is 25. On the y-axis, the AIC value is represented and on the x-axis the number of clusters of the model is presented. The positive difference and minimum point were found at eight clusters. The model with 8 clusters is the best performing model according to this method.

explained in section 4.3.

Whenever the function does not find a positive value in the difference matrix, it will return an error which suggests to enter a higher maximum amount of clusters. For all data sets to which the function is applied, a higher amount of clusters did result in a positive value in the difference matrix. Because this project aims to have as many as possible clusters to retain as most differences as possible, this AIC method for agglomerative hierarchical clustering is applied.

A finding that has to be noted is that the developed `AIC_HC()` function returns a different result (eight clusters) for the simulated data compared to the `Optimal_Clusters_KMeans()` function (23 clusters). For other data sets, like the *TIMSS* math workbook 1, 622 clusters were found in the `AIC_HC()` function but 55 clusters were found according to the other function where `cluster_max` was set to 60. The function `Optimal_Clusters_KMeans()` is computationally intensive, therefore the function has not been applied for a similar number of clusters, 700, as performed in the `AIC_HC()`. The assumption is that the difference occurred because of the error sensitivity of K-Means as explained earlier. The `AIC_HC()` function is less computationally intensive compared to `Optimal_Clusters_KMeans()`. The `AIC_HC()` function manages to determine the optimal number of clusters in approximately 10 minutes with a maximum amount of clusters set to 700 and a data set with 26 columns and approximately 3500 rows. The `Optimal_Clusters_KMeans()` function did not return a result (with plotting disabled) after 40 minutes of running after which the function was forced quitted.

4.5 Validation of Knowledge Graph Algorithm

It is important to validate the knowledge graph algorithm to ensure that the method is reliable. The method used for validation is based on the *Holdout method* (Raschka, 2018). In this method, the data sets have been split into a train (50 %) and test (50 %) set. The optimal number of clusters is determined by applying the `AIC_HC()` function (section 4.4) to the train set. Next, the knowledge graph algorithm is applied to the clustered train set. The best performing value for the `strictness` parameter is determined by applying different values and comparing them. To start the validation of the algorithm, the test set is clustered into the same amount of clusters as the train set. Next, the knowledge graph algorithm is applied to the clustered train set with a similar value for the `strictness` parameter. Finally, the two graphs can be compared to determine which connections between questions which connections are present in both graphs and are therefore more robust. To determine the best fitting value for `strictness`, values can be applied to both train and test sets. When the central nodes and most edges are similar in both knowledge graphs, the best fitting value for `strictness` is found.

The split is performed by the class `sklearn.model_selection.train_test_split()` from the library `sklearn` and module `sklearn.model_selection` in Python. By default, the split method equally random divides the data into two sets. The function that is called to split the data into a train and test set: `X_train, X_test = train_test_split(datamatrix, test_size=0.5)`. Only the parameter `test_size` is set to the desired value of 0.5 (50%). The other parameters from the `train_test_split()` have been set to default. This implies that the parameter `shuffle` has by default been set to `True` to shuffle the data first before splitting it into different sets.

The method of validation described above differs from the classical *Holdout method* where the test data exists out of labelled data. In this project, labelled data can be seen as data in which it is known what and how many skills are present. The main reason for applying validation is to be certain about the value of the `strictness` parameter. Validation methods like *K-fold Cross-Validation* (Hawkins, Basak, & Mills, 2003) have been considered but have not been applied due to the high amount of manually comparing of the returned knowledge graphs.

The validation method described above has been applied to the simulated data set and both sets returned a graph with the shape of a cube. The relations were similar but the labels of the nodes differed. The difference in node labels is caused by the fact that cluster numbers cannot be fixed when the data is clustered (into clusters of students). Therefore, it can be assumed that the graphs of both data sets are similar.

4.6 Step Overview of Knowledge Graph Algorithm

The different steps that have to be performed to apply the knowledge graph algorithm have been described in the previous sections. To give an overview of all the steps, the steps from the knowledge graph algorithm are described below.

1. Prepare data into NumPy matrix format where the students are presented on the rows and the questions are presented on the columns. The values only contain the accuracy score that a student scored for a question, the values are ranging between 0 and 1 (meets R3, section 3).

2. Split the data into train and test sets (section 4.5).

3. Apply the `AIC_HC()` function on the train set to determine the optimal number of clusters (section 4.4).
4. Apply agglomerative hierarchical clustering (section 2.4) with *n* number of clusters, based on the output of the previous step.
5. Apply the *knowledge graph algorithm* (section 4.2.1) with the clustered data and experiment with the value of `strictness` until most or all nodes are connected and not too many edges overlap.
6. Validate the algorithm by applying the knowledge graph algorithm to the test set with similar values as the train set for the number of clusters and `strictness`.
7. Higher and lower the value for `strictness` for both knowledge graphs (test and train), determine at which value the central nodes and most edges are similar.

5 Knowledge Graph Algorithm Applied to Multiple Data Sets

In this section, the application of the complete knowledge graph, including the results, will be discussed for multiple data sets. For each data set, a brief description of the data is given, followed by the applied preprocessing steps. Afterwards, the results of the knowledge graph algorithm are discussed.

5.1 Praxis from ETS

The first analysed data set, is provided by the organisation Educational Testing Service (ETS) (*About ETS*, 2021) (Copyright © 2021 ETS). The data is a pretest math measure consisting of 10 questions, performed by 1435 students. The offered test from ETS is the *Praxis® Core Academic Skills for Educators* test (*About the Praxis Tests*, 2021). This *Praxis* test, measures the academic math skills for candidates that are preparing to be teachers. Academic math skills are deemed by teacher educators to be essential for future teachers. The test was a multiple-choice test. Figures 12 and 13 illustrate two questions from the *Praxis* test.

5.1.1 Data Preprocessing

For the analysis, the student scores, response times and total scores were used. Therefore, the unnecessary columns had to be removed first. The preprocessing is performed in Python to create a NumPy matrix similar to the simulated data: student scores represented on the rows and questions represented in the columns.

After reading the data as a Pandas dataframe, the unnecessary columns were removed. Next, the first quantile of the reaction time per question was calculated. This is the first 5% of the reaction times (e.g., the lowest RT times per question of all students). A response quicker than the first quantile is assumed to be either a miss-click or a guess by the student. Therefore, scores with a reaction time quicker than the first quantile were changed to a score of 0. A total of 222 data entries have been changed to 0, this is 1.5% of the total data entries (10×1435).

Next, the reaction times column and total score column were removed from the dataframe and the dataframe was transformed into a NumPy matrix. Finally, the NumPy matrix was split into a train and test set by using the `train_test_split()` function from the `sklearn` library.

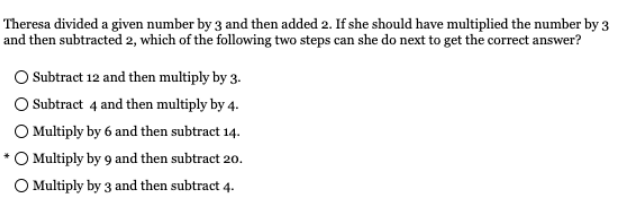


Figure 12: A question similar to question 3 from the *Praxis Core Academic Skills for Educators* test (*About the Praxis Tests*, 2021).

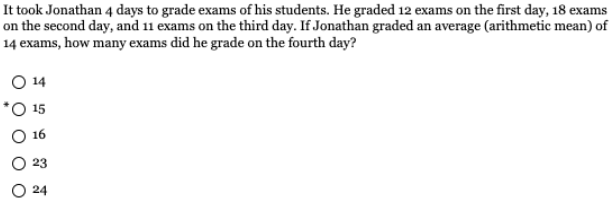


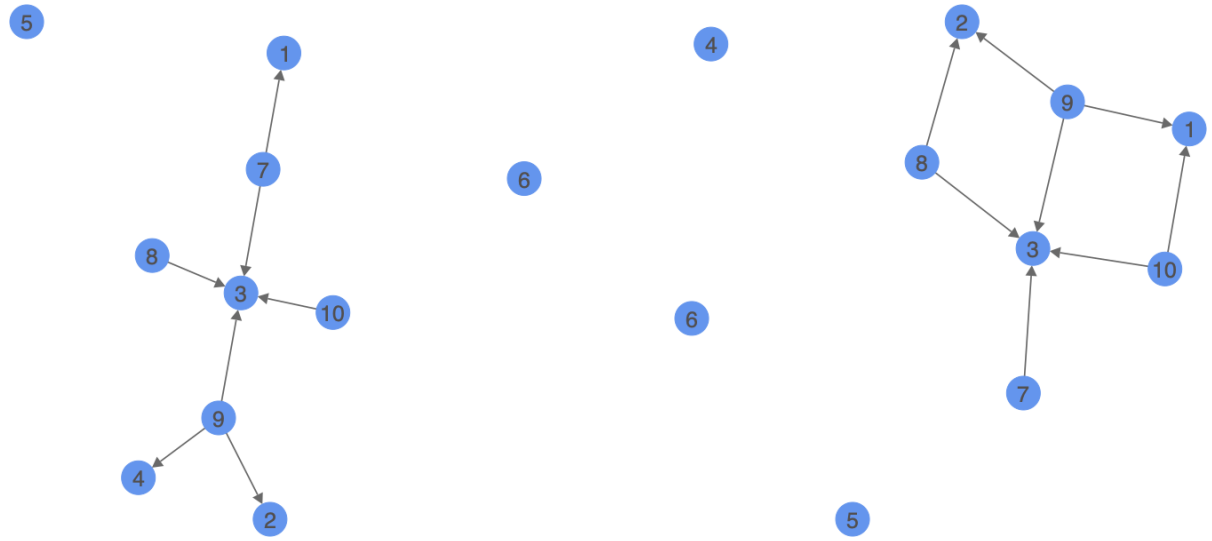
Figure 13: A question similar to question 9 from the *Praxis Core Academic Skills for Educators* test (*About the Praxis Tests*, 2021).

