NTNU
Norwegian University of
Science and Technology

Combining Elo Rating and Collaborative Filtering to improve Learner Ability Estimation in an e-learning Context

Ole Halvor Dahl
Olav Fykse

Master of Science in Informatics
Submission date: May 2018
Supervisor: Trond Aalberg, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

This master thesis details a proof-of-concept software system for estimating student abilities. The system combines Elo rating and collaborative filtering in order to present students with tasks that best reflect their knowledge. The system was designed and developed successfully. A substantial amount of 8th-grade math tasks were added to the system, and the system was tested on a relevant student population without technical issues. However, the size of the experiment was not sufficient to reach a conclusion regarding whether or not the Elo rating algorithm performs better when combined with collaborative filtering.

Sammendrag

Denne masteroppgaven omhandler et proof-of-concept programvaresystem for å måle elevers ferdigheter innen matematikk. Systemet kombinerer Elo rating og collaborative filtering for å presentere elever med oppgaver som best viser frem kunnskapen deres. Design og utvikling av systemet var vellykket. En vesentlig mengde matematikkoppgaver for åttende klasse ble lagt til i systemet, og systemet ble testet uten problemer på en gruppe studenter. Størrelsen på eksperimentet var ikke stor nok til å kunne konkludere med om Elo rating algoritmen fungerer bedre når den kombineres med Collaborative filtering.

Acknowledgements

This thesis was submitted to the Department of Computer Science, Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology in partial fulfillment of the requirements for the degree Master of Science in informatics.

We want to express our gratitude to the people who have been very helpful and supportive during this project. First and foremost we would like to thank our supervisor Trond Aalberg for his guidance and helpful advice, which has improved the quality of this work.

We would like to thank Boban Vesin and Katerina Mangaroska at the Department of Computer Science, for valuable input and discussions. We would also like to thank Trygve Solstad and Hermund Andre Torkildsen at the Department of Teacher Education for great cooperation and their time to conduct experiments.

Contents

List of Figures

List of Tables

List of Listings

Abbreviations

ALS	Adaptive Learning Systems
BTL	Bradley Terry Luce model
CAP	Computerized Adaptive Practice
CAT	Computerized Adaptive Testing
CF	Collaborative filtering
CSV	Comma-separated values
ERS	Elo Rating system
GRE	Graduate Record Examination
GUI	Graphical User Interface
HSHS	High-Speed High-Stakes
IRT	Item Response Theory
ITS	Intelligent Tutoring Systems
k-NN	k-Nearest Neighbor algorithm
LMS	Learning Management System
MAE	Mean Absolute Error
NCME	Council on Measurement in Education
NMF	Non-negative Matrix Factorization
RMSE	Root-mean-square error
SAT	Scholastic Aptitude Test
SVD	Singular Value Decomposition

Chapter 1

Introduction

In this master thesis, we seek to improve the process of testing students' ability by using a combination of the Elo rating system, a system for calculating relative skill [1] and collaborative filtering, which is a recommender system algorithm, normally used to make predictions based on similarities between users [2].

The goal is to present individual students with mathematical tasks that most accurately reveal their skill level. This will be achieved by using the Elo rating algorithm to find tasks that are close to the student's estimated abilities. After having found tasks that have a fitting difficulty, collaborative filtering is used to predict which of those tasks the student is most likely to answer correctly. This task will then be presented to the student. The desired effect of this is that the collaborative filtering algorithm will present each student with a task they understand, within a desired difficulty, giving them a better chance at displaying their knowledge.

One of the biggest obstacles in todays teaching methods is that the same curriculum and learning material is presented to all students. However, students possess different skills and enter education with different knowledge. Therefore, the ability to personalize and adapt the learning process to their needs and skills is a prerequisite for tailoring successful students that learn based on their current knowledge and advance based on their skills. Having a system that combines the

Elo rating method and collaborative filtering, might aid teachers to better understand students behavior and skills. It may also be used to optimize the learning process, by supporting interventions and aiding scaffolding in learning tasks. The system developed in this thesis is a proof-of-concept and will focus on fractions, but should be applicable to most other concepts and fields as well.

The combination of Elo rating and collaborative filtering may be beneficial to both students and teachers. By more accurately knowing the abilities of the students, a teacher will be able to tailor lectures based on the abilities of the students. Collaborative filtering may also be used to identify students who have similar abilities or lack of abilities. These groups may be used to create study groups or groups who receive a tailored curriculum. Having an accurate estimate of the student's ability also allows for adaptive e-learning environments to be used more effectively, as an estimate of ability is crucial when choosing which learning material to present the student for learning purposes. More accurate grading of students will also be helpful outside of the educational setting, for example when the student is applying for jobs or higher education, as grades often are criteria for admission. This emphasizes the importance of setting grades accurately.

1.1 Problem definition and hypothesis

While grades are the most important metric for grading students in the school system, they are usually not an exact representation of the true skill of the student. Using tests is the standard way of determining the skill of a student. A teacher puts together a set of tasks with the goal of choosing tasks that most accurately reveal whether the students have mastered the concept that is tested. The main problem this thesis is trying to explore is the problem of every student having different ideal tasks needed to best display their knowledge. Let's say you want to test whether a student knows how to divide a number by another number. There are several ways of representing this concept as a task. Some alternatives are:

1. "Calculate $\frac{10}{5}$"
2. "You buy ten pieces of cake and have four friends with you, how many pieces do you get?"
3. Display a picture of a chocolate bar divided into ten squares and ask the student to color a fifth of the squares.

One of the main challenges when creating and picking tasks to be used in testing is figuring out what the task is testing. The goal is usually to test only one concept, like division. Still, tasks like task number 2 listed above may test other concepts as well. Task number 2 tests, among other things, the ability to read. This greatly disfavors students who are dyslectic. Other smaller properties of tasks may also affect the outcome of a task. Task number 2 even tests the willingness to share food. There is usually not a definitive answer to which representation is best, as different students have different performance biases. We aim to develop a system that lessens the impact of students having different performance biases.

Another problem we wish to address is the challenge of item difficulty estimation. The difficulty of a task is possible to estimate by looking at how skilled you generally need to be to solve it. However, the difficulty is subjective. One task may be easy for one student and hard for another student, even though the students are equally skilled within the current concept. The system proposed in this thesis accounts for both the average difficulty as well as the individual subjective difficulty of tasks. The Elo rating of the task is based on overall difficulty, while collaborative filtering is used to estimate the difficulty of a specific task for a specific student.

Hypothesis:

- A combination of Collaborative Filtering and Elo rating will better represent the skill of a student, compared to using only Elo rating.

The first presumption of this hypothesis is that a better representation of the skill results in a higher Elo rating. We believe this is a fair presumption because the

Elo algorithm works independently from collaborative filtering, meaning that any increase in Elo rating is the result of a student solving a task with a higher rating, i.e. a harder task.

An analogy of this is: Consider a scenario where you are a parent who has a child that will have their knowledge of fractions tested. The child will be asked questions regarding fractions in front of judges who will estimate the child's ability based on the correctness of the child's answers. You, as the parent, are choosing which questions to ask the child. You know that your child loves to play hide-and-seek and is exceptionally good at calculating fractions when playing. You therefore ask your child "If there were four people hiding, and you found two, what fraction of players are still hiding?". This is, in essence, what we are trying to achieve using collaborative filtering and Elo rating, presenting students with problems that best allow them to display their knowledge.

The second presumption of our hypothesis is that the curriculum used is narrow and specific. If the curriculum is to wide, the students may be presented only with the concepts they master and never be presented concepts they do not master. This proof-of-concept system only looks at one concept, fractions.

1.2 Approach

We wish to address this problem by completing the following steps:

- Design and develop a hybrid method that combines the collaborative filtering algorithm and the Elo rating algorithm.
- Implement the method in an already existing e-learning system.
- Run simulations and test different recommender algorithms.
- Conduct an experiment in two phases with test users.
- Evaluate and analyze the results.

The experiment we will conduct will include two different groups; the control group and test group. The test group will be presented with tasks based on their current Elo rating and recommendations from collaborative filtering. The control group will be presented with tasks based only on their current Elo rating.

We will utilize an existing system for e-learning called Matistikk to implement the combination method described in this thesis. Matistikk does not provide any form of adaptivity, and it is our job to implement this as a feature in the already existing system. The details of the implementation itself are described in Chapter 4.

The Elo rating algorithm will be used to automatically quantify the skill of students, as well as the difficulty of tasks. This will be done by giving both users and tasks an initial rating. For the students, this will be 1300. This is below the expected average, to try keeping the students from being demotivated. For the tasks, this initial rating was set based on the authors' estimates combined with guidelines from math textbooks. The tasks were set to range between 1200 and 1800. A higher rating indicates a higher difficulty for tasks and a higher skill level for users. Elo rating is most widely known from the board game chess. Each player in chess has a rating that reflects their current skill level. After a completed game, the rating of the winner increases and the rating of the loser decreases. The change in rating is dependant on the difference in rating between the players. If a highly skilled player wins a game against a low skilled player, the ratings hardly change. If the low skilled player wins, their ratings will change significantly.

In the proposed software solution of this thesis, both students and tasks are considered to be players with a numerical skill rating. If a Student answers a task correctly, their rating will increase and the rating of the task will decrease. This allows the system to estimate the probability of a student answering a task correctly based on the ratings of the student and the task. The system will then select appropriate tasks for the user based on this probability. The Elo rating algorithm is fully automated and the accuracy of the student's Elo rating will increase for each task answered.

The second part of the proposed solution is the collaborative filtering algorithm. This algorithm will be used to predict whether a student will answer a task correctly, based on whether or not similar students answered correctly. The similarity between two students is based on how many tasks they answered similarly. The purpose of collaborative filtering in this thesis is to select tasks that best allow the student to display their knowledge.

Collaborative filtering will be used to present the students with the tasks they are most likely to solve. Note that collaborative filtering happens after the Elo algorithm has selected a set of tasks with a fitting overall difficulty. If this was not done, collaborative filtering would recommend the easiest tasks in the set every time. The way we will use collaborative filtering is by:

1. Calculating similarity between the target student and the other students.
2. Choosing the students that have performed most similarly to the target student.
3. Presenting the target student with the task most similar students have answered correctly.

The system should then be able to automatically find students that have the most correct answers within textual tasks, image tasks etc. Students that have a performance bias towards a certain representation may be presented tasks of this type.

Both the Elo rating algorithm and the collaborative filtering are used to estimate the chance of the learner answering a task correctly. However, although they try to estimate the same probability, the basis for these calculations is different, and both systems are needed for the proposed solution to function as desired.

Without the Elo rating algorithm, collaborative filtering would converge towards the easiest tasks in the database, as these tasks have the highest chance of being

correctly answered by anyone. The collaborative filtering algorithm will try to recommend tasks that are suited for the user, based on results from similar students, besides only considering the difficulty.

When the Elo rating algorithm is combined with the collaborative filtering algorithm, the process of selecting new tasks for the user will look like this:

1. Use the Elo rating algorithm to find tasks of appropriate difficulty levels.
2. Use collaborative filtering to choose which one of these tasks the student should be presented.

1.3 Structure of this thesis

Chapter 2 will describe the most important background theory in the field of adaptive e-learning. This includes test theories, Elo rating systems, and recommender systems in detail. In Chapter 3 we present the state of the art. This chapter contains relevant research and some existing systems are described in more detail. The architecture and implementation of the solution we propose in this thesis are described in detail in Chapter 4. Implementation of collaborative filtering and the Elo rating system are also explained in depth. The experiments and their corresponding results are presented and discussed in Chapter 5. Lastly, Chapter 6 contains discussion and conclusion to the problems raised in this thesis, as well as future work.

Chapter 2

Background theory

In the book *Online education: learning management systems: global e-learning in a Scandinavian perspective* [3] Paulsen describes e-learning as interactive learning in which the learning content is available online and provides automatic feedback to the students. Online communication with real people may or may not be included, but the focus of e-learning is usually more on the learning content than on communication between learners and tutors.

The term e-learning is also often used in a broader perspective than just online education. Kaplan-Leiserson has created an online glossary for e-learning terminology [4] which provides this definition of e-learning:

> *E-learning covers a wide set of applications and processes, such as Web-based learning, computer-based learning, virtual classrooms, and digital collaboration. It includes the delivery of content via Internet, intranet/extranet (LAN/WAN), audio and videotape, satellite broadcast, interactive TV, and CD-ROM.*

Today the term e-learning often refers to learning material provided through the World Wide Web. Often this experience is interactive and may be personalized in some way. The term e-learning may also be used to describe content management systems, which do not provide the training experience but rather to organize,

communicate and distribute learning objects. Two well know examples of this are It's Learning[1] and Blackboard[2].

2.1 Adaptation in e-learning

The main purpose of computerized adaptive e-learning is to perform iterative and adaptive administration of tasks given to users in order to aid a dynamic learning process. The systems customize the learner's experience based on the individual learner's performance and/or preferences. The degree of adaptiveness in these systems vary from a simple approach of predefining different curriculum based on difficulty and letting the user choose difficulty, to autonomous systems that continuously acquire a better understanding of the user's skill, needs, and preferences over time. Adaptive e-learning systems may also change the item sequence at any point in time for any user if the difficulty is not a good fit for the learner.

Adaptivity in an e-learning context can be divided into two different categories. The first category, *adaptive interaction*, describes adaptations that take place at the system's Graphical User Interface (GUI) and are intended to facilitate or support the users interaction with the system. This does not affect the learning content itself, only how it is presented. Examples of adaptive interactions can include font sizes, color schemes or using different graphical modules. This is often done to accommodate user preferences or user disabilities.

The second category, *adaptive course delivery*, focuses on the fit between course contents and user characteristics or requirements, aiming for the "optimal" learning result. This will only affect the learning content, and not how it is presented to the user. This technique will also keep time and effort economy to a minimum. This adaptation technique is the most common and is widely used among adaptive e-learning systems today.

[1] https://www.itslearning.com/
[2] https://www.blackboard.com/index.html

In this thesis, we define adaptation as a system's ability to change depending on the performance or preference of the users. We define personalization as the adaptation from the users' point of view.

Over the last decades there have been four major paths to adaptation in e-learning [5]:

Learning style identification

H. Pashler et. al. describes learning style as the concept that individuals differ in regard to what mode of instruction or study is most effective for them [6].

Many adaptive e-learning systems are mainly based on identifying the students learning style prior to the start of the course. This is often done by using a survey or questionnaire, where the students give answers about their own behavior in different settings will classify the students preferred learning style.

Felder-Silvermans model from 1988 [7] differentiates learning styles through four dimensions: perception (Sensory/Intuitive), information input (Image/Verbal), information process (Active/Reflective) and understanding (Sequential/Global). A survey conducted by H. M. Truong in 2016 [8] shows that the FelderSilvermans model was by far the most widely used theory in adaptive learning systems based on learning style identification. It was used by 70.6% out of the 51 papers surveyed. Truong also points out some major drawbacks to using learning styles adaptation. Firstly, the results may be biased, as the results of the survey depend on the students' judgment and how well they are able to evaluate their own behavior in different environments and settings. Secondly, the learning style is often measured only at a single point in time. The preferred learning style of a student may change over time, and also during a single course or subject. These surveys can also be time-consuming, demotivating and tiring for the students.

