LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT
PROGRAMMIER- UND MODELLIERUNGSSPRACHEN

Bachelorarbeit

in Medieninformatik

Collective Peer Evaluation of Quiz Answers in Large Classes through Pairwise Matching

Martin Gross

Aufgabensteller:	Prof. Dr. François Bry
Betreuer:	Sebastian Mader
Abgabedatum:	19.06.2017

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 19.06.2017

. . . .
Martin Gross

Abstract

Backstage is a a digital backchannel for large scale lectures in a university setting. It offers interaction to the participants through various means like annotations and quizzes, which can be given in a variety of styles. These include multiple choice or open answer. This thesis aims to provide an additional type of interaction: allowing participating students to submit their answers to problems while also allowing them to competitively vote on said answers. After careful study of multiple algorithms, the voting is done through pairwise comparision and evaluated using *ELO score* and *True Skill*.

Results can be used in and outside the classroom as a means to gauge the students' understanding of taught materials and to evaluate unrelated university contents.

Backstage ist ein digitaler Backchannel für große Vorlesungen in einem universitären Umfeld. Es bietet den Teilnehmern Interaktion in mehreren Formen wie beispielsweise Anmerkungen oder Quizzen an, welche in verschiedenen Ausprägungen dargereicht werden können, beispielsweise als Multiple Choice Quiz oder in Form von offenen Antworten. Diese Arbeit hat sich zum Ziel gesetzt eine weitere Art der Interaktion einzuführen: Teilnehmenden Studenten soll die Möglichkeit gegeben werden, Antworten zu gegebenen Problemen einreichen zu können und im Anschluß über diese abzustimmen. Nach ausführlicher Betrachtung einer Vielzahl von Algorithmen wurde der Beschluss gefällt die Abstimmungen mit Hilfe von paarweisen Vergleichen durchzuführen und mit Hilfe der *ELO Zahl* und des *True Skill*-Algorithmus auszuwerten.

Die gewonnenen Erkenntnisse können sowohl im Rahmen der universitären Lehre als auch außerhalb verwendet werden, beispielsweise um das Verständnis von vermittelten Lehrinhalten zu überprüfen oder um komplett von Lehrveranstaltungen unabhängige Inhalte bewerten zu lassen.

Contents

Chapter 1

Introduction

For quite some time, the *Teaching and Research Unit Programming and Modelling Languages* at the *Institute for Informatics* of the *Ludwig-Maximilians University of Munich* has offered a new tool for teaching large-scale lectures: *Backstage*. The self-declared mission is to provide a “Digital Backchannel for Large Class Lectures” [18]. As such, *Backstage* offers features aimed at improving participation and interaction in the classroom. First of all, participants are able to add annotations to the presented slides. Those annotations also provide the mentioned *Backchannel*: The notes may be answered by other - students or lectures - though providing direct feedback in both directions. The lecturer is provided with valuable information where inconsistencies or passages that require more in depth explanations are present in his presentation while the students get direct answers and feedback to their questions. The other notable feature is the addition of quizzes with which the students may test their knowledge during the lecture [17].

In-classroom quizzes are nothing new. However, exploiting the potential of quizzes is difficult in larger classes. The moment a bigger group is supposed to participate, the results are no longer easily quantifiable. If more complicated questions are asked - for example requiring more than one answer (n-out-of-m correct answers), manual survey is nearly impossible [17]. It should also be noted, that shame and/or fear of public speaking might be a factor when examining participation patterns. *Backstage* circumvents these problems by offering a digital classroom which serves larger audiences.

Though multiple choice quizzes are offered, those are often limited to a single form interaction; namely, clicking the correct answer.

In this bachelor thesis, the possibility of adding more interactivity to *Backstage* by offering a competition between students' solutions to a given problem will be analysed. The motivation behind this is to provide a more entertaining way for students to follow a lecture and to provide a better motiviation to participate in classroom-quizzes.

From a high level point of view, this interactive unit is divided into three phases. During the first phase, students are presented with the possibility of submitting their answer to a predefined problem. In the second phase, students are invited to vote on their peers' solutions, in order to find the best. In the third and final phase, the voting is closed and results are calculated and presented. This approach is described in a very generic way, as the approach is very flexible in terms of the student-provided solutions being voted on. This input may be of any type - ranging from text to code to uploaded pictures.

As *Backstage* is mainly used in computer science - specifically programming-related - classes, the idea of asking students to solve small problems by submitting a piece of code is natural. So far, those solutions can however only be graded by either

- compiling them and running them against tests,
- verifying them manually, one by one, by a human, or
- matching the written code against a prepared solution.