5.1.2 Knowledge Graph Analysis

As described in the steps of the complete algorithm (section 4.6), after preprocessing and splitting of the data, the optimal number of clusters has to be determined by applying the `AIC_HC()` function. The function was applied with the maximum number of clusters set to 200, lower applied values did not return an optimal number of clusters. The optimal number of clusters returned by `AIC_HC()` was 157. The knowledge graph algorithm was then applied to the train set of the data with 157 as the number of clusters.

In Figure 14a the result of the knowledge graph algorithm applied on the train set of the ETS data is presented. For `strictness`, the value 3 is determined to be the best fitting value. At this value, both graphs for the test and train set are most similar to each other and weak connections that can appear by luck are reduced. By analysing the graph from the train set, question 3 takes a central position where questions 7, 8, 9 and 10 are assumed to be prior knowledge to question 3. The questions 5 and 6 are not connected in the graph, none of the other questions are assumed to be prior knowledge and questions 5 and 6 are not assumed to be prior knowledge to the other questions.

Before discussing the assignments and their relations in more detail, the knowledge graph of the test set will be analysed first for determining the mutual edges in the graphs. The same maximum amount of clusters and a similar value for `strictness` was applied. Figure 14b shows the result of the application of the knowledge graph algorithm to the test set. Like in the knowledge graph from the train set (Figure 14a), question 3 takes a central position in which questions 7-10 are assumed to be prior knowledge



(a) ETS *Praxis* train set

(b) ETS *Praxis* train set

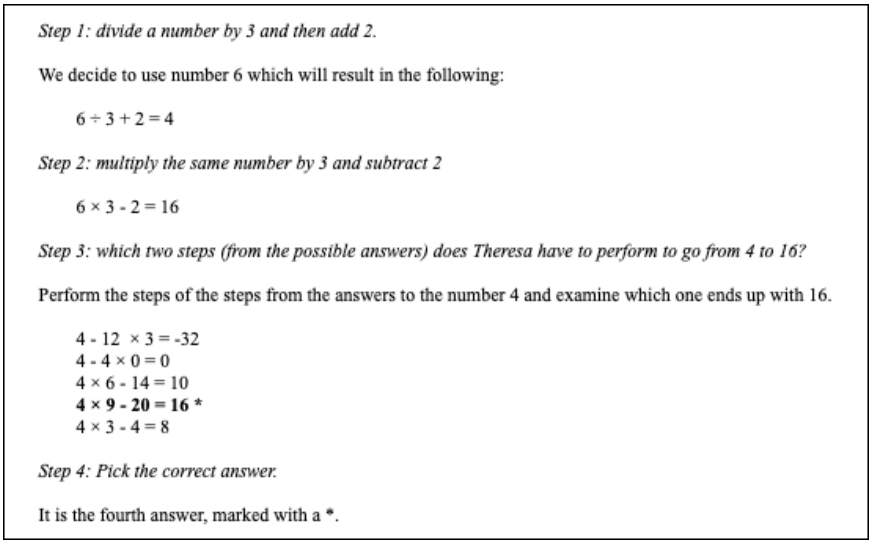
Figure 14: Knowledge graph of the ETS *Praxis* train set (a) and test set (b) that were both clustered into 157 clusters and a value of 2 for `strictness` was applied. The nodes represent each question in the ETS *Praxis* data set. The directed edges illustrate how the questions are related to each other: a question contains a subset of skills needed for solving another questions if the node has an out-going edge to another node.

Data set	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Train	0.18	0.35	0.29	0.39	0.46	0.42	0.55	0.59	0.66	0.57
Test	0.22	0.36	0.27	0.43	0.47	0.42	0.58	0.61	0.64	0.61

Table 2: Accuracy scores from ETS *Praxis* train and test data set.

to question 3. In this graph, questions 5 and 6 do not have any edges to other questions as well. The knowledge graph of the test set differs from the knowledge graph of the train set in that question 4 is not connected, questions 8 and 10 have more outgoing edges and the outgoing edges from question 9 are different.

To understand what the edges represent, the assignments have been analysed in detail. For example, question 3 can be marked as a difficult question because its accuracy score is the second-lowest score, compared to the other questions: 0.29 (train set) and 0.27 (test set) (Table 2). The content of the question is illustrated in Figure 12. To discuss why this question was difficult, a possible step-by-set solution is given in Figure 15. Although a basic math skill is required, it is a tricky question if the question and answers are not read carefully. The difficulty of question 3 is illustrated by the algorithm because question 3 only receives edges and it receives the most edges compared to the other questions. The questions from which question 3 receives edges are the questions 7-10 which implies that the skills present in these questions are required for answering question 3. The questions 7-10 are all story assignments where the key is to read carefully. Furthermore, the accuracy scores of questions 7-10 are the highest scores (0.55-0.66, Table 2). Because the questions 7-10 are all story assignments and the accuracy is high, it can be assumed that these questions require skills that are the base for the skill(s) needed in question 3. For the questions 7-10 it is assumed that in each question a different skill or the exact same skill is required, therefore these questions are not having in-going or out-going edges to each other. In both scenarios, no



Step 1: divide a number by 3 and then add 2.

We decide to use number 6 which will result in the following:

$6 \div 3 + 2 = 4$

Step 2: multiply the same number by 3 and subtract 2

$6 \times 3 - 2 = 16$

Step 3: which two steps (from the possible answers) does Theresa have to perform to go from 4 to 16?

Perform the steps of the steps from the answers to the number 4 and examine which one ends up with 16.

$4 - 12 \times 3 = -32$
$4 - 4 \times 0 = 0$
$4 \times 6 - 14 = 10$
$4 \times 9 - 20 = 16$ *
$4 \times 3 - 4 = 8$

Step 4: Pick the correct answer.

It is the fourth answer, marked with a *****.

Figure 15: A possible step-by-step solution to question 3 of ETS *Praxis* test.

edges will be drawn between the questions.

Except for the edge from question 9 to question 2, the different in-going and out-going edges of the questions 1, 2, and 4 in both graphs will not be analysed because these edges are not shared between both graphs. Therefore, these edges are assumed to be weak. The topics of these questions, including the questions 5 and 6, will be discussed to explain why these questions have none or weak edges.

The questions 1 and 2 both required the student to work with equations but in question 1 two equations were given that have to form a system of three equations where one value for x satisfies all three equations. The third equation has to be selected from the possible answers. In question 2, the student had to come up with a linear equation that fits the line graph where the crossing points at the x- and y-axis are can be determined from the graph. The accuracy scores (Table 2) show that question 1 was the most difficult question of all questions. The accuracy scores of question 2 are also low but the question is less difficult than the questions 1 and 3.

In both graphs, question 2 receives an in-going edge from question 9. Based on the accuracy scores, question 9 is the easiest question (Table 2). In question 9 mathematical basics like addition, subtraction, dividing and multiplying had to be applied. These basics are needed to understand the application of equations which is assigned in question 2. Therefore, question 9 can be assumed as prior knowledge to question 2.

In the graph of the test set (Figure 14b), question 4 does not have any in-going or out-going edges but in the graph of the train set (Figure 14a) the question is receiving an edge from question 9. In question 4 the student was asked to calculate how much the total expenses of a company were, based on the information that \$15 million of the expenses fill 100 degrees on a circle diagram. The student was required to know how much degrees fit in a circle diagram and how to calculate the total based on a fraction. In none of the other questions the student needed to work with circle diagrams, which explains why there are no (validated) edges to or from question 4.

Furthermore, the questions 5 and 6 are not connected in both graphs but are neither labelled as easy or difficult, based on the accuracy scores (0.42-0.47, Table 2). Both questions are unique compared to the other ones and the examined skill such as calculating ratios or probabilities is not present in other questions. The task for question 5 was to calculate the total number of employees by a given ratio of the male that is working there. In question 6, six unique characters were given and the student had to determine what the probability is for one of the six characters to appear after one other of the six. The questions 5 and 6 also differ from each other, therefore there is not edge between the questions.

5.1.3 Discussion

The *Praxis* test is designed to test the academic math skills of future teachers. To test many skills, questions that require different skills were selected for the test which is illustrated in the analysis of the knowledge graph. The analysis showed that the questions from the *Praxis* are diverse and require diverse skills, this results in a small amount of shared in- and out-going edges between the questions.

However, the knowledge graph does show structure in the knowledge and skills required for the questions 3 and 7-10 (meets R4, section 3). The analysis showed that question 3, a difficult story assignment, took a central position in the knowledge graph and four easier story assignment questions were assumed to be prior knowledge to question 3. Based on the analysis, it can be concluded that the easier story as-

signments (questions 7-10) allowed the student to answer the question by performing a simple operation based on the content of the question or guessing the answer by logical reasoning. To solve question 3, the students had to be able to interpret the story of the question in order to convert it to three separate math problems, solve the problems and then select the correct answer. For this question, the student has to perform several steps (Figure 15) which made it difficult to guess an answer from the multiple-choice answers and test that if that answer is correct.