In later years, the concept of learning style identification has been highly criticized. This is regarding the lack of scientific evidence, fundamental problems regarding

measuring learning styles, and that the significant empirical evidence is almost non-existent [9].

Learning styles are therefore considered as old and outdated today, and we have decided to utilize other techniques for adaptation in e-learning.

Recommender systems

The core functionality of a recommender system in an e-learning environment is to provide recommendations of appropriate educational material for the user. These recommendations are based on explicit or implicit feedback from the user or users that are similar to them. Over time the system will have better and stronger opinions about the users' preferences (as more data is available) and will have the ability to make better recommendations.

The most used techniques in recommender systems are *collaborative filtering*, *content-based filtering* or a combination of those. This makes it possible to take advantage of the strengths and minimize the weakness of each approach [10]. These are described in more detail in Section 2.4.

Link adaptation

Link adaptation is a class of hypermedia adaptation, which is a technique of adjustment (presentation, highlighting or concealment) of hyperlinks with the aim of selecting the appropriate content for the user. Link adaptation can also be described as Adaptive navigation support, and its goal is to guide the student through the curriculum with an optimal path by adaptively sorting, annotating, partly hide or disabling links or a direct guidance (e.g. a "continue" button). This will make the users choice easier in how to proceed [11].

Link adaptation is more suitable for a computerized adaptive practice system, than a test system like ours. We have therefore decided not to utilize this strategy.

Personalized software agents

In the book Artificial intelligence: A modern approach Russel and Norvig describe a software agent as anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [12]. More specifically, intelligent agents are defined as agents that have some degree of intelligence.

The growth of artificial intelligence over the last years has led to a trend of implementing intelligent software agents in adaptive e-learning systems. The use of intelligent agents in an e-learning environment can be categorized into two groups: harvesters and pedagogical agents. Harvester agents' main objective is to collect learning material. This is often done from heterogeneous repositories and its success depends on the quality and standard of teaching material representation. The main goal of pedagogical agents is to motivate and guide the users through the curriculum, by asking questions and proposing solutions to the user [13].

2.1.1 Optimal difficulty

Presenting a leaner with tasks of optimal difficulty is crucial for keeping the learner motivated. This is applicable to both learning scenarios and ability assessment scenarios because the motivation of the user may affect the ability to solve tasks. Csikszentmihalyis flow theory states that a students' intrinsic motivation and personal experience are dependent on the balance between the difficulty of the task and the skill of the user [14]. An adaptive e-learning system should aim towards placing every student in their personal flow zone. The optimal flow varies between students and means that presenting the same task to an entire class is not optimal. Adaptive e-learning systems should have the ability to find each student's flow zone [15].

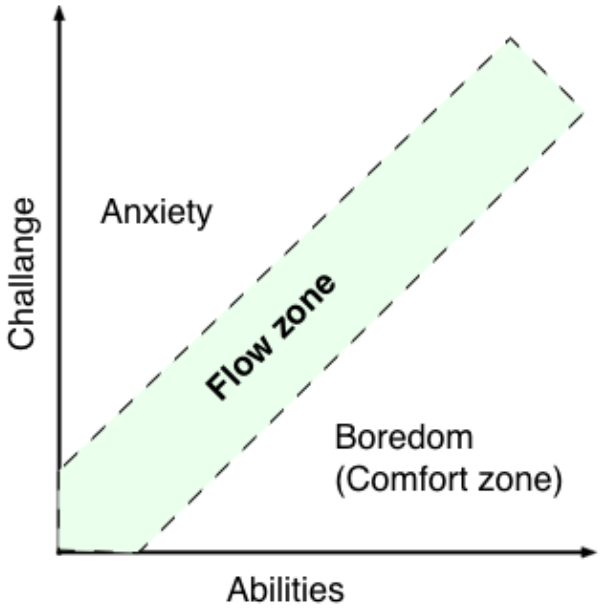

FIGURE 2.1: Flow zone chart

Previous research suggests that the optimal difficulty may differ depending on the type of subject to be learned and motivation (internal, external), particularly that in school-related activities students prefer lower levels of challenge [16]. Papoušek, J. and Pelánek, R. suggest a success rate target is around 65% [17]. They also point out that there is a difference between the optimal target success rate for activities in school, compared to out-of-school activities. Klinkenberg, S. et al, however, suggest in their study somewhere between 60% and 80% as the best target success rate [18]. Eggen and Verschoor state in their paper from 2006 that a target success rate below 50% will demotivate students [19]. They suggest raising this to 70%.

For this proof of concept we decided to go with the middle ground and use the target success rate of 65% as Papoušek, J. and Pelánek, R. suggests [17].

2.2 Test theories

A test can be studied from different angles. The estimate of a user's ability and the task difficulty can be evaluated according to different theories. This section will describe two; Classical Test Theory (CTT) and Item Response Theory (IRT).

In 1904, Charles Spearman published his article *The proof and measurement of two things* [20], describing how to find errors in measurement of item difficulty or user ability and how to obtain an index of reliability for making corrections. This is considered to be the beginning of Classical Test Theory. Item Response Theory was first described by Lord and Novick in their paper *Statistical theories of mental test scores* from 1968 [21], and is considered to be a more modern and complex approach.

CTT was originally the leading framework for analyzing and developing standardized tests. Since the beginning of the 1970s IRT has to some degree replaced the role of CTT, and is now the major theoretical framework used in the field of testing and statistical psychometrics [22].

2.2.1 Classical Test Theory

The most important concept in classical test theory is the reliability of the observed test scores. *The Glossary of Important Assessment and Measurement Terms*, by The National Council on Measurement in Education (NCME) describes classical test theory as a theory of testing, based on the idea that a persons observed or obtained score on a test is the sum of a true score (error-free score) and an error score [23].

$$\text{Observed score} = \text{True score} + \text{Error} \tag{2.1}$$

The reliability, X is denoted ρ^2_{XT}, and is calculated based on the ratio between observed score variance and the true score variance:

$$\rho^2_{XT} = \frac{\sigma^2_T}{\sigma^2_X} \tag{2.2}$$

According to the classical test theory the reliability cannot be calculated, hence the true score is not available. But CTT has other shortcomings as well. The

most important is the inability to separate the test characteristics and user characteristics. E.g the item difficulty could only be interpreted in the context of the users and vice versa. There are also other drawbacks to CTT. It lacks support for adaptive testing, it does not account for guessing, and it is built for the average student. The results of this are that the students who deviate from the average will receive less accurate skill estimations. On the upside, classical test theory is easy to calculate and to understand for humans in comparison to IRT, which has relatively complex estimation procedures.

Item response theory is the more modern approach and was developed to overcome some of the drawbacks with CTT. Today IRT is preferred in most cases and is described in more detail in Section 2.2.2.

2.2.2 Item Response Theory

Item Response Theory has roots in Psychometrics and is concerned with accurate test scoring and development of test items. IRT consists of statistical models that relate item responses to the latent abilities that the items measure [21].

IRT is widely used in education today. It is often used to calibrate and evaluate items in a test, and to score students abilities, attitudes, or other latent traits. Xinming An and Yiu-Fai Yung state that over the last decades, educational assessment has used more and more IRT-based techniques to develop tests [24]. Major educational tests like the Scholastic Aptitude Test (SAT)[3] and Graduate Record Examination (GRE)[4] are developed by using Item Response Theory. This is due to accuracy and reliability while providing potentially significant reductions in assessment time and effort, especially via computerized adaptive testing [24].

IRT can be divided into two categories: unidimensional and multidimensional. Unidimensional models a single ability dimension, θ (e.g user ability, item difficulty). The multidimensional models account for multiple traits. This greatly

[3]https://collegereadiness.collegeboard.org/sat
[4]https://www.ets.org/gre

increases the complexity, therefore most of IRT research and applications utilize a unidimensional model.

IRT models can also be classified based on the way responses are scored. Often multiple choice tasks are dichotomous, which means the tasks only can only be answered correctly or incorrectly. This is regardless of the number of multiple choice options. Another class of models applies polytomous responses. This means the outcome of the answer can have multiple score values. Examples of tasks of this type may be "rate this product on a scale from 1-10" or multiple choice tasks with more than exactly one correct answer.

The main building block of IRT is the item response function, which describes the probability of a given response as a function of a persons true standing on a latent trait or ability. Equation 2.3 shows the three-parameter logistic function. θ represents student's ability. The three item parameters a, b and c are representing discrimination, difficulty and guessing probability, respectively. D is a factor used for scaling. Students with higher ability have a higher probability of a correct response, but this probability cannot exceed 1.0.

$$P(\theta) = c + \frac{1-c}{1+e^{-Da(\theta-b)}} \tag{2.3}$$

By letting $c = 0$, $a = 1$ and $D = 1$ we obtain the one parameter logistic function (1PL, eq. 2.4) which is also known as the Rasch model. This is described in detail in Section 2.2.3.

$$P(\theta) = \frac{1}{1+e^{-(\theta-b)}} \tag{2.4}$$

The item response theory was designed to overcome some of the problems with CTT, and therefore features some advantages by implementing more sophisticated mathematical modeling:

- It makes stronger assumptions, and can, therefore, present stronger findings.

- Scaling of the item difficulty and the user's ability.
- User abilities and item difficulty can be meaningfully compared, using the same scale.
- IRT is more flexible and not test-dependent. The true score in CTT is defined in the context of a specific test.
- CTT tends to force a linear model on something that is nonlinear. IRT uses a more complex model to solve a complex problem.

2.2.3 The Rasch model

The Rasch model was developed by Georg Rasch [25] in 1960. In the Rasch model, the probability of a specific response to a task (correct or incorrect) is calculated using a logistic function of the difference between the person and item parameter.

The Rasch model is often considered to be the one-parameter logistic function in Item Response Theory model. Despite the fact that these models are identical, some argue that they differ in approach to conceptualizing the relationship between data and theory [26]. Item Response Theory emphasizes the primacy of the fit of a model to observed data over the requirements for fundamental measurement as in the Rasch model. The Rasch model also states that adequate data-model fit is an important requirement in order to claim a research instrument as valid for trait measurement [27].

$$P(X_{ni} = 1) = \frac{e^{(\beta_n - \delta)}}{1 + e^{(\beta_n - \delta)}} \tag{2.5}$$

β_n represents the users' ability, while δ represents the difficulty of the item. In an educational setting $P(X_{ni} = 1)$ is the probability that the student n will successfully solve task i.

For the combination method presented in this thesis, we will be using a modified version of the Rasch model. We will use this to calculate the probability of a user

solving a specific task correctly. This probability will affect the Elo rating for both the user and the task.

2.3 The Elo rating system

Elo rating is a system for calculating relative skill of players in competitive games. It is used in online multiplayer games like World of Warcraft and real-life games like chess. The main purpose of the system is to be able to rate and compare players using the same scale. Each player receives an initial numerical rating before their first match. This rating increases or decreases based on performance in subsequent matches. The difference in rating between the players is used to estimate the likelihood of each player winning the match. It scales so that if a highly skilled player wins a match against a low skilled player, the change in rating is minimal. If the high skilled player loses, the ratings of both players will receive a greater change. In the proposed solution, the Elo rating algorithm will be used to match users to tasks of appropriate difficulty. The Elo rating system also diminishes the impact of math problems with an inaccurate initial rating. The ratings of the tasks should converge toward the correct relative skill level over time. A precise estimate of the difficulty is still beneficial, as it reduces the time needed for the system to accurately estimate the difficulty. This also applies to users. A user will be classified quicker if the initial rating is close to the true skill of the user.

2.3.1 The Elo rating system for chess

The Elo rating system (ERS) was developed by Arpad Elo in 1960 for evaluating chess performances in United States Chess Federation. Elo himself describes the system as a numerical system in which differences in rating may be converted into scoring or winning probabilities [28].

The ERS consists of several different update formulas, designed for different type of matches. In this thesis, we will focus on current rating formula for continuous measurement.

$$R_n = R_0 + K(W - We) \tag{2.6}$$

R_n is the new rating after the event.

R_0 is the pre-event rating.

K is the rating point value of a single game score.

W is the actual game score, each win counting 1, each draw $\frac{1}{2}$.

W_e is the expected game score based on R_0.

Both players' ratings are updated after each match. Equation 2.6 shows that a player will gain rating points if his performance is above expected and likewise lose rating points if the performance is below expected. The consequences are therefore greater if the gap in rating between the players is large.

The expected outcome (winning probabilities), W_e is calculated by the Bradley Terry Luce (BTL) model in the original Elo rating system. This model is closely related to the Rasch model but is based on player versus player instead of player versus item. It is more suitable to use the Rasch model to calculate winning probabilities in an e-learning environment, as the scenarios are player vs item.

2.3.2 The Elo rating system in e-learning

Radek Pelanek states that the Elo rating system is suitable mainly for adaptive practice or low stakes testing in an educational application [29]. He also points out in his paper that the Elo rating system can easily be modified and applied in different domains. For example in educational systems with multiple choice tasks, where the students have a significant chance of answering correctly simply just by guessing.

Radek Pelanek's systematic survey shows that the Elo rating system is particularly attractive when the main focus is to build a reasonably behaving system quickly and cheaply. It provides a degree of adaptive behavior without expensive expert input. This achieved because the system can estimate the difficulty of items (questions, problems) and skills of users from the results of previously completed tasks. His system also shows that a system like this needs at least 100 students to get good estimates of item difficulty.

For every user i there is a rating score θ_i and θ_j for every task j. Both of theese ratings are updated after every task is completed, utilizing equations 2.7 and 2.8 respectivly for the user and the task. For every task j and user i, the probability of the outcome $R_{ij} \in \{0, 1\}$, is calculated ($R_{ij} = 1$ for success and $R_{ij} = 0$ for failure). The expected probability that player i will manage to solve the task is calculated by a logistic function (equation 2.9 and 2.10), known from the Rasch model.

$$\theta_i = \theta_i + K(R_{ij} - P(R_{ij} = 1)) \tag{2.7}$$

$$\theta_j = \theta_j + K(R_{ji} - P(R_{ji} = 1)) \tag{2.8}$$

$$P(R_{ij=1}) = \frac{1}{(1 + e^{\frac{(\theta_i - \theta_j)}{400}})} \tag{2.9}$$

$$P(R_{ji=1}) = \frac{1}{(1 + e^{\frac{(\theta_i - \theta_j)}{400}})} \tag{2.10}$$

If the user i manages to solve the task j, the result for the user would be $R_{ij} = 1$, and $R_{ji} = 0$ for the task. Likewise, if the user fails to solve the task, the outcome of functions 2.9 and 2.10 will be $R_{ij} = 0$ and $R_{ji} = 1$.

The K value in equations 2.7 and 2.8 are used to specify the sensitivity of the model. The K value also represents how many rating points a user or a task can possibly gain or lose. Often K is set to a constant value, but could also represent uncertainty. This by making it a function of time or items encountered

for example. A high K value will make a great impact on the Elo rating, and a small K value will make less impact.