Because of the diversity, the set of questions is not the best qualified for this analysis. However, the majority of edges are present in both graphs (test and train set). Therefore, the algorithm has been validated because the results of both graphs are comparable. In addition, the results also show that the algorithm is seeking shared skills and not just based on accuracy.

5.2 Trends in International Mathematics and Science Study from IEA

Other data sets that have been analysed in this project, are data sets retrieved from *Trends in International Mathematics and Science Study* (TIMSS) (International Association for the Evaluation of Educational Achievement (IEA), 2009). *TIMSS* is a project of the International Association for the Evaluation of Educational Achievement (IEA) together with the Progress in International Reading Literacy Study (PIRLS) that forms the regular cycle of core studies from the IEA (Mullis et al., 2005; Foy & Olson, 2009). Every four years the knowledge of courses like math and science is examined with a *TIMSS* exam at primary and high schools worldwide. *TIMSS* allows countries to map how well students are performing on math and science courses compared to other countries and compared to the previous distributed *TIMSS* exams.

The retrieved *TIMSS* data is publicly available via the international database on the website of *TIMSS & PIRLS* (*TIMSS 2007*, 2012). Data from the earlier and later years, as well as the 2007 project, can be retrieved from the general website of *TIMSS & PIRLS* (*TIMSS: Trends In International Mathematics And Science Study*, 2021). For this project, the available data sets of the fourth grade (primary school) from 2007 have been analysed. This data set includes student achievement data but also student, teacher, school, and curricular background data from 59 countries and eight benchmarking participants (Mullis et al., 2005). The student achievement data have been used for the analysis. The IEA offers a user guide that explains how the data is designed (Foy & Olson, 2009). Furthermore, the IEA offers an assessment framework publication which explains the assessment design that serves as the basis for the implementation of the *TIMSS* 2007 project (Mullis et al., 2005).

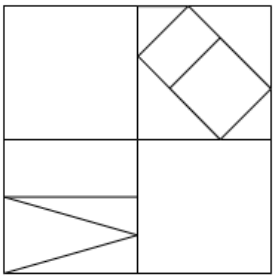
Data for the fourth and eighth grade were available for the courses mathematics and science. In this project, only the mathematical data of the fourth grade is analysed. For the mathematical data of the fourth grade, a total of fourteen workbooks, consisting of approximately 25 assignments per workbook, were available. However, only for the first four workbooks, the content of the assignments was available in the international database. Finally, only the first workbook has been used for the complete analysis to prevent an extensive analysis. Workbook 1 contained 25 assignments with answers of 6445 students. Which countries participated has not been analysed but the differences per country will be taken into account during the data preprocessing. Figure 16 illustrates some of the assignments that are used in the math workbook 1.

In the assessment framework, Mullis et al. (2005) outline the different types of questions that are assessed in the *TIMSS* project. The questions are labelled within a content domain together with a topic

Last year, 91 girls and 84 boys were member of the National Rowing Club. This year the club counts 230 members, and 96 are girls. How many more boys are there this year than last year? Show your work.

(a)

The square below is cut into 11 pieces. Put an X on each of the 2 triangles that are the same size and shape.



(b)

Figure 16: Two questions that are similar to question 3 (a) and question 9 (b) in the math workbook 1 of the TIMSS 2007 project for the fourth grade.

area. The authors state that the content domains represent mathematics subjects that are covered in the *TIMSS* 2007 project. Furthermore, the questions are labelled in the cognitive domain. The authors explain that the students need to be familiar with the assessed content in *TIMSS* 2007, but the students also need to use some cognitive skills. The authors mention that the description of these skills is crucial for the development of the questions in the *TIMSS* 2007 project to ensure that these skills are covered in each content domain. Both domains will be briefly discussed.

For the mathematical questions for grade four in the *TIMSS* 2007 project, the following content domains were included: *Number*, *Geometric Shapes and Measures*, *Data Display*. The aim for the *TIMSS* 2007 mathematics questions for the fourth grade was to offer 50% of the questions in the *number* content domain, 35% of the questions in the *geometric shapes and measures* domain and 15% of the questions in the *data display* domain (Mullis et al., 2005).

The *number* domain expects students to understand ways of representing numbers, the relationships between numbers, and to understand and apply operations to solve problems. The topic areas related to the *number* content domain are: *whole numbers*, *fractions and decimals*, *number sentences* and, *patterns and relationships*. In the assessment framework, for each topic area, a description is given of what a student is expected to be able to do. Per topic area a brief clarification will be given which does not describe the full topic area as explained in the assessment framework (Mullis et al., 2005). In the topic area *whole numbers* the student is expected to represent whole numbers by using words, diagrams, or symbols; compare and order whole numbers and; know and compute (with whole numbers) the basic operations (+, -, ×, ÷). For *fractions and decimals*, the student is expected to be able to recognize fractions as parts of unit wholes; represent fractions using words, numbers or, models and; add and subtract simple fractions and decimals. Furthermore, for *number sentences with whole numbers* the student is expected to be able to find the missing number or operation in a number sentence that is blanked and to model simple situations involving unknowns (expressions or number sentences). Finally, in *patterns and relationships* the student has to be able to extend patterns and find missing terms in them; generate pairs of whole numbers following a given rule and; write or select a rule for a relationship given some pairs of whole numbers.

In the *geometric shapes and measures* domain, the student is expected to be able to identify and analyse properties of geometrical figures (two- and three-dimensional) like lines and angles. Three topic areas are included in this domain: *lines and angles*, *two- and three-dimensional shapes*, *location and movement*. For the first topic area, the student is expected to be able to: measure and estimate lengths and; compare angles by size and draw angles. For the topic area *two- and three-dimensional shapes* the student has to be able to: identify common geometric shapes and classify and compare them and; determine areas and volumes. For topic area *location and movement*, the student has to be able to: recognize and draw figures with line symmetry and their reflections and rotations.

For the final content domain *data display* the authors state that the student has to be able to read and interpret displays of data but also to understand how to organise and display data. The topic areas that are enclosed in this content domain are *reading and interpreting* and *organising and representing*. In the topic area of *reading and interpreting* the student has to be able to: read from tables, graphs, and pie charts; use information from data displays to answer questions that also go beyond directly reading the display. In the second topic area the student is expected to be able to: compare and match different representations of the same data and; organise and display data using tables, pictographs, and bar graphs.

According to Mullis et al. (2005), the student requires certain types of cognitive skills that are divided into three cognitive domains: *knowing*, *applying* and, *reasoning*. In the domain of *knowing* the student is able to know and remember facts, procedures, and concepts. In the domain *applying*, the student has to apply the knowledge and understand the concepts to solve problems and answer questions. In the final domain, *reasoning*, the student goes beyond the solution of a problem to enclose complete contexts and multi-step problems. The cognitive and content domain labels of each question from the math workbook 1 grade four are presented in Table 4 in Appendix B.

5.2.1 Data Preprocessing

The preprocessing of the TIMSS data is more extensive compared to the other analysed data sets in this project. The data was retrieved in a zip file with student answers in SPSS files per country that participated in *TIMSS* 2007 for the fourth grade. First, all files were read as a dataframe by using the `pd.read_spss()` function from the Pandas library. Next, all dataframes were concatenated into one dataframe for the countries together. Only the necessary columns were kept, which were: country, workbook ID and all the assignments. Afterwards, the dataframe was split into dataframes per workbook in which all countries were included.

Next, the student answers were translated to zeros and ones. The answers contained the actual answer and the label *correct response* or *incorrect response* and for the multiple-choice questions, the answers included with a `*` were marked as correct. Whenever the first five lines of a column (assignment) were empty, it was assumed that the assignment was not present in the particular workbook. Therefore, these columns were removed from the workbook. The *TIMSS* data contains many missing values, IEA does not mention the reason for this. The assumptions are that a student might not have been able to answer all questions due to lack of time or lack of focus. Another possibility might be the differences between the countries.

The missing values in a column were substituted by the mean score of the column. This method has been applied to prevent influence on the overall mean of a column, due to the high amount of missing

values. Finally, the workbooks have been equally split into a train and test set. To split the data equally, half of the data entries from each country is represented in the train or test set.

Due to the substitution of the missing values, decimal values ranging between 0 and 1 occurred as scores in the matrix instead of only the two values 0 and 1. Therefore, the knowledge graph algorithm, in particular the `node_greater_than()` function (Listing 1, lines 16-33), was adjusted. To prevent that an answer score like 0.46 was marked as better performing compared to an answer score of 0.45, an adjustment of 20% was added to the comparison. The difference between the answer scores had to be higher than 0.2 in order to determine which answer score was better. To illustrate this, if the answer score `a[i]` = 0.45, then `b[i]` will only be marked as better performing if the score is 0.65 or higher. Whenever the distance between the scores is less than 0.2 and the score of `a[i]` is lower than 0.5, the score is marked as incorrect. The function `node_greater_than()` will neither add 1 to *better*, *worse* or *equal_correct* and not use the particular comparison for the determination of to possibly to be created edge in the knowledge graph.

5.2.2 Knowledge Graph Analysis

The optimal number of clusters for the train set of the *TIMSS* math workbook 1 questions was determined at 622 clusters. The knowledge graph algorithm was applied to the train set with 622 clusters. For `strictness`, a value of 4 has been selected for the best visible relations between the questions. For the test set, the same settings as for the train set were applied to the knowledge graph algorithm. At the value of 4 for `strictness`, both graphs for the test and train set were most similar to each other and the weak connections that appear at a lower `strictness` value are reduced. Table 3 also shows the similarity between both data sets, the accuracy scores are either similar or different from each other by a maximum of 0.3.

Figures 17 and 18 illustrate the results for the train and test set of *TIMSS* math workbook 1. The same nodes (questions) with a high in- and out-degree are present in both graphs. For example, the questions 9 and 12 are questions with a high out-degree in both visualisations, where the questions 3, 10, 22 and 23 have a high in-degree. With this information, the knowledge graphs are validated and the graphs can be analysed in detail.

According to both visualisations (Figure 17 and 18), the questions 9 and 12 are questions that serve as prior knowledge to many other questions in the data set. The accuracy score is ranging between 0.78 and 0.85 for both questions in both data sets. Therefore, the questions could be marked as easy. According to the authors of the questions Mullis et al. (2005), question 9 belongs in the content domain of *Geometric Shapes and Measures* and question 12 belongs to the content domain of *Number* as illustrated in Table 4 in Appendix B. Therefore, the questions will be discussed separately.

Analysing Question 9

Starting with question 9, the question requires a subset of skills that is required for 15 (train set, Figure 17) or 14 (test set, Figure 18) out of a total of 24 questions. Except for question 13, the same questions are receiving edges from question 9 in both graphs. The task in question 9 was to mark triangles with a similar size and shape inside a square that was cut into pieces (Figure 16b). Because question 9 is labelled in the content domain as *Geometric Shapes and Measures* (Table 4, in Appendix B), the expectation is

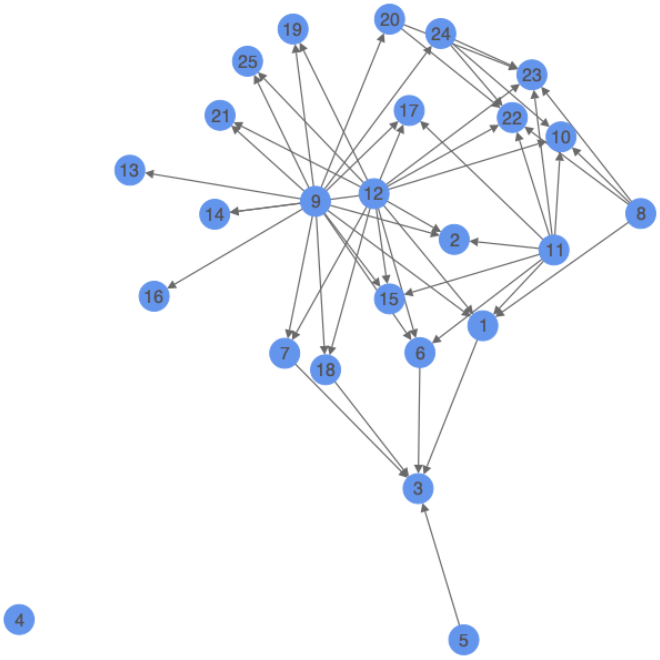


Figure 17: Knowledge graph of the *TIMSS* math workbook 1 train set that was clustered into 622 clusters and a value of 4 for `strictness` was applied. The nodes represent each question in the workbook. The directed edges illustrate how the questions are related to each other: a question is prior knowledge to another if it has an out-going edge to another question.

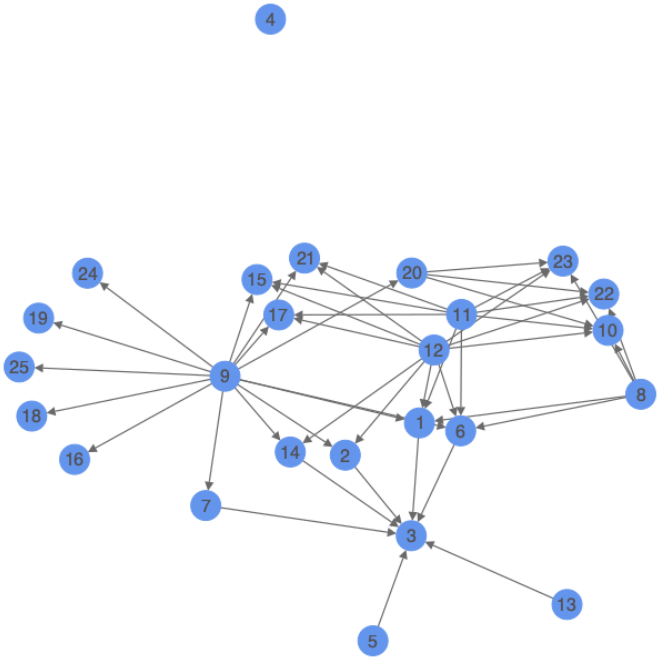


Figure 18: Knowledge graph of the *TIMSS* math workbook 1 test set that was clustered into 622 clusters and a value of 4 for `strictness` was applied. The nodes represent every question in the workbook. The directed edges illustrate how the questions are related to each other: a question is prior knowledge to another if it has an out-going edge to another question.

Data set	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
Train	0.46	0.45	0.22	0.24	0.64	0.49	0.57	0.72	0.85	0.39	0.75	0.8	0.6
Test	0.46	0.46	0.21	0.23	0.64	0.48	0.56	0.72	0.85	0.39	0.75	0.78	0.59

Data set	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	
Train	0.46	0.47	0.57	0.43	0.56	0.54	0.61	0.49	0.41	0.38	0.64	0.55	
Test	0.46	0.45	0.55	0.44	0.57	0.54	0.62	0.5	0.43	0.38	0.61	0.55	

Table 3: Accuracy scores from *TIMSS* math workbook 1 train and test data set.

that question 9 serves as prior knowledge to questions in which the same skill is required and therefore the questions are expected to be similar but more difficult. Table 4 (Appendix B), illustrates that the questions 18-23 are all labelled in the content domain as *Geometric Shapes and Measures*. In both graphs (Figures 17 and 18), question 9 has outgoing edges to the questions: 18, 19, 20 and 21. Questions 22 and 23 receive an in-going edge from question 20 (Figure 19), which implies that the skills required for answering question 9 and question 20 together are required for solving the questions 22 and 23. The connection from 20 to 22 can be explained because the questions 19-22 are follow-up questions (e.g., 1A-1D) for which the same introduction story was used. For the questions 19-22, the subjects were given six cardboard figures of two unique shapes, and the subjects were expected to create four different figures that fulfilled the given requirements. The questions 21 and 22 were found to be more challenging than

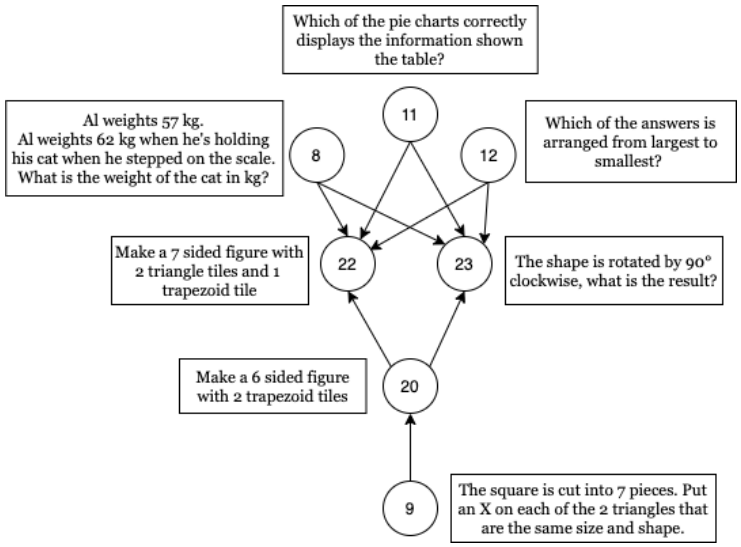


Figure 19: Overview of the questions that serve as prior knowledge to the questions 22 and 23 in both train en test results. Each assignment of the questions is briefly described. Other in- or out-going edges from the questions 9, 8, 11, 12, and 20 are not included for clarity.

the questions 19 and 20 because the figures that the students were asked to create, were more complex than the first two questions. The accuracy scores (Table 3) illustrate this difference in difficulty. Based on the graph, not only the skills required for answering question 9 but also the skills required for answering question 20 (and questions 8, 11, 12 and 24) are a subset of skills required for answering question 22 (Figure 19). In question 20, the subjects were asked to create a six-sided figure with two cardboard figures of the same shape and in question 22 the subjects were asked to create a seven-sided figure with three figures of two type of shapes.

Continuing with the similar content domain questions that receive an incoming edge from question 9: 18 and 23. As the label suggests, in all questions (18-23), the subjects were expected to use and apply their knowledge about geometric figures and shapes. The other questions that receive edges from question 9 have different content domain labels than question 9. The questions that only receive an incoming edge from question 9 and do not have outgoing edges, will not be further discussed (questions 7, 16, 24 and 25). Not enough information is present in the visualisation to reason why these relations were found by the algorithm.

Analysing Question 12

The questions 9 and 12 share about half of the nodes that receive edges from it. Some of these will be explained in detail. Question 12 is labelled as *Number* in the content domain and *Whole Numbers* in the topic area (Table 4, in Appendix B). The task of question 12 was to select one of the 4 answers of which the number sequence was placed in consecutive order. Based on the accuracy score, this was a relatively easy question (Table 3). In the graph of the train set (Figure 17), question 12 has 14 outgoing edges, in the test set (Figure 18) there are 10 edges. The edges that were not present in the graph of the test set, are the questions 7, 18, 19 and 25. These questions will therefore not be discussed for the analysis of question 12.