2.4 Recommender systems

Recommendation systems are systems designed to use explicit or implicit feedback from users to recommend items the user likes, wants or needs [2]. This is in contrast to systems where the user has to explicitly search or browse to find the items they are looking for. Recommender systems are used in a variety of domains. This includes online shopping, dating services, online newspapers and video streaming services. The core feature of a recommender engine is to estimate how good of a match an item is for a given user. The item may be a movie, a news story, or in this case, a math problem.

2.4.1 Content based filtering

Content-based filtering is the use of metadata to recommend items to users. Items need to have metadata defined and users need to specify properties, or preferences, about themselves. The system recommends items to users based on the preferences specified by the users and the properties of the items. Although this method of making recommendations is widely used in media and shopping domains, content-based filtering does not fit well within the e-learning domain [30]. This is because, in an educational setting, The category of problems is either chosen by the teacher or the student directly. Content-based filtering also relies heavily on explicit information from the user, which will reduce the time available for math problem-solving.

2.4.2 Collaborative filtering

Collaborative filtering exploits the principle that users who have had similar preferences in the past, most likely will have similar preferences in the future. This assumption is used to estimate the preference of a given item for a given user. Note that in this thesis, the chance of a successful answer is estimated instead of a preference. Collaborative filtering is a part of most recommender systems and is applicable in a wide variety of domains. The collaborative filtering algorithm generates a space with as many dimensions as there are items in the database. A vector is generated for each user, where the value for each dimension is the rating of the item. The system then calculates the similarity between every user. There are multiple approaches to calculate similarities, such as Pearson correlation and vector cosine. When the system predicts how a user will rate an unseen item, the ratings given by similar users are weighted by their similarity and averaged.

Table 2.1 shows the ratings of four users where user A has completed items 1-3 and users B, C, and D have completed items 1-4. The ratings are either 1 (like) or -1 (dislike). The similarity between user A and the other users are based on the items that both A and the other users have completed (items 1-3). The rating of user A on item 4 will be predicted by averaging the other users' ratings of item 4 multiplied by their similarity score. In this case, that would be

User	Item1	Item2	Item3	Item4	Similarity to user A
A	-1	-1	1	?	1
B	1	1	-1	-1	-1
C	-1	-1	1	1	1
D	-1	-1	-1	1	0.33

TABLE 2.1: Collaborative filtering example

$$\frac{(-1 \times -1) + (1 \times 1) + (1 \times 0.33)}{3} = 0.78 \tag{2.11}$$

This is the predicted value of whether or not user A will answer correctly on item 4.

k-Nearest Neighbors

Collaborative filtering is one of the most successful recommender techniques, but there are drawbacks to using it. The biggest one being the need for computing resources. Making predictions based on the entire data set will normally yield the most accurate results, but it also requires a large amount of computing resources [31]. Most implementations of collaborative filtering use a neighborhood-based algorithm [31]. These algorithms only look at the most relevant users when making predictions. Users that are significantly different from the target user will generally provide little information when making predictions and can be removed to reduce the need for computational resources.

k-Nearest Neighbors (k-NN) is the most widely used neighborhood algorithm. k-NN calculates the similarity between users in the same way as the version of collaborative filtering described in chapter 4.3.2. The difference is that k-NN only looks at a subset of users. The number of users to look at is either given explicitly or as a minimum requirement.

The algorithm places users in an n-dimensional space, where each dimension is one item the user has rated. In this thesis, items refer to math tasks and user rating refers to whether or not the user answered the task correctly.

Figure 2.2 is an example of a prediction scenario where k-nearest neighbors is used. The goal of the example is to predict the chance that users A and B will answer Task 3 (T3) correctly given their answers to T1 and T2. There are 13 other users that have completed all three tasks.

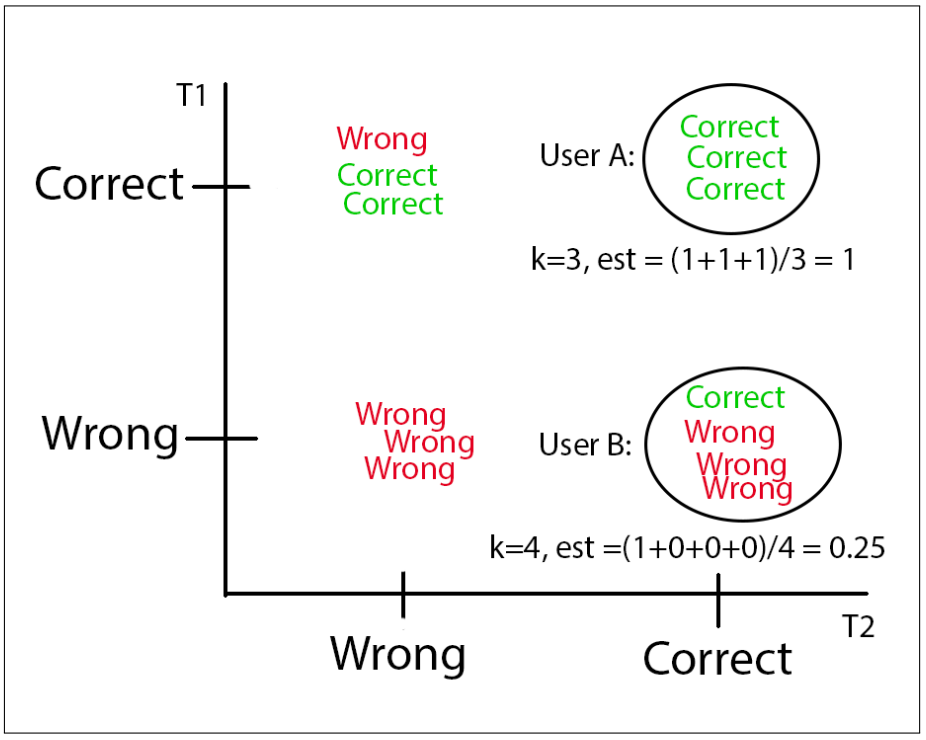

FIGURE 2.2: k-NN basic example

Let us look at user A. User A has answered both T1 and T2 correctly. There are three other users that have answered both T1 and T2 correctly. In this case, the number of most relevant users is three. When calculating the prediction for User A, you average the results from the neighbors. In this case, the algorithm predicts a 100% chance of success.

Now let us predict the chance of user B answering T3 correctly. User B has answered T1 falsely and T2 correctly. Four other users have also answered T1 false and T2 correct. Out of the four users, three answered T3 falsely, and one answered correctly. The algorithm will then predict that user B has 25% chance of answering T3 correctly.

If the system uses a minimum number of neighbors, and this number is not met, the system will either be unable to predict a score, or the system uses less similar neighbors. Neighbors who are not completely similar may still provide some data. The data from less ideal users are weighted by their similarity score.

k-NN will be used to implement collaborative filtering in this thesis. This choice is based on the tradeoff between the accuracy of the algorithm and the computational resources needed. A simulation was performed to test the accuracy of k-NN and its alternatives within a relevant scenario. The simulation procedure, as well as the alternative algorithms, is described in detail in Section 4.3.2

2.4.3 Implicit vs. explicit feedback

The basis of all recommender systems is data. A movie recommender needs to know what movies each user likes so that other users can receive suggestions based on other people's preferences. A news story recommender needs data about what articles each user have read, and a music recommender needs data about how frequently each song is listened to by each user.

Each recommender system designer needs to choose whether to use implicit feedback, explicit feedback, or a combination of both to gather data. Explicit feedback may contain more accurate data, but it requires users to actively rate the item, which is intrusive. Implicit feedback may contain less usable data, but require no additional work for the user. If a combined feedback mechanism is possible, it is usually the best approach. A combined system relies on both data gathered automatically and on data given by users. Netflix is an example of a service that uses a combined feedback system. Netflix has recently changed the explicit feedback system from a 5-star rating system to a binary thumbs-up-or-down system. This change resulted in more data, as more users took the time to give feedback when the choice was simpler [32].

Service	Feeback Style	Explicit Feedback	Implicit Feedback
Netflix	Combined	Thumbs up-or-down	Content watched
Spotify	Implicit	N/A	Songs played
Amazon	Combined	Rate products	Products bought and viewed

TABLE 2.2: Type of feedback in popular services

The system developed in this thesis will use the results of completed tasks as implicit feedback. This data will be the basis for calculating the similarity between students.

2.4.4 Cold start problem

The cold start problem within the domain of recommender systems refers to the challenge of making predictions to new users and suggesting new items. The cold start challenge can be divided into three types of problems: (1) recommendations for new users, (2) recommendations for new items, and (3) recommendations on new items for new users [33].

If a new user has made no ratings, the system will not have any data to base their recommendations on. A new item with no ratings may never be recommended, as recommendations are based on prior ratings.

One solution is to combine collaborative filtering with content-based recommendations. An example of this could be a movie recommender that queries the user about genre preferences. This can be used as a basis for recommendations until the system has enough data to include collaborative filtering recommendations.

The proposal of this thesis is to use the Elo ratings of users and items to recommend items until the RS has enough data to accurately predict outcomes. This should let the system recommend tasks based on the difference in relative skill until predictions can be made accurately.

Because the experiment we will carry out will consist of two phases, the first phase will be a completely cold start for the recommender system. The second phase will have new users, but the data from the previous users should lessen the impact of the cold start.

Chapter 3

State of the art

Adaptation has been studied thoroughly in the context of Computerized Adaptive Testing (CAT) with the use of Item Response Theory [34]. The main idea of CAT is to determine the ability level of a person dynamically. In CAT, item administration depends on the subjects previous responses. If the preceding item is answered correctly (incorrectly), a more (less) difficult item is presented. Hence, each person is presented with a test tailored to his or her ability. Computerized Adaptive Practice (CAP) systems, which focuses more on the learning and development of the learners' abilities. One example of such system is the Math Garden system, which focuses on practicing basic arithmetical operations. This system is described further in Section 3.2.

Adaptivity in an e-learning context has also been studied in the field of Intelligent Tutoring Systems (ITS). This type of system focuses more on learning complex cognitive skills (e.g. mathematics, physics). An ITS also often provides guidance to mastering a specific problem. This could hint to solving a specific problem, tailored feedback, or step-by-step solution monitoring.

Learner modeling is an important factor in most adaptive e-learning systems [35]. The learner model provides an estimate of the students' abilities and skill level at a certain state, based on the students' performance. The modeling is used to give feedback to the students, recommend specific parts of the curriculum or to tailor

the instructions to the students. One example of this is in the ALEKS[1] system. It tries to estimate the students' knowledge state at a certain point in time. This is in order to monitor the students' current skill level, progress, and highlight specific topics where the student is performing well and topics the student is struggling with. The student model could also be used to recommend relevant topics the student is ready to learn or not. According to a paper written by Pelnek, R. et al. in 2017 [36], the two most popular approaches to learner modeling is the Bayesian knowledge tracing [37] and models based on the logistic function. These models can be seen as extensions of the Rash model [25] from Item Response Theory. Learner modeling techniques most related to our work in this thesis are item response theory and the Rasch model, using a modified version of the logistic function.

3.1 Combining ERS and IRT for adaptive item sequencing

In 2013 Antal, M. presented a new item response model by combining item response theory and the Elo rating system [38]. This was developed as an alternative model for adaptive item sequencing, offering estimates for both learner's abilities and task difficulty. In her paper, she compared three different methods for difficulty estimation: IRT, ELO, and proportion correct, which is a simple approach where the estimates are given by dividing the number of incorrect answers by the number of total answers for the given item.

Two comparison tests were conducted, and consisted of 30 questions each (single choice, multiple choice, fill in), and were administered using Moodle[2]. A total of 137 students participated in the two experiments. The results show that all estimation methods produced difficulty estimates, which highly correlated with the other estimation methods.

[1] https://www.aleks.com/
[2] https://moodle.org/

The system developed shows that the combined Elo-IRT system obtains reliable examinee ability estimates after using a test of at least 30 tasks. This result supports the findings of van der Maas and Wagenmarkers, who concluded that 25 games were needed to obtain a reliable Elo rating for chess players [39]. Antal also points out that combining IRT with an Elo rating system has the benefit of providing both learner's ability estimates and item difficulty.

3.2 Use of ERS in the Math Garden system

The Math Garden system is a challenging web environment for children to practice arithmetic. It was used in a study by S. Klinkenberg et. al in 2010 [18]. In a period of ten months, 3648 children completed over 3.5 million arithmetic problems within the domains addition, subtraction, division, and multiplication. The Elo rating system was used for on-the-fly estimation of item difficulty and person ability parameters. The Rasch model was used for calculating the probability of a student to solve a specific task. In addition to measuring the student's success or failure of a task they also included response time in the estimation of the student's ability.

The system was built as an educational game, to motivate the students. They also used two reward systems to keep the students interested and motivated over time. First, performing well rewarded virtual coins and having the flowers grow in their virtual garden. The flowerbeds would, therefore, represent the Maths ability of the student. The coins awarded could be spent on rewards in an own section of the website (The Price Cabinet). Secondly, the user's flowerbeds would wither over time. The only way to undo this was completing a session of 15 tasks.

Equation 3.1 describes how the K value of the Elo system is calculated after solving a task with user j, and the task(item) i. The same calculation was used for both user and task. The K value can be interpreted as the learning rate.

$$\begin{aligned} K_j &= K(1 + K_+U_j - K_-U_i) \\ K_i &= K(1 + K_+U_i - K_-U_j) \end{aligned} \tag{3.1}$$

$K = 0.0075$ is the default value, when there is no uncertainty($U = 0$). $K_+ = 4$ and $K_- = 0.5$ are weights for rating the uncertainty for the user and the task.

$$\hat{U} = U - \frac{1}{40} + \frac{1}{30}D \tag{3.2}$$

The uncertainty equation (eq. 3.2) depends on recency and frequency. This equation is used for calculating uncertainty for the user and the task, with a provisional uncertainty of $U = 1$ and $0 \leq U \leq 1$. After completing 40 tasks, the uncertainty will reach the minimum of 0. After D number of days of not playing the uncertainty will increase, and after 30 days without playing it will reach the maximum of 1.

The system also uses a scoring rule to incorporate speed. This is known as the high-speed high-stakes (HSHS) scoring rule. This is a trade-off for the user between speed and accuracy, where quick responses will grant higher scores. The score in case of a correct answer will be equal to the remaining time. In case of an incorrect answer, the score will equal the remaining time multiplied by -1.

The tasks are selected based on the mean probability of answering correctly is $\sim .75$. The probability is restricted to be $> .5$. Repetition of tasks is restricted to only be reused after 20 tasks.

The results concerning the validity and reliability presented are promising. It showed an increase in player ability rating across all grades tested on (from kindergarden through secondary education), except grade 5 and 6, where the ratings did not differ. The results also show high correlations to the reference scores used (CITO scores). Concerning the item bank, they noticed that the item difficulty ratings converged in about 8 weeks, and stayed consistent across time. No indication of learning effects was found as some of the items were reused.