The questions 1, 2, 6 and 14 all have the label *Number* for the content domain and *Whole Numbers* for the topic area, which is similar to question 12. The questions 1, 2 and 6 also share that they receive an incoming edge from the questions 9, 11 and 12. The latter also accounts for the questions 15 and 17 but these questions are labelled as *Factions and Decimals* in the topic area although they share the content domain label.

The questions 1, 2, and 6 are questions for which basic math skills are content like adding, subtracting, multiplying and dividing. The skill required to answer question 12, which is being able to count, is one of the required skills for answering the questions 1, 2 and 6. The questions 15 and 17 require the same basic skills as the questions 1, 2 and 6 but contain fractions or decimals which causes the difference in labels. It can therefore be stated that the questions 1, 2, 6, 15 and 17 are similar to each other, they share the same incoming edges and the accuracy scores are close to each other (ranging between 0.43 and 0.49).

Analysing Questions With a High In-Degree

Next, for this analysis, the questions with a high in-degree that have not been explained in detail yet will be discussed. The questions 3, 10, 22 and 23 all receive four or more edges from other questions. Only the edges that are present in both graphs of the train and test set will be analysed as the other are assumed to be weak. Question 3 receives an incoming edge from the questions 1, 5 and 6 (Figure 20). The questions 10, 22 and 23 all receive an incoming edge from the questions 8, 11, and 12 (Figure 19

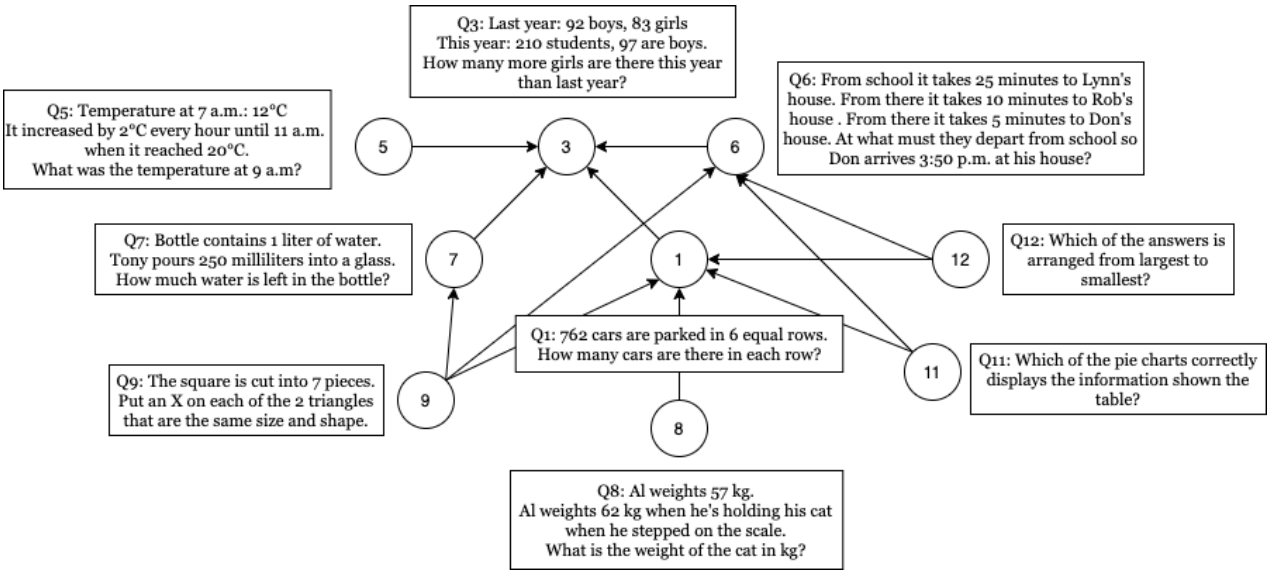


Figure 20: Overview of the questions that serve as prior knowledge to question 3 in both train en test results. Each assignment of the questions is briefly described. Only the incoming edges to question 3 and the questions it receives questions from are visualised.

and 21). Furthermore, the questions 22 and 23 both receive an incoming edge from question 20.

Question 3 is labelled as *Number* for the content domain and *Whole Numbers* for the topic area. The accuracy score for question 3, is the lowest compared to the other questions: 0.22 or 0.21 (Table 3). Based on the accuracy information, it can be stated that question 3 was the most difficult question of all 25 questions. The task of question 3 (Figure 16a) was to determine how many more girls there are present in a school compared to last year. The total number of boys and the total number of girls of the previous year was given, next to the total number of boys that are present this year. First, the student had to determine how many girls are present this year, followed by determining what the difference is from this year compared to the previous year. The question does not clearly state if the boys and girls from the previous year are still present in the school. Therefore this question could be confusing, besides the multiple steps the student has to perform adding and subtracting. The questions 1, 5 and 6 all have the same labels as question 3. These questions have more in common, the questions are story assignments (Figure 20) where the key is to read carefully to give the right answer.

The skills required for solving the questions 8, 11 and 12 were all required for solving question 10 (Figure 21), 22 and 23 (Figure 19). The questions 8 and 12 are both labelled as *Number* for the content domain and *Whole Numbers* in the topic area. Furthermore, question 11 is labelled as *Data Display* (Table 4) in the content domain. The content of question 8 can be marked as a story assignment in which a simple subtraction has to be performed. To solve question 12 the skill to place numbers in consecutive order is required, as explained above. In question 11, the student is asked to select the right circle diagram based on data presented in a table. According to Mullis et al. (2005), question 10 is labelled as *Data Display* in the content domain, like question 11. The incoming edge from question 11 is therefore justifiable. The task of question 10 was to finish the given tally chart based on given data. To work out

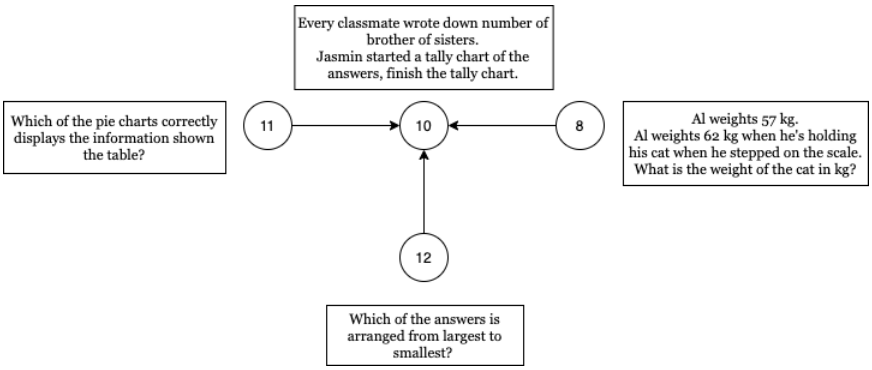


Figure 21: Overview of the questions that serve as prior knowledge to question 10 in both train en test results. Each assignment of the questions is briefly described. The in- or out-going edges from the other questions are not included for clarity.

the task, the student had to know what a tally chart is and had to be able to count to know what how many marks had to be written down. For this reason, the incoming edges from the questions 8, 11 and 12 to question 10, can be explained.

The questions 22 and 23 have been mentioned earlier in this section. Both questions are labelled as *Geometric Shapes and Measures* in the content domain (Figure 19). In question 22 the student was asked to create a 7 sided figure with given figures and in question 23 the student was asked which figure is the result of a rotation of 90 degrees clockwise of a given figure. The incoming edge from question 20 is to be expected because of the shared content domain label, but in particular, for question 22 the edge is expected since it is a follow-up question together with the questions 19-21. Why the questions 8, 11 and 12 are prior knowledge to the questions 22 and 23, cannot directly be explained. It can be assumed that the skills required for answering the questions 8, 11 and 12 are a subset of skills required for answering the questions 22 and 23.

Analysing Question 4

The final question that is analysed, is question 4. This question does neither have in- or outgoing edges, which implied that none of the questions serves as prior knowledge nor that question 4 is prior knowledge to other questions. The accuracy scores of the question are the second-lowest (0.22 and 0.21, Table 3), which marks the question as a difficult question. Question 4 is labelled as *Number* in the content domain and in the topic area the question is labelled as *Pattern and Relationships* (Table 4, in Appendix B). From all questions labelled as *Number* in the content domain, question 4 is the only question with the label *Pattern and Relationships* in the topic area. In the question, the student had to determine which similar rule has been applied to 4 number combinations (Figure 22). First, the student had to determine the difference between the number combinations after which he had to determine which similar rule is applied. The difficulty of this question is that the numbers on the right-hand side of the combinations are all higher than on the left-hand side but the differences are not similar. Since this question was difficult and in none of the other questions a similar approach or skill combination had to be applied, it is justifiable why question 4 is not connected in both graphs.

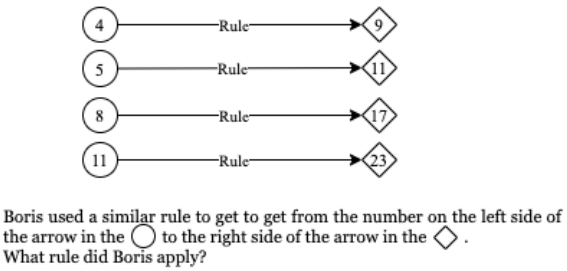


Figure 22: Question that is similar to question 4 in the math workbook 1 of the TIMSS 2007 project for the fourth grade.

5.2.3 Discussion

In both graphs from the train and test set of the *TIMSS* 2007 data set the same questions were central nodes and most edges around these central nodes were similar. In this data set, multiple questions were present between which the content domain or topic area was shared. This implies that the questions are similar and require a similar set of skills. The algorithm was able to find these similarities in required skills, as illustrated in both graphs (Figures 17 and 18). The discovery of the hierarchy meets R4, section 3.

The *TIMSS* data set is designed to test the current mathematics knowledge of students in grade four on primary schools. In the workbook, multiple questions tested the same skills. The algorithm can group the questions with similar labels (content domain or topic area) and therefore proves the assumption of the authors of the *TIMSS* project that these questions share similar labels. Furthermore, the knowledge graph algorithm also discovered questions with different labels that were assumed to be prior knowledge to a question. These connections show that there might be more underlying skills to a question than what is expected by the authors.

For example, the knowledge graph algorithm marks question 9 as a question that is prior knowledge to questions with a similar label in the content domain but also to questions with a different label. The label of question 9 in the content domain is *Geometric Shapes and Measures*. Therefore, it is expected that the question will be connected to others within a similar content domain. This is true for 4 out of 14 questions connected to question 9. The other questions belong to different content domains such as *Number* and *Data Display*. This implies that the skills that are required in question 9, are also necessary for the remaining 10 questions in the content domains of *Number* and *Data Display*.

Furthermore, the application of the knowledge graph algorithm to the *TIMSS* math workbook 1 illustrates that the algorithm is not based on the accuracy scores alone. For example question 4, which was characterised by the second-lowest accuracy score, implying that required prior skills, it was not connected to any other question within the knowledge graph. Manual analysis confirmed this judgement, as question 4 turned out to be one assigned the *Pattern and Relationships* label the topic area. Therefore, question 4 was in fact qualitatively different from the remainder of the knowledge graph.

As a final remark, in the analysis only the labels in the content domain and topic area were discussed.

The skills of the cognitive domains as introduced in the data description and illustrated per question in Appendix B, Table 4 were not discussed. The reason for this is that these cognitive skills are more general skills that can be applied in each content domain and topic area. Additionally, the questions labelled within the same cognitive domain were marked in varying degrees of difficulty based on the analysis. To illustrate, in the cognitive domain *reasoning*, the student is required able to go beyond the solution of a problem to enclose complete contexts and multi-step problems. This implies that the cognitive domain *reasoning* can be marked as the most challenging skill. For example, for the questions 3, 4, 6, and 22 it was indicated that the cognitive skill *reasoning* was required. As discussed in the analysis, these questions were receiving either none or many edges and were marked as more challenging questions. However, the content of these four questions varied. On the other hand, for question 11 the cognitive skill in the *reasoning* was required as well. Based on the results of the analysis, question 11 is marked as an easier question requiring skills that are prior skills needed for other questions. Although these questions might all require the abstract, cognitive skill *reasoning*, the required content skills differ as demonstrated in the analysis. This project highlights content skills. Therefore, the three cognitive domains were not further analysed.

The findings of the analysis of the *TIMSS* math workbook 1, suggest that applying the knowledge graph algorithm to a larger data set with more homogeneous questions, leads to more robust results. Using such a data set would result in highly similar and therefore directly comparable train and test set knowledge graphs, allowing for meaningful analysis. Furthermore, the findings show that the knowledge graph algorithm can be used to discover similarities the content of questions and the skills that are required for answering them.

5.3 Basic Math Skills from SOWISO

Another data set that has been examined in this project, is a data set retrieved from SOWISO (*SOWISO Basic Maths*, 2021) via the University of Groningen. SOWISO is a Dutch organisation that offers e-learning modules to practice and examine introductory mathematics at a university level. The University of Groningen utilises the software for first-year *Basic Math Skills* Bachelor courses at the Faculty of Science and Engineering.

The retrieved data is data from exams and resits from the 'Basic Match Skills' courses at the University of Groningen of the academic years 2019-2020 and 2020-2021. A total of nine unique courses, consisting of 21 unique versions are part of the data. The number of students that participated in an exam version ranged from 26 to 315. An important remark of this data is that the data is randomised per

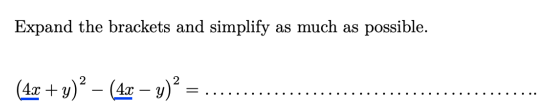


Figure 23: This figure is one of the assignments from the SOWISO Basic Math Skills tests (*SOWISO Basic Maths*, 2021). Both the digits 4 are underlined with a blue line. These digits are randomised for each unique student. The rest of the assignment remains the same but the correct answer differs. The required skill for the question remains the same.

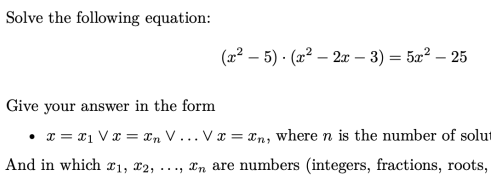


Figure 24: This figure illustrates an example of an assignments from the SOWISO Basic Math Skills tests (*SOWISO Basic Maths*, 2021).

student, this implies that each student completed questions where the type of question was similar but different digits were used. Figure 23 illustrates which digits are randomised in questions. Although the digits are different, the assumption is that the same skill is tested.

To further clarify what kind of assignments are part of the Basic Math Skills exams, another example is given in Figure 24. As SOWISO states about their Basic Math package: the assignments are a "*good fit for introductory math courses on both university and college level. The assignments are also a good fit for high school students (upper secondary)*" (*SOWISO Basic Maths*, 2021).

5.3.1 Data Preprocessing

The data has been preprocessed in Python into a similar format as the simulated data set. This implies a NumPy matrix with each student as a row and their answers to the assignments in the columns where each question represents one column. First, the data was read into a Pandas dataframe. Multiple rows were present per student, in each row the answer to a unique assignment was given. The data set had to be grouped per student to shift the multiple rows for each question into one row with the answers per question in the columns. The next step was to split the data per unique Basic Math Skills test and then to version type. The final step was to convert the Pandas dataframe into a NumPy matrix and save it as a CSV-file.

5.3.2 Conclusion

During the preprocessing of the multiple data sets and analysing the assignments per exam version, it seemed that the different exams could not be combined into one data set. The questions differed too much from each other. The expectation was that the different versions of the exams would be similar enough to enable them to merge. However, some exams contained more assignments than others and the type of assignments also differed. Therefore, the data sets were not merged.

The largest data sets ranged between 315 and 266 students. By splitting the data into a train and test set, the data would only contain about 150 students. For the analysis, that amount of data is not enough. The knowledge graph algorithm has been applied to both train and test sets but with similar settings, the returned graphs differed too much from each other to perform an analysis. For the train set, 17 edges were drawn and one question was not connected, in the test graph only 10 edges were drawn and four questions were not connected.

6 Discussion

This project aimed to develop an unsupervised machine learning algorithm that can extract underlying skills from accuracy scores of a mathematical problems by calculating a partial ordering. Such an algorithm can aid in the development of cognitive tutors since the required skills could be extracted automatically instead of having to be deduced by the cognitive tutoring system designer. This project tackled this goal by the development of a new, tailored machine learning approach.

An unsupervised machine learning algorithm, the knowledge graph algorithm, has been developed that calculates a partial ordering on the accuracy scores from student answers to math problems. The knowledge graph algorithm has been applied to a simulated data set. The results were visualised in a directed acyclic graph where the questions or students are presented as nodes and the edges represent the relations. As result of applying the knowledge graph algorithm, relations between the questions were determined which was traced back to the possible eight skill combinations of the data. However, due to the high number of questions (100) that formed the data set, it was impossible to visualise the hierarchy in a clear and concise manner. Therefore, the agglomerative hierarchical clustering method was examined in combination with the knowledge graph algorithm. The students were clustered based on similarities in their solutions to the problems. In the simulated data set, eight different skill combinations were possible for a student to have. Therefore, the students were clustered into these eight different skill combinations. Next, the knowledge graph algorithm was applied to the clustered data. Additionally, in the visualisation the eight skill combinations were presented. The assumption was that the hierarchy of the eight skill combinations, having zero to maximum three skills, would be visible. The assumption was found to be true and therefore the knowledge graph algorithm was validated.

In the simulated data, the number of present skill combinations was known. In real data sets, the skill combinations are unknown. This project attempted uncovering such skill combinations from multiple data sets. Therefore, the added step of clustering the data before applying the knowledge graph algorithm, raised the question of how many clusters the data should be grouped into. Although the specific number of clusters did not need to be determined prior to applying agglomerative hierarchical clustering, it was necessary to determine at what number of clusters the resulting dendrogram should be cut off. A method based on the AIC value was developed to determine the optimal number of clusters within a data set. This method aimed to maximize the number of clusters to preserve as many differences between the students as possible. It was assumed that clustering the students within a data set allowed the knowledge graph algorithm to be more efficient because of the lower number of dimensions as well as by emphasising the performance of students who performed differently from the rest.

The application of the knowledge graph algorithm to the clustered data sets showed that relations between questions were detected based on the skills required to complete them. Although not all the connections could be explained, the majority of the connections could have been explained by analysing the content of specific questions. This project did not succeed at uncovering the specific skills or skill sets from accuracy scores of mathematical exam data sets. Nevertheless, the knowledge graph algorithm was able to uncover a hierarchy of the questions that allows discovering what kind of questions are prior knowledge to a question. Furthermore, an analysis showed that the discovered relations were not based purely on the overall accuracy scores of questions. Instead, the algorithm did in fact build relations corresponding to the similar skills required to answer questions with a shared topic.