The Math Garden system has many similarities to our system. However, they differ in some major concepts. First of all, the Math garden system's concept is practicing, where our focus is more towards testing. The Math Garden system is

only considering the tasks' and user's Elo rating to select new tasks for the users. For calculating the uncertainty (k value) it uses the number of days since last playing as a factor, in addition to the number of tasks completed. When selecting tasks they have used a target success rate of .75 with a lower limit of .5. This is slightly higher than in our system, where we use .65, with lower and upper limits of $-25/+75$ Elo rating points.

The system is also built more like an educational game, with more focus towards keeping students motivated. The use of HSHS adds a competitive dimension to the system.

3.3 Other adaptive e-learning systems

There exists a multitude of Learning Management System (LMS) environments that have implemented adaptive learning algorithms. Examples of these are Fishtree, Smart Sparrow, and Knewton among others. However, these systems are commercial closed-source products that do not share details about the underlying technologies and algorithms used, therefore they cannot be evaluated in detail nor compared.

FishTree[3]

"Fishtree was created to solve the challenges of scaling personalized learning. Fishtree automates resource generation, automatically aligns content to learning objectives or competencies and adapts to the learning profile and knowledge gaps of every learner with one click".

FishTree is a commercial for-profit service and does not go into detail about the algorithms they use. The patent owned by FishTree Inc. shows that the adaptation of FishTree mainly stems from the identification of learning styles [40].

[3]https://www.fishtree.com/

Smart Sparrow[4]

Smart Sparrow is an adaptive learning platform incubated at the School of Computer Science and Engineering at the University of New South Wales in Sydney, Australia in 2011. It provides adaptive learning through different pathways through a course, which can be different for each student. Different modules in the courses can also be customized, but this needs to be done by the instructor. Course creation can also include interactive multimedia such as pictures, videos, and sliders. The system also provides information about the students and the parts of the curriculum where they are suffering and which parts the students find easy. The instructor can also distribute learning content online, so teachers can access it from anywhere. Smart Sparrow is a global leader in adaptive and personalized learning technology and is used by over 500 institutions.

Knewton[5]

Knewton is a web-based adaptive learning platform founded in 2008 and is concentrated in the fields of science, technology, engineering, and mathematics in both lower and higher education. The Knewton system keeps track of the student's profile, and log how they overcame problems. This information is reused to help other similar students in similar situations. The system also gives parents the opportunity to follow their child's progression, to learn what they are struggling with and to understand specific concepts. The content is presented based on the preferences of the user. This can be video, text, games or provide short or long and detailed explanations. The difficulty of the practice questions can also be adjusted. Knewton also partners with several companies to provide learning material.

[4]https://www.smartsparrow.com/
[5]https://www.knewton.com/

3.4 The use of Recommender Systems

Recommender systems are currently being used in a wide array of domains. They are used by Netflix to recommend movies to users [41]. Recommender system have been used by Amazon to recommend products to buyers for over two decades [30]. These systems are also reportedly used to match users in dating services and to find restaurants when on vacation.

Content-based filtering uses metadata to recommend items. An example of this is that Netflix may ask you if you like action movies, and if you do, it will rank action movies higher than other recommendations. These kinds of recommendations work well in domains where the user know what they like or want [41]. However, in the domain of e-learning, this does not typically apply. Preferences of students are relevant on the item-level, i.e. representations of concepts, but the overall curriculum is usually predetermined. There have been attempts at generating content-based profiles for students by using their browsing history to classify their preferences of learner material [42]. But this did not yield good results and is quite intruding to the privacy of the students.

The ordering of tasks in e-learning is referred to as item sequencing. Item sequencing has successfully been performed by using collaborative filtering [43]. This was done by creating similarities between students based on previously given answers, and by using k-Nearest Neighbors to predict the chance that a student will be able to successfully answer a given question.

Chapter 4

Architecture and implementation

An implementation of the proposed solution was needed to conduct the experiment as planned. Some of the design choices were forced, as the system was already fully developed, and rewriting parts of the system is time consuming and out of our scope. The implementation of the proposed solution in Matistikk can be viewed as a proof of concept and is meant for scientific purposes only. This chapter will present the chosen architecture, the reasoning behind design choices, and the implementation.

4.1 Matistikk

The proposed solution was implemented as an extension to Matistikk. Matistikk is an e-learning system that focuses on organizing, presenting and correcting mathematical problems for students of all ages. Matistikk lets teachers create math tasks, organize the tasks into tests, and publish the tests to specific groups of students, schools, or classes. Matistikk was developed as a bachelor thesis at NTNU Trondheim, during the spring of 2017 [44]. The proposed solution is based on the underlying software of Matistikk, but it functions as a standalone feature. A brief overview of how Matistikk functions will be presented to illustrate how the proposed solution works.

The index page of Matistikk is a log-in page (fig. 4.1). Based on your account privileges, you will either be presented with available tests if you are a student, or administrative functionality if you are a teacher or administrator, such as creation or management of learning content.

FIGURE 4.1: Matistikk: Login view

In this thesis, the administrative features are only used to monitor statistics and to create tasks and tests. Two new buttons have been added to this page. "Tilbakemelding" will redirect the user to a feedback form used in the experiment. "Testområde" will initiate the test by automatically generating a new user account and logging the user in. This process is not visible to the user. The user is presented with some basic instructions (fig. 4.2). When the user clicks the "start test" button, the first task is displayed. Examples of tasks are shown in Figure 4.3.

FIGURE 4.2: Matistikk: Start page view

After the first task is answered, the system decides which task should be presented next, then the user is presented with the new task. Note there is no explicit feedback to the user if the last answer was correct or not. The only indication is the displayed Elo rating of the user. This continues until the system has no more fitting tasks, or if the student wishes to end the session. The session may be ended by clicking the "Logg ut" button. For the purpose of this thesis, every task is in the form of a multiple choice question. This was chosen because Mattisikk already had built-in features for auto-correcting multiple choice questions.

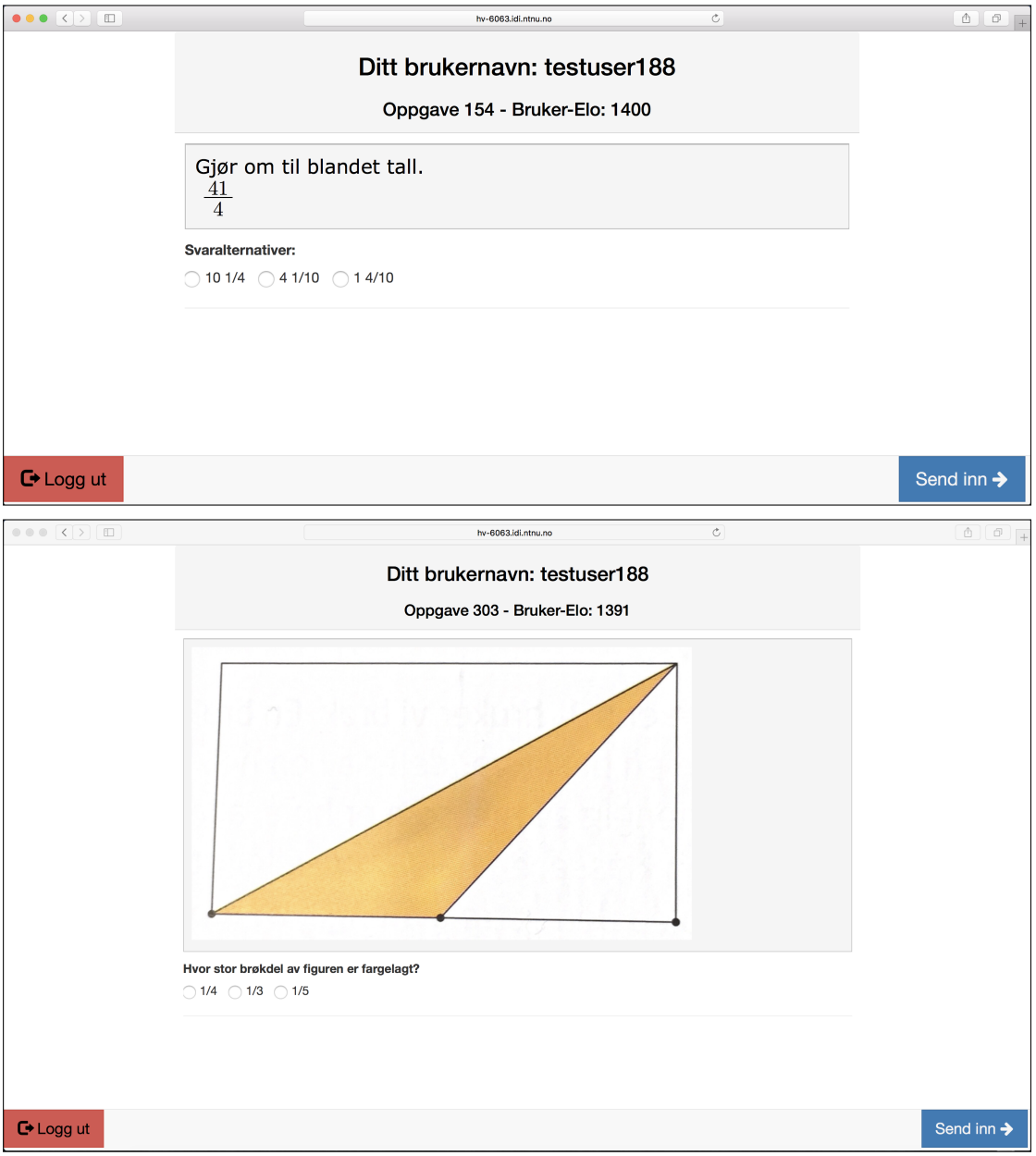

FIGURE 4.3: Matistikk: Example task views

4.1.1 System architecture

Matistikk is a web application that uses the Django framework for python[1]. Django handles the core features of the web-server, as well as providing a direct administration interface used to manipulate the database entries. Django was used in combination with HTML5, JavaScript, CSS, and Bootstrap. Matistikk was

[1]https://www.djangoproject.com/

initially run on an SQLite database. This was replaced with a MySQL database ahead of the experiment because the SQLite database does not support concurrent handling of multiple users. The open source libraries Pandas and Surprise were used to structure the data and to implement the collaborative filtering algorithm. These libraries are described in detail in Section 3.

The system did not feature a convenient way of selecting tasks to be presented on the fly. The system required all tasks to be handled within tests. It was not possible to give a student a specific task, only tests (sets of tasks). There was no way to rearrange the tasks within a test while the test was ongoing. This was a problem, as the goal of the proposed solution was to select the most fitting task after each completed task. We addressed this problem by creating tests for every task in a one-to-one relationship. This means that all tests consisted of only one task. That way, a student would be given a complete test at a time, but the test would only contain one task. After the test was completed, a new test was selected. This had no practical impact on the users other than requiring more button presses to advance through the tasks. This also made the creation of the learning content more challenging and time-consuming, as each new task needed to be placed in a unique test.

When a new user is logged in, the user is assigned to either the test group or the control group. This is done by assigning even user ids to the test group and odd user ids to the control group (listing 1). This happens behind the scenes and the user is not shown which group they are assigned to.

```
1     if Person.objects.count() % 2 == 0:
2         user.inControlGroup = False
3     else:
4         user.inControlGroup = True
5     user.save()
```

LISTING 1: User assignment to test- or control group

The attribute *inControlGroup* is stored in the Person model in the database. This is used in the selection of the next task. The implementation of the Person model can be studied in appendix B.

All new users were initiated with the same Elo rating. For test phase one this was set to 1300. For test phase two this was adjusted to 1400. The initial Elo ratings of the tests were set in advance by the authors. The initial task ratings were based on estimates from the authors and textbook guidelines, and is further described in Section 4.4. The Elo rating score was stored in the Person and Test models respectively, in the attribute *elo_rating* (Appendix B).

4.2 Task selection

The next task presented to the user is selected in the functions *get_next_task(user_id)* for users in the test group and *get_next_task_for_control_group_user(user)* for users in the control group. These functions are located in *Matistikk/administration/templatetags/administration_extras.py*. The algorithm used to select the next task is dependent on which group the user is in; either the test group or the control group. The test group algorithm use both Elo rating and collaborative filtering, while the control group only use Elo rating. The task selection for the test group and the control group are described in detail in Section 4.2.1 and Section 4.2.2, respectively.

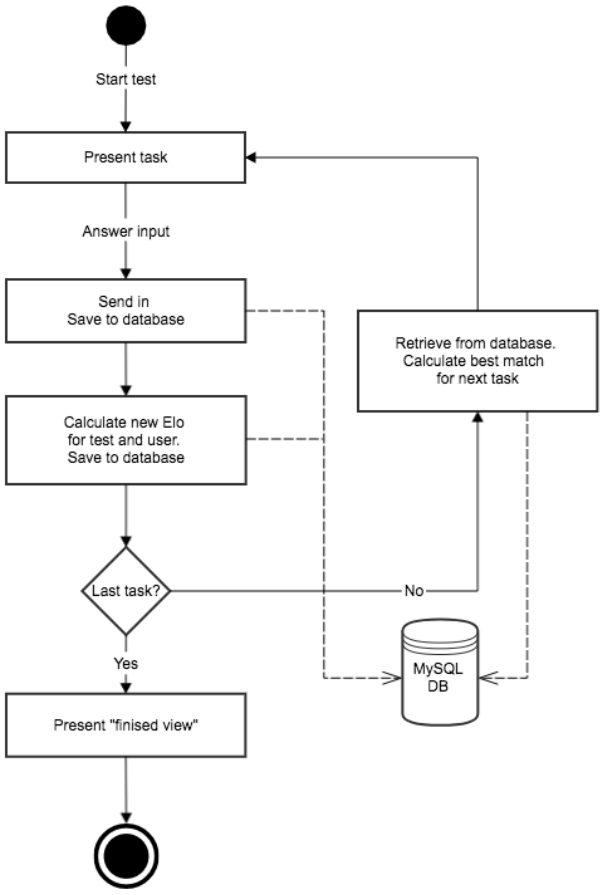

FIGURE 4.4: Task selection in Matistikk - flowchart

As Figure 4.4 shows, When a user starts a new session, they are instantly presented a task. When the user submits an answer, this is sent to the database and new Elo ratings for both the user and the task is calculated and stored in the database. If there are no more relevant tasks for the user, the test is terminated and the user will be logged out. The user is then presented with the "finished view" as depicted in Figure 4.5.

FIGURE 4.5: Matistikk: Finished view

If there are more relevant tasks available for the user, the next task will be selected and presented to the user. This will repeat until there are no more relevant tasks available for the user.

4.2.1 Task selection for test group

For the test group, the task selection is dependent on the user's Elo rating and the recommendations from the collaborative filtering algorithm. This process is described by the Algorithm 1.

Algorithm 1: Select next task for test group

Input: user_id

Output: selected_task

```
1  user = get user from database with current user_id
2  if user is in control_group then
3  |  return get_next_task_for_control_group_user(user)
4  end
5  elo_candidates = get all uncompleted tasks for user which is in the range -25/+75
6  if length of elo_candidates == 0 then
7  |  no tasks available, redirect to home
8  end
9  collaborative_filtering_estimates = get list of tasks sorted on best
     recommendations
10 write meta data to log file
11 return collaborative_filtering_estimates[0][0]
```

This algorithm is called for all users, regardless of whether they are in the control group or the test group. If the user is in the control group, they will be redirected to the appropriate algorithm, Algorithm 2, in step 2 and 3.

If the user is in the test group, the algorithm will retrieve all uncompleted tests with a maximum Elo difference of -25/+75 Elo rating points compared to the user. If there are no tasks available within the Elo rating interval, -1 is returned. The session will then be terminated. If the list is not empty, it will be passed on to *get_collaborative_filtering_estimates*.

The collaborative filtering algorithm receives the list of tasks and returns the list sorted by the probability of success for the specific user. The first element of the list is the task that the user most likely will answer correctly, based on the predictions made by the collaborative filtering algorithm. The algorithm also returns data

about the accuracy of the prediction model and the number of neighbors used to predict the values. If there is a tie between the best tasks returned by the collaborative filtering algorithm, the task closest to the target Elo difficulty is selected. The ideal Elo rating is set to be one that that results in a 65% chance of success for the user. The selected task is then presented to the user.

4.2.2 Task selection for control group

The task selection for the control group is only dependent on the Elo rating of the user. Collaborative filtering is not applied to this group of users. The selection process is described in Algorithm 2.

Algorithm 2: Select next task for control group

Input: user

Output: selected_task

```
1 uncompleted_tests = get all uncompleted tasks for user which is in the range
   -25/+75
2 if length of uncompleted_tests == 0 then
3 |   no tasks available, redirect to home
4 end
5 uncompleted_tests_sorted = sort uncompleted_tests on closest to target success
   rate
6 return uncompleted_tests_sorted[0][0]
```

The function *get_next_task_for_control_group_user(user)* handles the task selection for the control group users. First, the relevant tasks (not completed by the user and are in the range -25/+75 Elo rating points from the user) are found. Then the tasks are sorted by how close they are to the target success rate, utilizing the modified version of the Rasch model (eq. 4.1). The first item is chosen and returned. If no relevant tasks are available the test will be terminated and the user will be presented with the "finished view" (fig. 4.5).

4.3 Implementation

4.3.1 Implementation of the Elo rating algorithm

The calculation of both the user's Elo rating θ_i and the task's Elo rating θ_j is computed directly after each task is completed. First, the expected outcome of the task is calculated. The equation 4.1 is a slightly modified version of the logistic function 2.9 from Section 2.3.2. These adjustments are implemented to better fit the rating scale used in this thesis and do not alter the underlying principles.

The following equation is used to estimate the probability of success for a user attempting a task. Let the user i encounter task j. The probability of success from the user's perspective will be:

$$P(R_{ij} = 1) = \frac{1}{1 + 10^{-\frac{\theta_i - \theta_j}{100}}} \tag{4.1}$$

This probability of success is then used to calculate the new Elo rating θ_i, based on the result of the task and the probability calculated in equation 4.1.

$$\theta_i = \theta_i + K(R_{ij} - P(R_{ij} = 1)) \tag{4.2}$$

The implementation of these functions can be studied further in Appendix A.

Example:

If a user i, with an Elo rating of 1330 encounters a task j with the Elo rating of 1345, the probability of success for the user $P(R_{ij} = 1)$, would be:

$$P(R_{ij} = 1) = \frac{1}{1 + 10^{-\frac{1330 - 1345}{100}}} \tag{4.3}$$

$$P(R_{ij} = 1) \approx 0.4145 \tag{4.4}$$

The probability of success for the task will equal the probability of failure for the user. This can simply be calculated like this:

$$P(R_{ji} = 1) = P(R_{ij} = 0) = 1 - P(R_{ij} = 1) \approx 0.5855 \tag{4.5}$$

The probabilities are then carried over to the update functions (eq. 4.6 and 4.7). If the user is able to solve the task, the result for the user is set to 1, and the result of the task is set to 0.

$$\theta_i = 1330 + 20(1 - 0.4145) \approx 1342 \tag{4.6}$$

$$\theta_j = 1345 + 20(0 - 0.5855) \approx 1333 \tag{4.7}$$

The new Elo rating for the user θ_i will be 1342, and 1333 for the task θ_j. The K value in this example is set to 20 for demonstration purposes and was chosen arbitrarily.

Listing 2 shows the implementation of *calculate_new_elo(user_id)*. The listing shows the process of calculating and storing the Elo ratings for both the user and the task. This function runs directly after a user completes a task.

```
1  def calculate_new_elo(user_id):
2
3      # Get user from DB
4      user = Person.objects.get(id=user_id)
5      user_elo = user.elo_rating
6
7      # Get last answer for user from DB
8      all_answers_for_user = Answer.objects.filter(user=user_id)
9      if not all_answers_for_user:
10         return json.dumps(True)
11
12     last_answer = all_answers_for_user[len(all_answers_for_user) - 1]
13
```

```
14      # Get task from DB
15      test = Test.objects.get(id=last_answer.test.id)
16
17      if len(all_answers_for_user) <= 0:
18          return json.dumps(False)
19      if last_answer.elo_has_been_calculated:
20          return json.dumps(True)
21
22      new_user_elo =
23              elo_calc.calculate_new_elo_for_user(
24                      calculate_K_user(user_id),
25                      user_elo,
26                      int(last_answer.correct),
27                      elo_calc.calculate_expected_success_for_user(user_elo,
28                                                                   test.elo_rating))
29
30      # Reversed last answer represents the result of the last task from the
31      # opponents(task) point of view
32      # 1 = user failed, 0 = user succeeded
33
34      reversed_last_answer = 1
35      if int(last_answer.correct) == 1:
36          reversed_last_answer = 0
37
38      new_test_elo =
39              elo_calc.calculate_new_elo_for_task(
40                      calculate_K_test(test),
41                      test.elo_rating,
42                      int(reversed_last_answer),
43                      elo_calc.calculate_expected_success_for_task(test.elo_rating,
44                                                                   user_elo))
45      test.elo_rating = new_test_elo
46      test.save()
47
48      if new_user_elo < 1200:
49          user.elo_rating = 1200
50      else:
51          user.elo_rating = new_user_elo
52
53      user.save()
54
55      user_logger(user, test, last_answer.correct)
56      test_logger(user, test, reversed_last_answer)
57      all_answers_for_user[len(all_answers_for_user) - 1].elo_has_been_calculated = True
58      all_answers_for_user[len(all_answers_for_user) - 1].save()
```

Listing 2: Implementation Elo task selection

First, the user and the last answered task is fetched from the database. Then the new user Elo is calculated and the result is reversed($1 = 0$ and $0 = 1$), to represent the result of the task. This result is used when calculating the new Elo for the task. The new user Elo rating is checked against a threshold of 1200, to prevent users from decreasing their rating below the available tasks. Finally, the user is saved to the database and some logging is done for monitoring purposes.

K value

Pelanek [29] and Glickman [45] suggest using an uncertainty function for the K value instead of using a constant. Because the ratings become more accurate when the user has answered more tasks, the rate of change in Elo rating should decrease as a function of the number of tasks completed. This allows for users of all skill levels to quickly be placed near their correct rating while avoiding spikes in the rating when the system has gained a high accuracy. In this thesis, we used a logistic uncertainty function (eq. 4.8), proposed by J. Ninan et. al in 2015 [46].

$$K = \frac{a}{1 + b \times n} \tag{4.8}$$

where $a = 125$, $b = 0.2$, $n =$ number of tasks completed

The variables a and b can be adjusted to fit the scale of the system. Figure 4.6 shows how the sensitivity of the Elo rating will decrease with the number of tasks completed by the user. We can see that after 15-20 tasks completed the certainty is stabilizing. This is approximately the number of consecutive tasks needed to solve to reach the highest rated task.

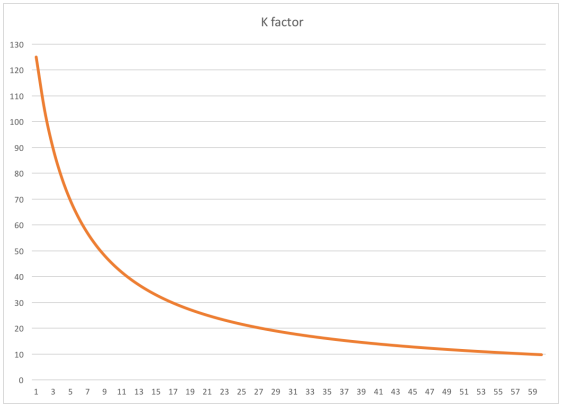

FIGURE 4.6: Logistic uncertainty function

The Elo calculations were done using a constant K value. This was done because the experiment that would be carried out would be held in multiple phases. The constant K value would lessen the impact of one phase having higher skilled users than the other. Another reason for not having a dynamic K value is to decrease the number of variables affecting the system. For future work, using a dynamic K value for both users and tests should be investigated.

4.3.2 Collaborative filtering design choices

Collaborative filtering can be implemented using a variety of algorithms. These algorithms have different efficiency and accuracy. In order to pick an implementation, a simulation of a prediction scenario was conducted.

This was done by creating fictional tasks, students, and answers to run a simulation on. To populate the database with data that represented a relevant scenario, each of the fictional students was labeled as either type A or type B. The tasks were also labeled as either type A or B. This was done to simulate that students may have a performance bias towards a certain task representation. Type A students have a performance bias towards tasks of type A. Type B students has a performance bias towards type B tasks.

Note that these two types were only simulated, and could represent any kind of performance bias e.g. textual task bias, numerical task bias etc. In the simulation,

this meant that a Student of type A had a higher chance of answering tasks of type A correctly.

The rule used to generate answers in the database was fairly simple. If a student had an equal or higher rating than the task, then an answer is generated and set to be 'correct'. If a student has a lower rating than the task, then the answer will be set to be 'wrong'. To represent the two types of students and their performance biases, students received a +50 Elo rating while solving a task of compatible type.

	Type A Student	Type B student
Type A task	+50	+0
Type B task	+0	+50

TABLE 4.1: Extra Elo received while solving tasks

					A	B	A	B	A	B	A	B	A	B	A	B	A	B
				Task ELO ->	1430	1430	1445	1445	1475	1475	1505	1505	1535	1535	1550	1550	1565	1565
User	Type	ELO	W/Benefit	User/Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	A	1440	1490	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
2	A	1450	1500	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0
3	A	1460	1510	3	1	1	1	1	1	0	1	0	0	0	0	0	0	0
4	A	1470	1520	4	1	1	1	1	1	0	1	0	0	0	0	0	0	0
5	A	1480	1530	5	1	1	1	1	1	1	1	0	0	0	0	0	0	0
6	A	1490	1540	6	1	1	1	1	1	1	1	0	1	0	0	0	0	0
7	A	1500	1550	7	1	1	1	1	1	1	1	0	1	0	1	0	0	0
8	A	1510	1560	8	1	1	1	1	1	1	1	1	1	0	1	0	0	0
9	A	1520	1570	9	1	1	1	1	1	1	1	1	1	0	1	0	1	0
10	A	1530	1580	10	1	1	1	1	1	1	1	1	1	0	1	0	1	0
11	B	1440	1490	11	1	1	0	1	0	1	0	0	0	0	0	0	0	0
12	B	1450	1500	12	1	1	1	1	0	1	0	0	0	0	0	0	0	0
13	B	1460	1510	13	1	1	1	1	0	1	0	1	0	0	0	0	0	0
14	B	1470	1520	14	1	1	1	1	0	1	0	1	0	0	0	0	0	0
15	B	1480	1530	15	1	1	1	1	1	1	0	1	0	0	0	0	0	0
16	B	1490	1540	16	1	1	1	1	1	1	0	1	0	1	0	0	0	0
17	B	1500	1550	17	1	1	1	1	1	1	0	1	0	1	0	1	0	0
18	B	1510	1560	18	1	1	1	1	1	1	1	1	0	1	0	1	0	0
19	B	1520	1570	19	1	1	1	1	1	1	1	1	0	1	0	1	0	1
20	B	1530	1580	20	1	1	1	1	1	1	1	1	0	1	0	1	0	1

FIGURE 4.7: Data used to simulate the collaborative filtering algorithms

Each algorithm was tested on the artificially generated data-set(Figure 4.7) and the data-set was reset after the test of each algorithm. After completing 20 tasks of each algorithm, the Root-mean-square error (RMSE) was recorded and used to evaluate the accuracy of the algorithms. The number of tasks was chosen to

reflect the amount of task the students should be able to complete the upcoming experiment.

Algorithm	**RMSE**	**Computational complexity**
Singular Value Decomposition (SVD)	0.09	$O((k^2) * n)$
Non-negative Matrix Factorization (NMF)	0.15	NP-hard
k-NN Basic	0.11	$O(nd + kn)$

TABLE 4.2: Evaluation of algorithms after 20 completed tasks

From Table 4.2 we can see that the NMF algorithm provides the least accuracy. Both SVD and k-NN Basic scores better in this simulation. We decided to use k-NN Basic since this has the lowest computational complexity of those two.

4.3.3 Implementation of collaborative filtering

```
1  def get_collaborative_filtering_estimates(user_id, tasks_to_consider):
2
3      answers = Answer.objects.all()
4      ratings_dict = {'itemID': [],
5                      'userID': [],
6                      'rating': []}
7      for answer in answers:
8          ratings_dict['itemID'].append(str(answer.test_id))
9          ratings_dict['userID'].append(str(answer.user_id))
10         ratings_dict['rating'].append(str(answer.correct))
11
12     df = pandas.DataFrame(ratings_dict)
13     reader = surprise.Reader(rating_scale=(0, 1))
14     data = surprise.Dataset.load_from_df(df[['userID', 'itemID', 'rating']],
15                                          reader)
16     train_set = data.build_full_trainset()
17
18     algo = surprise.KNNBasic()
19     algo.train(train_set)
20
21     estimates = 0
22     tests_with_predictions = []
23     predictions = []
24     for test_id in tasks_to_consider:
25         prediction = algo.predict(str(user_id),
```

```
                                    str(test_id),
                                    r_ui = 0,
                                    verbose=True)

        predictions.append(prediction)
        tests_with_predictions.append([prediction[1],
                                       prediction[3],
                                       get_elo_diff(user_id, prediction[1])])
        estimates += prediction[3]

    mae = surprise.accuracy.mae(predictions, False)

    average_estimate = round(estimates/len(tests_with_predictions), 2)
    sorted_tests_with_predictions = sorted(tests_with_predictions,
                                           key=lambda x: (x[1], -x[2]),
                                           reverse=True)

    above_average = round(sorted_tests_with_predictions[0][1] - average_estimate, 2)
    return [sorted_tests_with_predictions, above_average, mae]
```

LISTING 3: Implementation collaborative filtering selection

After the task selection algorithm (Algorithm 1) has filtered out the tasks with too low or too high difficulty, the collaborative filtering algorithm sorts the remaining task candidates based on the estimated likelihood of successful completion. The list is sorted so that the first element of the list is the task that the student most likely will answer correctly.

A step-by-step description of the *get_collaborative_filtering_estimates()* is as follows:

1. Retrieve all answers on the database.
2. Create a 'rating dictionary' which contains taskID, userID, and whether the answer was correct or false.
3. Convert the rating dictionary into a data structure called a DataFrame.
4. Set that the rating scale to start at 0 (incorrect answer) and to end at 1 (correct answer).