6.1 Limitations

6.1.1 Desired Data for Knowledge Graph Algorithm

The project realised for the purpose of this thesis was one of an exploratory nature wherein it was unknown what kind of mathematical exam data sets were needed to uncover skills from it. Therefore, it was decided to gather existing data instead of performing an experiment. The results of this project showed that data with a high set of questions and a high group of students, for example the *TIMSS* 2007 project, resulted in the most robust results. Applications to data like the *Praxis* test retrieved from ETS still allowed for visualisation of some relation between the questions but because of the small amount of questions, not many edges could be drawn between them. While small sets of questions such as in the *Praxis* test, performed by a relatively large group of students, allow the algorithm to determine a knowledge graph, data sets such as the Basic Math Skills test retrieved from SOWISO do not. A high amount of students is required to create robust results. In the next section (section 6.2), a description is given of the kind of data the knowledge graph algorithm can be successfully applied to.

6.1.2 Interpretation of Results Knowledge Graph Algorithm

As mentioned earlier, this project aimed to uncover skills from mathematical exam data. The hierarchies determined by the knowledge graph algorithm could not be labelled because there was no ground truth. Hence, a subjective interpretation of the relations between the questions was performed based on the validated knowledge graphs.

6.2 Future Work

6.2.1 Clustering of Questions Before Application Knowledge Graph Algorithm

In this project, the data has been clustered on student level prior to the application of the knowledge graph algorithm. This method aided in discovering a hierarchy of questions but did not allow for uncovering skills from that hierarchy. An alternative approach that could be explored in the future would be that of clustering the data on the level of questions. This approach could not be applied to the data sets in this project due to the small set of questions present. Clustering on the questions within a data set would, after applying the knowledge-graph algorithm, result a directed acyclic graph consisting of clusters of questions presented as nodes and the relations between the clusters of questions as edges. These clusters of questions could help to better identify the skills required for such a cluster, as questions with a shared topic are clustered together, making it easier to determine what the tested set of skills is.

6.2.2 Recommended Data for Knowledge Graph Algorithm

The application of the knowledge graph algorithm to different data sets showed that a larger set of questions and a large group of students resulted in a more robust result and similar train and test set knowledge graphs. For example, compared to the *Praxis* test from ETS, the application of the knowledge graph algorithm to the *TIMSS* data showed more relations (edges) between the questions (nodes). In the *TIMSS* data, many assignments that require similar skills were present that resulted in expected edges between

these questions in the knowledge graphs. Meanwhile, the steps needed for completing the questions in the *Praxis* test from ETS were characterised by higher variability from each other and the number of questions was low, which resulted in a low number of edges between the questions. The uniqueness as well as the number of questions can be explained due to the fact that the *Praxis* test is an exam. Therefore, to test as most different skills from the student like in the *Praxis* test, unique questions have to be assigned. In contrast to the *Praxis* test, the *TIMSS* data set is a workbook that is meant to teach a student new skills. Therefore, questions might overlap so that the same skill is examined and strengthened repeatedly.

For future research, it is recommended to use data with a high number of questions that have overlap in the skills that are examined. A perfect example of such a set is a workbook wherein various difficulty levels of questions are used and the questions overlap in the skill content. The knowledge graph algorithm should show the hierarchy of these questions where the easier questions serve as prior knowledge to the more challenging questions. Data from exams is most likely to contain unique questions, which challenges the knowledge graph algorithm to detect a relation between them.

Suggestions for existing data sets are the data sets from the *TIMSS* project. In this project, only the data from the first grade four math workbook is analysed. For the fourth grade, 14 workbooks are available for mathematics but also for science. Furthermore, for the project of 2007, data for grade eight for both mathematics and science is available as well. The *TIMSS* project has been performed every four years since 1995. Data of all projects is available at: `https://timssandpirls.bc.edu/timss-landing.html`.

6.2.3 Knowledge Graph Algorithm Applied in Personalised Learning Environment

Although this project did not succeed in uncovering skills from data, the current knowledge graph algorithm could be of value if applied in a personalised learning environment. In such an environment, the skills themselves do not have to be known to coach a student in topics that they find challenging. With the knowledge graph, a hierarchy of the questions is known. A group of students must perform the set of questions first to create the knowledge graph. The scores of a student in the personalised learning environment can then be compared to the knowledge graph to determine which set of questions, and thus skills, the student finds challenging. The questions which are at a lower level in the hierarchy can then be assigned to the student to practice these skills.

6.2.4 Further Development of Knowledge Graph Algorithm for Cognitive Tutors

As is, the developed algorithm is not able to determine a definition of the skills required to successfully tackle a given problem. Complete defined skills would be the best option for the development of cognitive tutors, therefore the algorithm should be further developed. However, the current version of the knowledge graph algorithm could still be a base for the development of a cognitive tutor. This version of the knowledge graph algorithm is a start for objectively identifying skills instead of a subjective approach of the designer of the cognitive tutor that determines the skills based on own experience and knowledge.

7 Conclusion

The composed research question for this project: “How can skills required for solving mathematical problems be determined by a machine learning algorithm which only uses the accuracy scores from the mathematical exam data?”, cannot be completely answered based on the results. The developed knowledge graph unsupervised machine learning algorithm did not succeed in determining the required skills for a mathematical exam from accuracy data. However, the knowledge graph algorithm can successfully determine a hierarchy of the questions from a mathematical exam. This is accomplished by calculating a partial ordering on the questions using the accuracy scores. The hierarchy of the questions is visualised in a directed acyclic graph.

In section 3, five requirements (listed below) were stated that the developed algorithm should meet. The knowledge graph algorithm meets 4 out the 5 requirements. Only requirement 5 (R5) is not met by the knowledge graph algorithm as described above. Although the algorithm cannot uncover skills from an existing mathematical exam data set, it can uncover skills from a simulated data set where the possible skill combinations are known. This implies that the knowledge graph algorithm needs further development to determine skills in experimental data sets.

> R1: The algorithm has to be able to find skill sets in a simulated data set where the skill set is known.
>
> R2: The algorithm has to be able to show a hierarchy of the skill sets in a simulated data set where the skill set is known.
>
> R3: The algorithm has to be developed in a way that it can be applied to any data set.
>
> R4: The algorithm has to be able to show a hierarchy of the questions from existing mathematical exam data sets.
>
> R5: The algorithm has to be able to uncover skills from existing mathematical exam data sets.

The results of this project bring us a step closer to the objective determination of skills in math problem-solving tests. Nonetheless, more research has to be conducted to get closer to identifying specific skills.

References

About ETS. (2021). Retrieved 2021-11-02, from https://www.ets.org/about

About the Praxis Tests. (2021). Retrieved 2021-11-02, from https://www.ets.org/praxis

Aho, K., Derryberry, D., & Peterson, T. (2014). Model selection for ecologists: the worldviews of aic and bic. *Ecology*, *95*(3), 631–636.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, *19*(6), 716–723.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (2009). *How can the human mind occur in the physical universe?* Oxford University Press.

Anderson, J. R., & Corbett, A. (1995). Knowledge decomposition and subgoal reification in the act programming tutor. In *Proceedings of aied '95, 7th world conference on artificial intelligence in education* (pp. 1–7).

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The journal of the learning sciences*, *4*(2), 167–207.

Boehmke, B., & Greenwell, B. (2019). *Hands-on machine learning with r*. Chapman and Hall/CRC.

Christofides, N. (1975). *Graph theory: An algorithmic approach (computer science and applied mathematics)*. Academic Press, Inc.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. (2008). *Introduction to information retrieval*. Cambridge University Press.

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, *Complex Systems*, 1695. Retrieved from https://igraph.org

Dendane, A., & Math, U. (2009). Skills needed for mathematical problem solving. In *tenth annual research conference, united arab emirates university, al-ain, united arab emirates.*

Elo, A. E. (1978). *The rating of chessplayers, past and present*. Arco Pub.

Fleming, N. (2021). *After Covid, will digital learning be the new normal?* Retrieved 2021-11-10, from https://www.theguardian.com/education/2021/jan/23/after-covid-will-digital-learning-be-the-new-normal

Foy, P., & Olson, J. F. (2009). *Timss 2007 user guide for the international database.* TIMSS & PIRLS International Study Center, Lynch School of Education, Boston College.

Fuchs, L. (1963). *Partially ordered algebraic systems (international series of monographs on pure and applied mathematics, vol. 28), ix+ 229 pages.* Oxford—London—New York—Paris, Pergamon Press.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. doi: 10.1038/s41586-020-2649-2

Hawkins, D. M., Basak, S. C., & Mills, D. (2003). Assessing model fit by cross-validation. *Journal of chemical information and computer sciences*, *43*(2), 579–586.

Hinton, G. E., & Sejnowski, T. J. (1999). *Unsupervised learning: foundations of neural computation*. MIT press.

Hoekstra, C., Martens, S., & Taatgen, N. A. (2020). A skill-based approach to modeling the attentional blink. *Topics in Cognitive Science*, *12*(3), 1030–1045.

International Association for the Evaluation of Educational Achievement (IEA). (2009). *Timss 2007 assessment copyright © 2009.* TIMSS & PIRLS International Study Center Lynch School of Education Boston College. (data retrieved from TIMSS & PIRLS, https://timssandpirls.bc.edu/TIMSS2007/idb_ug.html)

Kassambara, A. (2018). *Determining the optimal number of clusters: 3 must know methods.* Retrieved 2021-09-27, from https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/#average-silhouette-method

Kassambara, A., & Mundt, F. (2020). factoextra: Extract and visualize the results of multivariate data analyses [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=factoextra (R package version 1.0.7)

Kaufman, L., & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis* (Vol. 344). John Wiley & Sons.