5. Load the data from the DataFrame into the collaborate filtering framework Surprise.
6. Create a subset of the data to be used as a training set.
7. Set k-NN Basic as the algorithm to be used for making predictions.
8. Train the model of the recommender system on the training data.
9. Predict, and store, the estimated likelihood of success on all the tasks previously chosen by the Elo skill rating.
10. Calculate the Mean Average Error of the predictions.
11. Calculate the average predicted success rate.
12. Sort the predictions based on the estimates from the collaborate filtering algorithm. If there are tasks with similarly predicted outcomes, sort those tasks based on the skill rating deviation from the user.
13. Calculate the difference between the average estimates and the best estimate. This is only logged and used as statistics for the authors.
14. Return the sorted list.

In essence, this complete algorithm takes a list of tasks with fitting difficulty and returns the same list sorted based on predicted success based on collaborative filtering.

Pandas and Surprise

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language[2]. Pandas was used in this thesis to structure data properly prior to the calculation of collaborative filtering estimates. It structures the data into matrices with the columns *userID*, *itemID*, and *rating* (Listing 3).

[2]https://pandas.pydata.org/

Surprise[3] is an open-source Python scikit for recommender systems. For tasks selection based on collaborative filtering, the *prediction_algorithms* package was used. More specifically the k-NN Basic algorithm. Surprise was also used to calculate the Mean Absolute Error (MAE) to measure the difference between the observed result and the predicted result from collaborative filtering.

k-NN Basic

The k-Nearest Neighbor (k-NN) algorithm is a non-parametric lazy learning algorithm which is well suited for classification problems. The implementation of the algorithm is in the Surprise library and was used to compare similarities between students. The only parameter used to compare students was whether the specific task was solved successfully by the user or not. Equation 4.9 demonstrates how the similarity between two users is calculated.

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u, v)} \tag{4.9}$$

4.4 Learning content

Before moving on to the experiment, a large set of tasks (math problems) with estimated difficulties was needed. Because Matistikk did not feature a method for importing tasks, they were manually transcribed from 8th-grade math curriculum books [47][48][49][50]. The tasks were constrained to only feature fractions. The books used to find tasks were borrowed from the NTNU library for teacher education. The process of importing the tasks was divided into three phases.

The first phase consisted of going through the chapters on fractions to transcribe and create the tasks in Matistikk. The total number of tasks needed was based on

[3] http://surprise.readthedocs.io/en/stable/index.html

the scale of the upcoming experiment. It was important to cover all skill segments with a sufficient number of tasks. This was crucial because a wide variety of tasks with similar difficulty greatly improves the ability to recommend specific tasks to specific users. If there were too few tasks, the recommender system would not be able to make specific recommendations. Every task was made into a multiple choice problem. This was done to simplify the auto-correcting. Using input fields may yield a better representation of the students' skill, but they also require a significant amount of parsing of the answers. The authors created the answer choices for the tasks that did not feature multiple choice in the curriculum books. After having transcribed and created over 300 tasks the next phase was initiated.

The second phase consisted of estimating difficulty ratings of the tasks. Although the Elo rating algorithm should find the correct ratings for the tasks after a sufficient amount of usage, a good initial estimate would shorten the time needed to calibrate the system, as well as give the first users more accurate recommendations. Each of the tasks created was given an estimated difficulty rating based partly on the difficulty scheme in the curriculum books, and partly on the judgment of the authors. The curriculum books had different color ratings for the different sub-chapters which somewhat represented difficulty. The colors represented the learning curve more than the difficulty. The blue chapter was the easiest and the tasks were defined (by the authors) to range from 1200 to 1500 difficulty rating. The yellow chapter represented the medium difficulty chapter. The yellow tasks were estimated to range from 1400 to 1700 rating. The red chapter was the hardest and consisted of tasks ranging from 1600 to 1800 in Elo rating. These color ratings were used as a baseline for the Elo ratings set by the authors. After setting the initial rating based on the color scheme, the authors went through every task manually to judge if the difficulty rating was appropriate. The initial Elo rating was applied directly to the tasks in the database.

The third phase consisted of both authors solving all of the tasks. This was performed in order to find errors in phrasing or in the multiple choice options.

The third phase also resulted in an additional judgment of the difficulty ratings of the tasks.

Chapter 5

Experiment and results

The infrastructure presented in Chapter 4 will be used and tested as presented in this chapter. The results and details about each phase are presented chronologically. The size and scope of this experiment introduced some limitations, which are described at the end of this chapter.

The hypothesis of this thesis is

- A combination of Collaborative Filtering and Elo rating will better represent the skill of a student, compared to using only Elo rating.

To test this hypothesis we decided to carry out an experiment. We divided the users into a test group and a control group. The test group will use our combination of Elo rating and collaborative filtering, and the control group will use only Elo rating. We will evaluate the final Elo ratings of the users with an independent two tail sample t-test to see if there is a statistically significant increase of ratings in the test group compared to the control group. This test is further described in Section 5.2.3.

This experiment is split into two phases. Phase one will evaluate the system with an empty database. This means there are no students for the collaborative filtering to base recommendations on in phase one. The performance of the recommender algorithm is therefore expected to be worse in phase one. Phase two will be initialized with the data gathered in phase one. Because of this, the cold start

problem should be mitigated to some degree. We expect to see a steeper increase in the accuracy of the recommendations in phase two. The users were all given the same initial Elo rating of 1300 for the first phase and 1400 for the second phase. The initial rating was raised between the test phases to better fit the ability level of the participants in the experiment. This was based on results from test phase one.

Phase	**Users**	**Duration**	**Empty DB**	**Initial rating**	**# of users**
One	Teacher Students	$\sim$ 45 min	YES	1300	23
Two	Teacher Students	$\sim$ 45 min	NO	1400	25

TABLE 5.1: Experiment phase details

All of the tasks used in the experiment was in the form of multiple choice, with 3-4 options in each task. All tasks had exactly one correct answer. The types of tasks given varied in layout (textual, figures, plain calculations). Examples of tasks are shown in Figure 4.3. There were over 300 tasks in the database with an initial task difficulty ranging from 1200 to 1800.

The two test phases were performed on March 1st and 8th, 2018, in a standard classroom-setting at the Department of teacher education, NTNU. A total of 48 first grade teacher students participated in total (23 in phase one, and 25 in phase two). It should be noted that the users in the two different phases were not in the same class, meaning that none of the test users participated in both phases.

Approximately half of the participants were placed in the test group and the other half in a control group. They were assigned to a group automatically by the system, upon starting the test. The users were never informed about the groups or which group they were placed in. The test group was given tasks based on their Elo rating combined with suggestions from collaborative filtering. The control group was given recommendations only based on their current Elo rating.

The experiment was conducted during a normal lecture, which was paused for the entire duration of the experiment. At the very beginning of the lecture, the test users were given some background information about the system and what we wanted to test. They were also given a short demo of how the system works, and how they should proceed. This part of the experiment took about 15 minutes. After this, the test users started the test, by solving as many tasks as they possibly could in 45 minutes using their own computers and the university network. During the experiment, the system developed was hosted on a server running on the university network. The participants were allowed to use pen and paper to aid them, but no calculators nor collaboration was allowed during the test. The authors were present during the whole experiment to answer questions the users may have, or to deal with issues that may occur. The procedures for the test were:

1. Navigate to the server URL.
2. Enter the test environment (fig. 4.1).
3. Read the instructions and click the "Start test" button (fig. 4.2).
4. Answer the presented task (fig. 4.3)
5. Click the "Send inn" button to submit their answer.
6. Click "Logg ut" to quit, or click "Neste oppgave" to go to step 4.

Upon completion of a task, a new task automatically calculated and presented, as described in Section 4.2.

5.1 Data collection

When a user initiated the test, a test user account was automatically created and logged in behind the scenes. This was done to simplify the process for the user while carefully protecting their privacy. The user was not prompted to input any information besides answers to the tasks presented. The users were given

information concerning their privacy and how the data collected would be handled, before starting the test (fig. 4.2). At the end of the test, the user would click the "Logg ut"-button, or close the browser window to quit. Upon completion, a small questionnaire about perceived difficulty was given. The users were asked to rate the fairness of the tasks given in terms of difficulty on a scale from 1 to 5, where 1 represents too easy and 5 is too hard. The purpose of this questionnaire is to get an estimate of the difficulty from the users' point of view.

For every active user and every task completed, the system created a log file in CSV-format which was stored on the server. The task log files were updated with the values shown in Figure 5.1 after a task was completed by a user. The user log files were updated with the values from Figure 5.2 after every task the user completed. By saving the progression throughout the test we were able to monitor the progression of the ratings and to evaluate the recommendations provided by the collaborative filtering algorithm. The log files also contained user id or test id, and whether the user was in the test group or the control group. The log files were named by id and name.

```
TEST ID: 112
User,User-Elo,Test-Elo,K-value,Success,Time
1462,1564,1519,40.00,0,2018-03-08 12:34:42.324172
1358,1508,1539,40.00,1,2018-03-08 12:43:41.710075
1360,1570,1524,40.00,0,2018-03-08 12:46:46.804529
1341,1559,1510,40.00,0,2018-03-08 12:57:27.093235
```

FIGURE 5.1: Example of task log file

```
USER ID: 1355
Test,Test-Elo,User-Elo,K-value,Success, Collab filtering, MAE, Time
197,1352,1432,104.17,1,0.0,0.7341,2018-03-08 12:21:23.629357
277,1412,1477,89.29,1,0.32,0.6812,2018-03-08 12:23:32.125589
222,1443,1508,78.12,1,0.47,0.5348,2018-03-08 12:29:43.514363
270,1446,1524,69.44,1,0.41,0.477,2018-03-08 12:31:56.681922
273,1516,1561,62.50,1,0.45,0.4435,2018-03-08 12:32:30.745528
32,1521,1581,56.82,1,0.35,0.4883,2018-03-08 12:33:35.422711
268,1579,1614,52.08,1,0.26,0.4422,2018-03-08 12:36:41.624414
33,1614,1645,48.08,1,0.21,0.358,2018-03-08 12:39:33.140790
287,1631,1669,44.64,1,0.24,0.2419,2018-03-08 12:47:43.602221
242,1695,1650,41.67,0,0.0,0.0,2018-03-08 12:50:26.859327
```

FIGURE 5.2: Example of user log file

5.1.1 Data filtering

Some of the test data was removed prior to the analysis. This was done due to incomplete data. Users that had completed too few tasks were removed prior to the analysis. The lower limit of tasks completed was set to include as many of the users as possible without making the sample size too small. The limit for the first test phase was 16 tasks and 10 for the second test phase. In phase one, one user was removed from the control group. In test phase two, three users from the control group and one user from the test group were removed.

5.2 Test results

The results are presented in their respective phases. This is done because the data from phase one was carried over to phase two.

5.2.1 Test phase one

Out of 22 participants in the first phase, the control group consisted of 13 users and the test group consisted of 9 users. The numbers presented in Table 5.2 does not differ between users in the test group and in the control group. This is meant to give an overall summary of the test phase. The results show that the users averaged at 29.82 tasks completed, which means about 1.5 minutes per task on average. The average Elo rating for the users was 1592.09. The median value also supports this result. The standard deviation was 98.78.

Users total (test group/control group)	22(9/13)
Number of tasks completed in total	656
Average number of completed tasks per user	29.82
Average Elo for users	1592.09
Median Elo for users	1596
Standard deviation for users (Elo)	98.78

TABLE 5.2: Key numbers from test phase one

Figure 5.3 illustrates the average Elo rating of both user groups after a number of completed tasks.

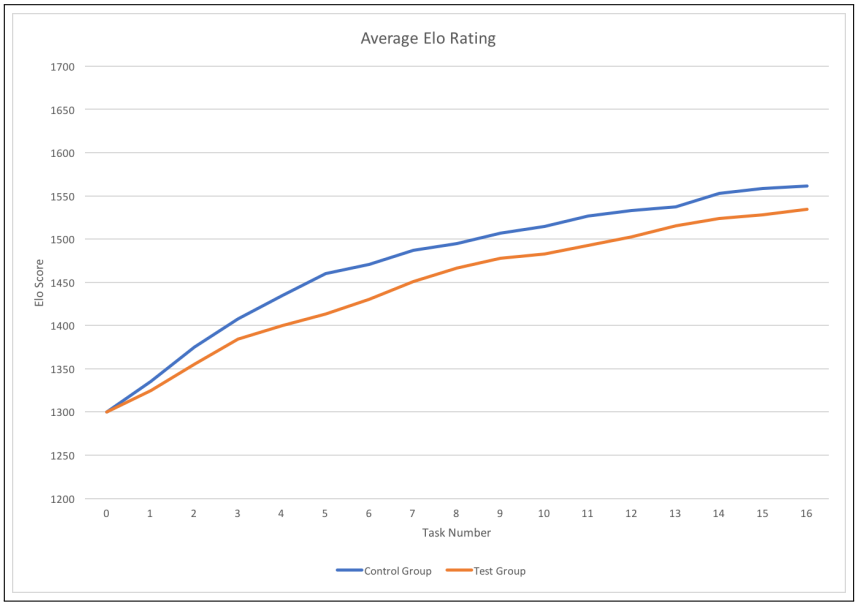

FIGURE 5.3: Average Elo rating in test phase one

Task Elo calibration

The Elo ratings of the tasks were initially set by the authors. This procedure is described in detail in Section 4.4. As users complete tasks the system will calibrate the task Elo rating over time. The initial and final distribution is shown in Figure 5.4. After the test phase, we can see that the distribution is getting closer to a normal distribution as more tasks get pushed against the middle. We also observe that some of the harder tasks were solved by some students and the Elo rating of these tasks decreased.

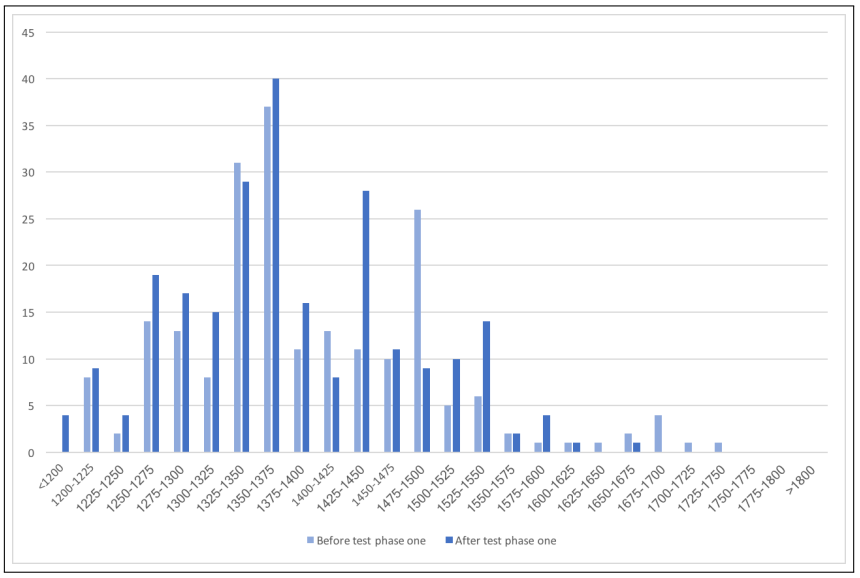

FIGURE 5.4: Test Elo distribution before and after test phase one

5.2.2 Test phase two

For test phase two we decided to increase the initial rating for each user from 1300 to 1400. By increasing the initial Elo rating, the skill rating of the user could be found more efficiently as for most users the Elo ranking will converge faster. On average the users in test phase one used the 3-4 first tasks to only reach the rank of 1400.

Test phase two consisted of 21 users in total, spread between 9 users in the test group, and 12 in the control group. On average, they completed 16.71 tasks each. This means around 2.7 minutes per task.

Users total (test group/control group)	21(9/12)
Number of tasks completed in total	351
Average number of completed tasks per user	16.71
Average Elo for users	1565.18
Median Elo for users	1565
Standard deviation for users (Elo)	66.71

TABLE 5.3: Key numbers from test phase two

Figure 5.5 shows the average Elo rating through test phase two. the average Elo rating converges faster, in this case to $\sim$ 1550. This is as expected since we decided to raise the initial user Elo ratings.

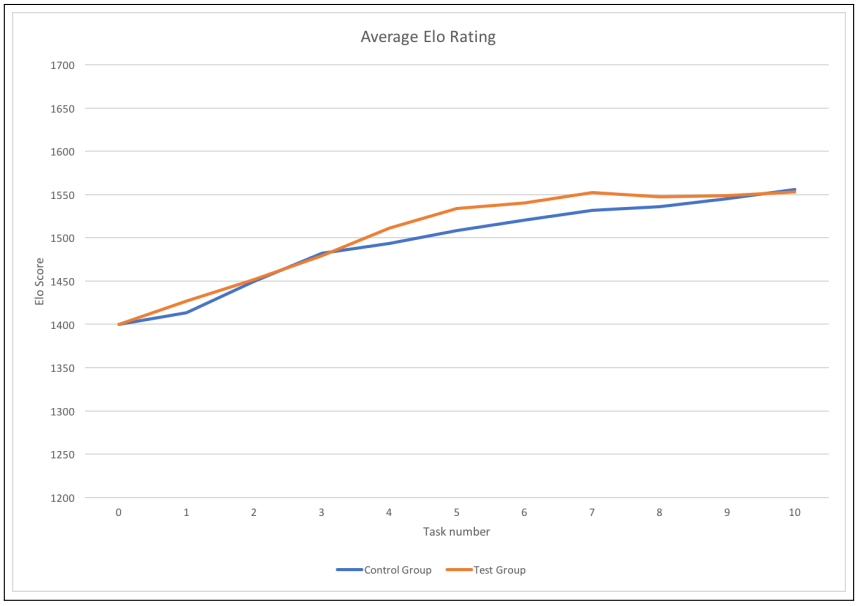

FIGURE 5.5: Average Elo rating in test phase two

Task Elo calibration

The data from test phase one was carried over to the second test phase. This means that the final ratings from phase one was used as initial ratings in phase two. This means the task difficulty calibration from phase one continued in phase two. Figure 5.6 shows that the distribution is even closer to the Gaussian distribution. That means after 1007 tasks completed in total, by 43 users, we have a good estimate for the difficulty of the tasks.

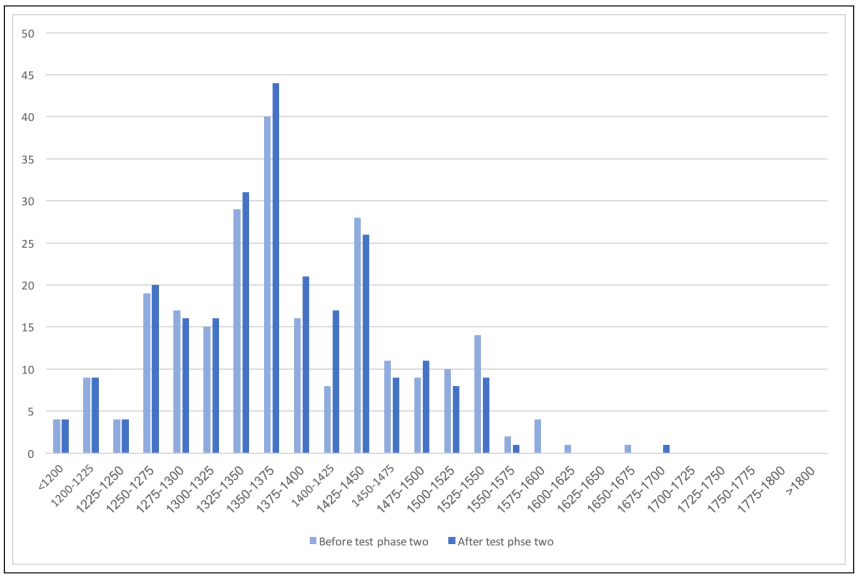

FIGURE 5.6: Test Elo distribution before and after test phase two

5.2.3 t-test

The main hypothesis of this thesis is

- A combination of Collaborative Filtering and Elo rating will better represent the skill of a student, compared to using only Elo rating.

The presumption for this hypothesis is that a better skill estimate results in a higher Elo rating. We believe this is a fair presumption. The Elo rating algorithm works independently from collaborative filtering, meaning that any increase in Elo rating is the result of a student solving a task with a higher rating, i.e. a harder task.

As a part of testing the hypothesis, we will test if there is any improvement in Elo rating for the test group compared to the control group.

- H_1 : A combination of Collaborative Filtering and Elo rating will result in an increased Elo rating for the test group compared to the control group.
- H_0 : There is no statistically significant difference in the Elo rating between the test group and the control group.

in order to prove the hypothesis H_1, we must be able to reject hypothesis H_0. We will conduct this test with a confidence level of 95%. For the degrees of freedom, we have used 20 for phase one and 19 for phase two.

To be able to reject the hypothesis H_0 for both phases the t-value must be higher than the critical t-value for the confidence level of 95%.

	Phase one	**Phase two**
Critical t-value	2.086	2.093

TABLE 5.4: Critical t-values

The t-values for both phases were calculated as shown in Equations 5.1 and 5.2.

$$t - value_{p1} = \frac{|1595.11 - 1589.08|}{\sqrt{\frac{5238.86}{9} + \frac{14278.91}{13}}} \approx 0.1460 \tag{5.1}$$

$$t - value_{p2} = \frac{|1553.78 - 1576.58|}{\sqrt{\frac{3878.19}{9} + \frac{4588.81}{12}}} \approx 0.7997 \tag{5.2}$$

As we can see from the calculations the t-value is far lower than the critical t-values required to reject the hypothesis H_0.

We can therefore state that there is no statistically significant difference in Elo rating when using the combination of Collaborative Filtering and Elo rating.

	Phase one	**Phase two**
p-value	0.8944	0.4394

TABLE 5.5: p-values from the t-test

The entire data-set and results from the t-test can be studied in Appendix C.

5.2.4 Collaborative Filtering accuracy

Figure 5.7 shows the difference and the development of Mean Absolute Error (MAE) average during the two test phases. Figure 5.7 shows that the MAE

descends more rapidly in test phase two.

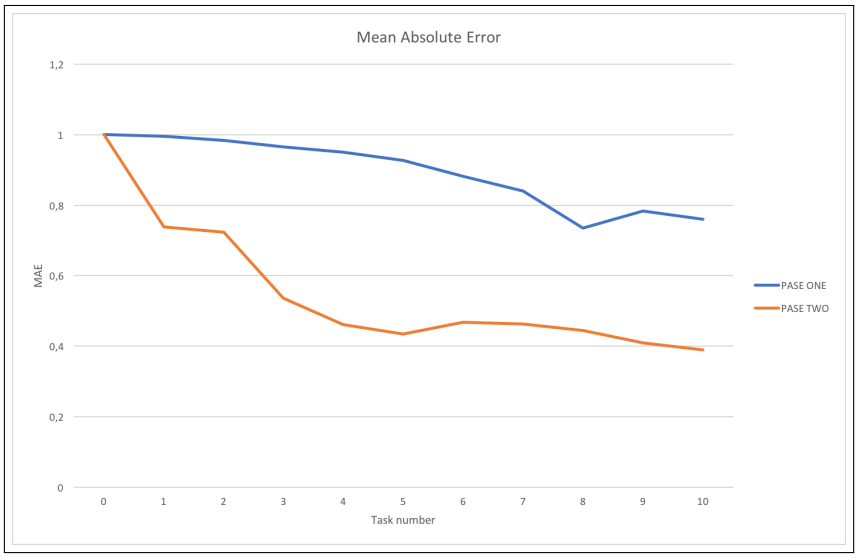

FIGURE 5.7: Mean Absolute Error average

5.2.5 Feedback from users

After completing the test, the users were asked to evaluate the difficulty of the tasks they were given. This data was collected using Google Forms[1] directly after completing the test. Responding to this survey was optional.

Out of 23 participants in test phase one, 18 responded to this survey. The average score of the responses was 2.944. Out of 20 participants in phase two, 16 of the users responded. The average score on difficulty fairness of this phase was 3.375. Figure 5.8 shows all the results collected in this survey.

[1]https://www.google.com/intl/en_en/forms/about/

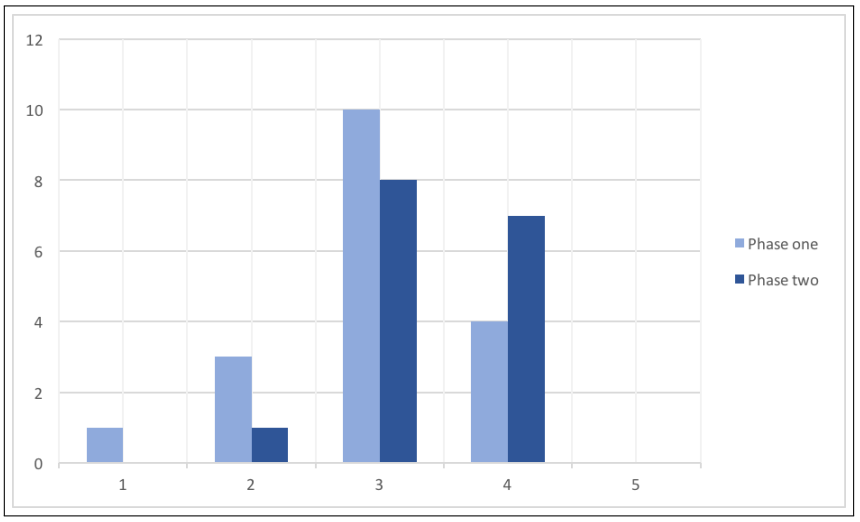

FIGURE 5.8: Task difficulty fairness score

5.3 Limitations

This experiment has a few limitations that imply the future directions of this study. First, the sample size ($n = 48$) is relatively small. However, by analyzing 1039 tasks completed over two test phases provided a clear data-set, and allowed to present this proof of concept for the proposed architecture.

Another limitation of this experiment is the participants. Tasks in the database were suited for 8th-graders (12-14 years), while the participants in this experiment where first-grade university students. This meant that the students needed to complete several trivial tasks before being presented challenging tasks.

Besides conducting another study with a bigger sample size to enrich the analysis, other estimation algorithms than k-NN Basic could be beneficial if investigated further.

Chapter 6

Discussion and future directions

6.1 Conclusion

This master thesis details a proof-of-concept software system for estimating student abilities. The system combines Elo rating and collaborative filtering in order to present students with tasks that best reflect their current skill level. The system was developed successfully. A substantial set (> 300 tasks) of 8th-grade math tasks was added to the system, and the system was tested on a population of students without technical issues. The following tasks were completed during the development, testing, and writing of this thesis:

- Designed a hybrid method for recommendations of tasks in e-learning.
- Implemented the same method in an existing e-learning system.
- Conducted two experiments, simulation and testing with users.
- Proved the feasibility of the approach and provided a proof of concept.
- Performed a t-test to test if there is a statistically significant difference in Elo rating between the control group and the test group.

The hypothesis we were testing in this thesis was:

- A combination of Collaborative Filtering and Elo rating will better represent the skill of a student, compared to using only Elo rating.

The t-test for phase one and phase two show that there is no significant difference between the Elo ratings in the control group and the test group in either test phase. Therefore, the hypothesis H_0 can not be rejected. With p-values of 0.8944 and 0.4394, the difference between the two groups is far from significant. The small sample size influences this result to some degree. Increasing the number of participants in the two test phases would cause a lower p-value.

We can not draw a conclusion on whether or not the collaborative filtering algorithm resulted in a better Elo skill rating of the users. However, Figure 5.7 shows that the accuracy of the recommender algorithm was increasing throughout both test phases. As expected, the accuracy of the recommender algorithm increased faster when the database was initialized with data from phase one. Because the accuracy kept increasing throughout both test phases, we do not know how accurate the recommendations may become over time, but this result is promising. A bigger experiment is needed to fully evaluate the potential of using collaborative filtering in combination with the Elo rating algorithm.

The automatic calibration of task difficulty has shown pleasing results. After the experiment was concluded, the difficulty ratings were converging towards a Gaussian distribution (Figure 5.6), which is to be expected when the tasks cover sufficient parts of the difficulty scale.

The feedback regarding the difficulty level received from the users after the experiment was conducted indicates that the tasks given were at an appropriate level. The difficulty reported in the survey, 2.944 was rated slightly lower than average (3.000) by the users after the first phase. This was partially the reason for the increased initial rating in phase two. In addition to the feedback from the users, the initial Elo rating was increased to reach the point of convergence earlier, and spend less time climbing towards the true skill level of the participants, which was expected to be higher than the main audience.

The phase two participants rated the difficulty higher with a score of 3.375. This may be because the participants did not need to complete trivial tasks with a low rating before being placed within their correct skill level.

6.2 Future work

There are multiple potential improvements that can be made to the proposed solution of this thesis. These improvements were not implemented either because they required more time than was available, or they would make the solution too complicated to accurately analyze the results.

6.2.1 Applying the recommender system to answers

The recommender system part of the proposed solution creates similarity ratings between users based on whether the users answered the tasks correctly or falsely. Another approach is to base similarity on the exact answer given. Because the tasks used are exclusively multiple choice, creating similarity ratings based on what multiple choice option was chosen may yield more accurate similarities. In essence, this means that the system could cluster users that not only answered the same problems correctly or falsely, but gave exactly the same answer. This can then be used to find more specific clusters of users. With more specific clusters, the system should perform better at recommending fitting tasks. In addition to better recommendations, finding smaller clusters of students may assist a teacher in finding groups of students that have the same lack of knowledge within a specific area of the curriculum.

The reason this was not implemented was because the size of the experiment that was conducted. With a small group of participants, dividing the clusters into more specific clusters increases the impact of the cold start problem. If the proposed solution was to be tested in a real-life scenario on a bigger scale, this feature may yield better results.