Klinkenberg, S., Straatemeier, M., & van der Maas, H. L. (2011). Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, *57*(2), 1813–1824.

Li, C., & Lalani, F. (2020). *The covid-19 pandemic has changed education forever. this is how.* Retrieved 2021-11-10, from https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/

Lloyd, S. (1957, 1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*(2), 129-137. doi: 10.1109/TIT.1982.1056489

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., & Hornik, K. (2021). cluster: Cluster analysis basics and extensions [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=cluster (R package version 2.1.2 — For new features, see the 'Changelog' file (in the package source))

McKinney, W. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th python in science conference* (Vol. 445, pp. 51–56).

McQuitty, L. L. (1957). Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies. *Educational and psychological measurement*, *17*(2), 207–229.

McQuitty, L. L. (1961). Elementary factor analysis. *Psychological Reports*, *9*(1), 71–78.

Mouselimis, L. (2020). Clusterr: Gaussian mixture models, k-means, mini-batch-kmeans, k-medoids and affinity propagation clustering [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=ClusterR (R package version 1.2.2)

Mullis, I. V., Martin, M. O., Ruddock, G. J., O'Sullivan, C. Y., Arora, A., & Erberber, E. (2005). *Timss 2007 assessment frameworks.* ERIC.

NOS. (2020). *Tentamens gestaakt vanwege storing in systeem rijksuniversiteit groningen.* Retrieved 2020-25-11, from https://nos.nl/artikel/2355038-tentamens-gestaakt-vanwege-storing-in-systeem-rijksuniversiteit-groningen.html

R Core Team. (2017). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from https://www.R-project.org/

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic bulletin & review*, *14*(2), 249–255.

Rokach, L., & Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook* (pp. 321–352). Springer.

Russell, B. (1901). Is position in time and space absolute or relative? *Mind*, *10*(39), 293–317.

SOWISO basic maths. (2021). Retrieved from `https://sowiso.nl/en/courses/basic-maths/`. (Accessed: 2021-09-15)

Suzuki, R., & Shimodaira, H. (2006). Pvclust: an r package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, *22*(12), 1540–1542.

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, *18*(4), 267–276.

Thulasiraman, K., & Swamy, M. (1992). 5.7 acyclic directed graphs. *Graphs: theory and algorithms*, *118*.

TIMSS 2007. (2012). Retrieved 2021-11-02, from `https://timssandpirls.bc.edu/TIMSS2007/index.html`

TIMSS: Trends In International Mathematics And Science Study. (2021). Retrieved 2021-11-02, from `https://timssandpirls.bc.edu/timss-landing.html`

Trouw. (2020). *Duizenden tentamens uva afgelast door computerstoring.* Retrieved 2020-25-11, from `https://www.trouw.nl/onderwijs/duizenden-tentamens-uva-afgelast-door-computerstoring~b6a57e6e/`

Van Rijn, H., van Maanen, L., & van Woudenberg, M. (2009). Passing the test: Improving learning gains by balancing spacing and testing effects. In *Proceedings of the 9th international conference of cognitive modeling* (Vol. 2, pp. 7–6).

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, *58*(301), 236–244.

Wilson, R. J. (1979). *Introduction to graph theory*. Pearson Education India.

Appendices

A AIC Method for Agglomerative Hierarchical Clustering

```
def AIC_HC(matrix, max_cluster):
    """ This function determines the optimal number of clusters for a dataset in hierharchical clustering. The optimal number is determined by calculating the AIC value for each number of clusters in the range of the given maximum. The optimal AIC is the first minimum value in the rangs of AIC. This is the AIC that was prior to the first positive result in the difference matrix of the AIC values.

    Input:
    matrix: a Numpy matrix of the data that needs to be clustered.
    max_cluster: integer, the maximum number of clusters to be observed.

    Note1: When the function returns the given number for maximum clusters as optimal number of clusters, it is recommended to run the function again with a higher number for the input max_cluster. It could be that the optimal number of clusters is a bit higher than expected.

    Note2: the function can return the below warning, sometimes multipe times. The warning can be ignored.
    -/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
    # Remove the CWD from sys.path while we load stuff.-
    """
    AICs = {}
    print('Running iterations for', max_cluster, 'clusters...')

    # a loop for each number (n_cluster) in the range of maximum number of clusters
    for n_cluster in range(1, max_cluster+1):
        # agglomerative hierachical cluster the data into n_cluster
        # note: use memory = 'mycachedir' to remember the created dendrogram. The speeds up the function for the next iteration of the loop.
        hclustering = AgglomerativeClustering(n_clusters=n_cluster, memory='mycachedir', compute_full_tree=True).fit(matrix)
        hc_labels = hclustering.labels_
        # use the function get_cluster_means to receive the cluster means.
        hc_centers = get_hcluster_means(n_cluster, hc_labels, matrix)

        # translate which student is assigned to which cluster by creating a dictonary in the dict: the students as keys and the cluster labels as value.
        student_dict = {}
        for student in range(matrix.shape[0]):
            student_dict[student] = hc_labels[student]
```

```
        # create a dictionary for the clusters with the cluster number as key and a list of the students as value
        clust_dict ={}
        for cluster in range(max_cluster+1):
            clust_keys = [key for key in list(student_dict.keys()) if student_dict.get(key) == cluster]
            clust_dict[cluster] = clust_keys

        # calculate the Within Cluster Sum of Squares (WCSS) for each cluster
        means = []
        for key in list(clust_dict.keys()):
            dist_clust = []
            for student in list(clust_dict.get(key)):
                dist = distance.euclidean(hc_centers[key],matrix[student])
                dist_clust.append(dist**2)
            means.append(sum(dist_clust))
        # calculate the total Withing Cluster Sum of Squares
        tot_withins = sum(means)
        m = matrix.shape[1] # number of rows/students
        n = matrix.shape[0] # number of columns/questions
        k = len(clust_dict.keys()) # number of cluster centers

        # calculate the AIC
        AIC = (tot_withins + (2*m*k))
        AICs[n_cluster] = AIC
    print('Calculating optimal amount of clusters')

    # create a difference matrix between the AIC values
    for diff in np.diff(list(AICs.values())):
        # whenever the difference is positive, it means the optimal number of clusters is the previous index
        if diff > 0:
            optimal = list(np.diff(list(AICs.values()))).index(diff)
            break
        else:
            optimal = None

    # return the result
    if optimal == None:
        print('No positive difference was found in the distance matrix. Run the function again with a higher maximum number of clusters.')
    else:
        # add 1 to optimal because of Python index starting at 0
        print('Optimal amount of clusters is', optimal+1)
    print(Counter(hc_labels))

    return
```

Listing 5: `AIC_HC()` function.

B Item Information *TIMSS* Math Workbook 1

Question	Item ID	Content Domain	Topic Area	Cognitive Domain	Item label
1	M031286	Number	Whole Numbers	Knowing	762 cars parked in 6 equal rows
2	M031106	Number	Whole Numbers	Reasoning	Mano's missing digit
3	M031282	Number	Whole Numbers	Reasoning	Number of boys and girls in a school
4	M031227	Number	Pattern and Relationships	Reasoning	Sean's rule to transform numbers
5	M031335	Number	Whole Numbers	Reasoning	Temperature increase at 9am
6	M031068	Number	Whole Numbers	Reasoning	Time Don, Rob, Lynn to leave school
7	M031299	Number	Whole Numbers	Applying	Water in bottle after pouring 250 ml
8	M031301	Number	Whole Numbers	Applying	Al wanted to weigh his cat
9	M031271	Geometric Shapes and Measures	2-and 3-Dimensional	Knowing	The square is cut into 7 pieces
10	M031134	Data Display	Organizing and Representing	Applying	Jasmin's tally chart
11	M031045	Data Display	Organizing and Representing	Reasoning	Types of trees in pie charts
12	M041014	Number	Whole Numbers	Knowing	Arranging numbers in order
13	M041039	Number	Whole Numbers	Applying	Number of tables needed
14	M041278	Number	Whole Numbers	Knowing	Multiply 53 and 26
15	M041006	Number	Fractions and Decimals	Knowing	Fraction of the rectangle shaded
16	M041250	Number	Fractions and Decimals	Knowing	Subtract 5.3 - 3.8
17	M041094	Number	Fractions and Decimals	Applying	How much money has Bob left with
18	M041330	Geometric Shapes and Measures	2-and 3-dimensional shapes	Applying	Perimeter of the rectangle

Question	Item ID	Content Domain	Topic Area	Cognitive Domain	Item label
19	M041300A	Geometric Shapes and Measures	2-and 3-dimensional shapes	Applying	Make a 4-sided figure
20	M041300B	Geometric Shapes and Measures	2-and 3-dimensional shapes	Applying	Make a 6-sided figure
21	M041300C	Geometric Shapes and Measures	2-and 3-dimensional shapes	Reasoning	Another 6-sided figure
22	M041300D	Geometric Shapes and Measures	2-and 3-dimensional shapes	Reasoning	Make a 7-sided figure
23	M041173	Geometric Shapes and Measures	Location and Movements	Knowing	Shape after 90 degree rotation
24	M041274	Data Display	Organizing and Representing	Applying	Put the correct number on the scale
25	M041203	Data Display	Organizing and Representing	Reasoning	Draw a bar to show Alonso's point

Table 4: Item information of the *TIMSS* Math Workbook 1 questions.