6.2.2 Using Time Spent

The LMS Matistikk has built-in functionality for recording the time it takes a user to complete a task. This is currently not being used in the proposed solution, but it has a few potential use cases. Time spent could be used as another parameter for the recommender system. This would cluster students that work quickly or slowly together in their respective groups. The authors did not find any documented evidence that time spent on tasks correlates with what type of tasks the user will answer correctly. Because of this, as well as to keep the proposed solution simple, this was not implemented. However, the correlation may exist, and it may be worth looking into.

Time spent could also be used to make the Elo calculations reflect how well the user did. However, time spent does not always equal the effective time spent. Users could load a task, do something else for a while, and then go back to solve it. A solution where exceedingly long times were discarded was considered, but it was dismissed to keep the solution from becoming too complicated.

6.2.3 Using Elo calculations in a non-binary way

The Elo calculation algorithm used in the proposed solution operates in binary. The leaner either passed (1) or failed (0) the task they were given. The Elo calculation could be modified to reflect a degree of success. An example of this could be to scale the calculations by how much time was spent on the task such that a fast correct answer gives a higher change in rating. The presumption is that users that have mastered the topic should be able to answer quicker than those who are unfamiliar with the topic.

It is also possible to give the users multiple attempts at each task but weight the calculation differently if the task was completed on the second or third attempt. In practice, this could be to multiply the Elo rating gained, by a fraction, if the task was completed on the second attempt.

6.2.4 Alternative ways of combining Elo and RS

In the proposed solution, the Elo rating algorithm is used to choose tasks with appropriate difficulty. Afterward, the collaborative filtering algorithms sorts those tasks based on the estimated probability of the user answering the task correctly. An alternative approach to this could be:

The Elo rating algorithm and collaborative filtering both return a probability of success. This allows for calculating an average estimate for the total probability of success. A task can then be selected based on this estimate.

Appendix A

Elo calculations in python

```python
import math

def calculate_expected_success_for_user(user_elo, task_elo):
    res = 1 / (1 + math.pow(10, (-(user_elo - task_elo) / 100)))
    return res


def calculate_expected_success_for_task(task_elo, user_elo):
    res = 1 / (1 + math.pow(10, (-(task_elo - user_elo) / 100)))
    return res


def calculate_new_elo_for_user(k_value, user_elo, result, expected_success):
    res = user_elo + k_value * (result - expected_success)
    return round(res)


def calculate_new_elo_for_task(k_value, task_elo, result, expected_success):
    res = task_elo + k_value * (result - expected_success)
    return round(res)
```

Listing 4: Implementation of Elo calculations

Appendix B

Implementation of Models

```
class Person(AbstractUser):

    grades = models.ManyToManyField(Grade,
                                    default="",
                                    blank=True,
                                    verbose_name="klasse")
    SEX = [
        ("M", "Gutt"),
        ("F", "Jente")
    ]
    ROLE = [
        (1, "Elev"),
        (2, 'Laerer'),
        (3, 'Skoleadministrator'),
        (4, 'Administrator')
    ]
    sex = models.CharField(max_length=1,
                           choices=SEX,
                           verbose_name="kjonn",
                           null=True)
    date_of_birth = models.DateField(max_length=8,
                                     verbose_name='Fodselsdato',
                                     null=True)
    role = models.IntegerField(choices=ROLE,
                               default=1,
                               verbose_name='brukertype')
    tests = models.ManyToManyField('maths.Test',
                                   blank=True,
                                   verbose_name='tester')
    elo_rating = models.IntegerField(default=1400,
                                     verbose_name='ELO rating')
```

```
32      inControlGroup = models.BooleanField(default=False,
33                                           help_text='',
34                                           verbose_name='Kontrollgruppe')
```

Listing 5: Implementation of the Person model

```
class Test(models.Model):

    task_collection = models.ForeignKey(TaskCollection)
    published = models.DateTimeField(verbose_name='Publisert')
    dueDate = models.DateTimeField(verbose_name='Siste frist for besvarelse',
                                   null=True,
                                   blank=True)
    randomOrder = models.BooleanField(default=False,
                                      verbose_name='Tilfeldig rekkefolge',
                                      help_text='')
    strictOrder = models.BooleanField(default=False,
                                      verbose_name='Laas rekkefolge')
    public = models.BooleanField(default=False)
    elo_rating = models.IntegerField(default=1500,
                                     verbose_name='ELO rating')
```

Listing 6: Implementation of the Test model

Appendix C

t-test

	Test group	Control group
	1565	1732
	1593	1750
	1629	1661
	1488	1579
	1599	1709
	1617	1537
	1620	1487
	1736	1663
	1509	1466
		1478
		1388
		1505
		1703
Mean	1595.11	1589.08
Standard deviation	72.38	119.91
Variance	5238.86	14378.41
n	9	13
Degrees of freedom		20
Critical t-value		2.086
p-value		0.8944
t-value		0.1460

TABLE C.1: T-test for phase one

	Test group	**Control group**
	1584	1540
	1612	1631
	1650	1604
	1511	1564
	1464	1560
	1542	1484
	1596	1645
	1479	1565
	1546	1620
		1661
		1431
		1614
Mean	1553.78	1576.58
Standard deviation	62.28	67.74
Variance	3878.19	4588.81
n	9	12
Degrees of freedom		19
Critical t-value		2.093
p-value		0.4394
t-value		0.7997

TABLE C.2: T-test for phase two

Bibliography

[1] Arpad E Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[2] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40 (3):56–58, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245121. URL http://doi.acm.org/10.1145/245108.245121.

[3] Morten Flate Paulsen. Online education : learning management systems : global e-learning in a scandinavian perspective, 2003.

[4] Eva Kaplan-Leiserson. E-learning glossary. URL http://www.lupi.ch/Schools/astd/astd2.htm.

[5] Vesin B. Ivanovic M. Budimac Z. Klasnja-Milicevic, A. E-learning personalization based on hybrid recommendation strategy and learning style identification. *Computers Education*, 56(3):885899, 2011.

[6] Harold Pashler, Mark McDaniel, Doug Rohrer, and Robert Bjork. Learning styles: Concepts and evidence. *Psychological Science in the Public Interest*, 9(3):105–119, 2008. doi: 10.1111/j.1539-6053.2009.01038.x. URL https://doi.org/10.1111/j.1539-6053.2009.01038.x. PMID: 26162104.

[7] Richard M Felder, Linda K Silverman, et al. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674–681, 1988.

[8] Huong May Truong. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. *Computers in Human Behavior*, 55(Part B):1185 – 1193, 2016. ISSN 0747-5632. doi: https://

doi.org/10.1016/j.chb.2015.02.014. URL http://www.sciencedirect.com/science/article/pii/S0747563215001120.

[9] Paul A. Kirschner. Stop propagating the learning styles myth. *Computers Education*, 106:166 – 171, 2017. ISSN 0360-1315. doi: https://doi.org/10.1016/j.compedu.2016.12.006. URL http://www.sciencedirect.com/science/article/pii/S0360131516302482.

[10] Gordan Durović. Educational recommender systems. *University of Rijeka, FHSS, Department of Polytechnics*. URL http://www.inf.uniri.hr/files/studiji/poslijediplomski/kvalifikacijski/Gordan_Djurovic-Kvalifikacijski_ispit.pdf.

[11] Peter Brusilovsky et al. Adaptive educational systems on the world-wide-web: A review of available technologies. In *Proceedings of Workshop" WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems (ITS'98), San Antonio, TX*, 1998.

[12] Stuart J Russell and Peter Norvig. *Artificial intelligence: A modern approach*. Pearson Education Limited, Edinburgh Gate, Harlow, Essex CM20 2JE, England, 3 edition, 2016. ISBN 1292153962.

[13] Aleksandra Klašnja-Milićević, Boban Vesin, Mirjana Ivanović, Zoran Budimac, and Lakhmi C Jain. *E-Learning Systems: Intelligent Techniques for Personalization*, volume 112. Springer, 2016.

[14] David J. Shernoff, Mihaly Csikszentmihalyi, Barbara Schneider, and Elisa Steele Shernoff. *Student Engagement in High School Classrooms from the Perspective of Flow Theory*. Springer Netherlands, Dordrecht, 2014. ISBN 978-94-017-9094-9. doi: 10.1007/978-94-017-9094-9_24. URL https://doi.org/10.1007/978-94-017-9094-9_24.

[15] Po-Ming Lee, Sin-Yu Jheng, and Tzu-Chien Hsiao. Towards automatically detecting whether student is in flow. pages 11–18, 2014.

[16] Sami Abuhamdeh, Mihaly Csikszentmihalyi, and Baland Jalal. Enjoying the possibility of defeat: Outcome uncertainty, suspense, and intrinsic motivation. *Motivation and Emotion*, 39(1):1–10, Feb 2015. ISSN 1573-6644. doi: 10.1007/s11031-014-9425-2. URL https://doi.org/10.1007/s11031-014-9425-2.

[17] Jan Papoušek and Radek Pelánek. Impact of adaptive educational system behaviour on student motivation. In Cristina Conati, Neil Heffernan, Antonija Mitrovic, and M. Felisa Verdejo, editors, *Artificial Intelligence in Education*, pages 348–357, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19773-9.

[18] S. Klinkenberg, M. Straatemeier, and H.L.J. van der Maas. 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, 2011. ISSN 0360-1315. doi: https://doi.org/10.1016/j.compedu.2011.02.003. URL http://www.sciencedirect.com/science/article/pii/S0360131511000418.

[19] Theo J. H. M. Eggen and Angela J. Verschoor. Optimal testing with easy or difficult items in computerized adaptive testing. *Applied Psychological Measurement*, 30(5):379–393, 2006. doi: 10.1177/0146621606288890. URL https://doi.org/10.1177/0146621606288890.

[20] C Spearman. The proof and measurement of association between two things. *International Journal of Epidemiology*, 39(5):1137–1150, October 2010. ISSN 0300-5771.

[21] Frederic M Lord. Statistical theories of mental test scores, 1968.

[22] Linda Crocker and James Algina. *Introduction to classical and modern test theory.* ERIC, 1986.

[23] The National Council on Measurement in Education. Glossary of important assessment and measurement terms, 2018. URL http://www.ncme.

org/ncme/NCME/Resource_Center/Glossary/NCME/Resource_Center/Glossary1.aspx?hkey=4bb87415-44dc-4088-9ed9-e8515326a061#anchorC.

[24] Xinming An and Yiu-Fai Yung. Item response theory: what it is and how you can use the irt procedure to apply it. *SAS Institute Inc. SAS364-2014*, 2014.

[25] Georg Rasch. Probabilistic models for some intelligence and attainment tests, 1980.

[26] David Andrich. Distinctions between assumptions and requirements in measurement in the social sciences. *Mathematical and theoretical systems*, 4:7–16, 1989.

[27] David Andrich. Controversy and the rasch model: A characteristic of incompatible paradigms? *Medical Care*, 42(1):I7–I16, 2004. ISSN 00257079. URL http://www.jstor.org/stable/4640697.

[28] Arpad E Elo. The rating of chess players past and present, 1978.

[29] Radek Pelnek. Applications of the elo rating system in adaptive educational systems. *Computers Education*, 98(Supplement C):169 – 179, 2016. ISSN 0360-1315. doi: https://doi.org/10.1016/j.compedu.2016.03.017. URL http://www.sciencedirect.com/science/article/pii/S036013151630080X.

[30] B. Smith and G. Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18, May 2017. ISSN 1089-7801. doi: 10.1109/MIC.2017.72.

[31] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009. ISSN 1687-7470. doi: 10.1155/2009/421425. URL http://dx.doi.org/10.1155/2009/421425.

[32] Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007. ISSN 1931-0145. doi: 10.1145/1345448.1345465. URL http://doi.acm.org/10.1145/1345448.1345465.

[33] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065 – 2073, 2014. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.09.005. URL http://www.sciencedirect.com/science/article/pii/S0957417413007240.

[34] R. J. De Ayala. *The Theory and Practice of Item Response Theory*. Guilford Publications, 72 Spring Street, New York, NY 10012, 2095. ISBN 9781593858698.

[35] Michel C. Desmarais and Ryan S. J. d. Baker. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, 22(1):9–38, Apr 2012. ISSN 1573-1391. doi: 10.1007/s11257-011-9106-8. URL https://doi.org/10.1007/s11257-011-9106-8.

[36] Radek Pelánek, Jan Papoušek, Jiří Řihák, Vít Stanislav, and Juraj Nižnan. Elo-based learner modeling for the adaptive practice of facts. *User Modeling and User-Adapted Interaction*, 27(1):89–118, Mar 2017. ISSN 1573-1391. doi: 10.1007/s11257-016-9185-7. URL https://doi.org/10.1007/s11257-016-9185-7.

[37] Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, Dec 1994. ISSN 1573-1391. doi: 10.1007/BF01099821. URL https://doi.org/10.1007/BF01099821.

[38] Margit Antal. On the use of elo rating for adaptive assessment. LVIII, 01 2013.

[39] Hlj Van Der Maas and Ej Wagenmakers. A psychometric analysis of chess expertise. *American Journal Of Psychology*, 118(1):29–60, 2005. ISSN 0002-9556.

[40] Terry Nealon and Jim Butler. Systems and methods for providing a personalized educational platform, January 23 2014. US Patent App. 13/939,897.

[41] Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4):13:1–13:19, December 2015. ISSN 2158-656X. doi: 10.1145/2843948. URL http://doi.acm.org/10.1145/2843948.

[42] M. K. Khribi, M. Jemni, and O. Nasraoui. Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*, pages 241–245, July 2008. doi: 10.1109/ICALT.2008.198.

[43] Avi Segal, Ziv Katzir, Kobi Gal, Guy Shani, and Bracha Shapira. Edurank: A collaborative filtering approach to personalization in e-learning. In *Educational Data Mining 2014*, 2014.

[44] Refsns J N Myre H, Oldervoll S. Developing a web application for creating, solving, assessing and collecting data from interactive mathematical tasks in python/django, html5, javascript, css and mysql, 2017.

[45] Mark E Glickman. A comprehensive guide to chess ratings. *American Chess Journal*, (3):59–102, 1993.

[46] J. Ninan, R. Pelnek, and J. ihk. Student models for prior knowledge estimation. *Proc. of Educational Data Mining*, pages 109–116, 2015.

[47] Bjørn Bakke and Inger Nygjelten Bakke. *Grunntall 8*, volume 1. Elektronisk Undervisningsforlag AS, Pettersvollen 3, 3032 Drammen, 1 edition, 2006. ISBN 82-91813-16-7.

[48] Espen Hjardar and Jan-Erik Pedersen. *Faktor 8 oppgavebok*, volume 2. Cappelen Damm, Akersgata 47/49, 0180 Oslo, 1 edition, 2015. ISBN 978-82-02-44131-9.

[49] Grete Normann Tofteberg and Janneke et .al Tangen. *Maximum 8 Grunnbok*, volume 2. Gyldendal Norsk Forlag AS, Sehesteds gate 4, 0164 Oslo, 1 edition, 2013. ISBN 978-82-05-39501-5.

[50] Grete Normann Tofteberg and Janneke et .al Tangen. *Maximum 8 Oppgavebok*, volume 1. Gyldendal Norsk Forlag AS, Sehesteds gate 4, 0164 Oslo, 1 edition, 2013. ISBN 978-82-05-39510-7.