It should be obvious that all of those three offered solutions lack in their own ways: Either they cannot be used for a large number of participants or they do not offer any kind of personalized feedback or do not take into account that there are many different ways to solve a problem.

Therefore, it is desirable to implement a mixture of the above solutions: A practical approach to add interactivity to lectures by offering a semi-automated ranking system.

It should be noted, that the obtained final result does not necessarily reflect the *best* solutions but the most popular ones. They can be used in various ways. First and foremost would be a display of the selected TOP X solutions during the lecture, offering a direct feedback to the lecturer and students. As there is no guarantee that the results actually reflect an ordered list of the not only subjectivly but factual best results, a quick overview if the tested concept has been correctly understood by the students can be gained.

As this interactive unit may also be used in an asynchronous way, meaning that the submission of solutions, the voting and the presentation of the results are not shown within a short timeframe during a lecture, the results may also be used in retrospect to provide targeted training for students during practise and review sessions.

1.1 Goal

This paper's goal is to implement an interactive unit into the *Backstage* application containing three phases that are scheduled to occur successively.

In the first phase, students are enticed to solve a given problem. In order to keep the unit flexible and allow for usage in a multitude of teaching contexts, the designated problem can be any media the *Backstage* application is able to handle. It may be for example a text-question or a picture that is displayed to the students accordingly, the answer to the problem may also be of any input the system is able to process. In order to stay with the mentioned example of a text-based or an image-based question, the students' answers may also be given by text or file upload.

As this system is quite flexible, it can allow to daisy-chain input-methods and validators. The input-field for a text-answer may be decorated with a validator for program-code. It can even submit its contents to a compiler to validate the output before allowing the student to proceed.

As soon as the first phase is concluded, the second phase that is dedicated to a voting process is initiated. During this phase, participating students are invited to vote on their fellow students' answers. The exact way in which this voting is done and which algorithms are being used to track the performance of each individual answer will be discussed in the third chapter of this thesis.

Upon conclusion of the voting-phase, a third and final phase offers the possibility to explore results that have been obtained through the students' vote. If this interactive unit was being used during a classroom-session, the results may be presented on the big screen by the lecturer. However it should be noted that the results are not necessarily in order from best to worst, as they have been judged by the students themselves: while they might be in the “correct” order, they should only be regarded as a list of the most popular solutions.

1.2 Stucture

After having discussed the current state of the *Backstage* system and the general idea of the voting mechanism to be implemented, this thesis will present related work in the broader field of interactive elements within learning environments and audience participation systems. This will be followed by a chapter discussing a variety of algorithms that have been taken into consideration for implementing this new interactive unit for *Backstage*. Finally, a chapter discussing actuial implementation will act as a closing section.

Chapter 2

Related Work

The term *lecture* is not a random one. In the 14th century, universities were distinctively different from their modern counterparts. A *lecture* was exactly that: a professor would read publications (either his or others') and comment on them [4]. Despite lectures being prevalent in today's university curiculums, litteral lectures are seldom.

Lectures tend to be a one-way-street, where students are on the receiving end of a monologue. Enduring sometimes multiple hours of talk without any active involvement can cause the audience to get distracted easily.

> In a lecture given by a brilliant scholar with an outstanding topic and a highly competent audience, ten per cent of the audience displayed signs of inattention within fifteen minutes. After eighteen minutes one-third of the audience and ten per cent of the platform guests were fidgeting. At 35 minutes everyone was inattentive; at 45 minutes, trance was more noticeable than fidgeting; and at 47 minutes some were asleep and at least one was reading. A casual check twenty-four hours later revealed that the audience recalled only insignificant details, and these were generally wrong. [6, p. 283]

Adding some kind of interaction to lectures has thus been an important mission to many scholars in the past years. Common tactics include the usage of patterns less associated with universities but rather "the classroom", specifically covering the learning-environments before the university-stage. These methods frequently include addressing questions to the listening public and inciting them to respond. However, as presence and participation is completely voluntary at this level of education [21, Art. 3, Sec. 4], a complete inclusion of all students cannot be achieved: contrary to highschool, the lecturer cannot pick out a specific student to answer a question or sentence him to more active participation.

The choice of (non-)participation is thus completely in the hands of the students - and many opt to maintain anonymity or hide possible incompetence by recusing themselves from answering or asking questions [16]. As everyone has lived to experience themselves being unprepared to answer a teacher's question in front of all their classmates in the past, this fear seems to be reasonable.

The obvious solution to this dilemma, is to find new ways to include students into the lecture equally - whether they are top performers and are only confirming their knowledge or they are lacking in knowledge and/or understanding and need further support to gain the necessary knowledge. In fact, it is important that the latter group become tended to, as it is known that students lacking specific knowledge often remain silent in class to avoid said embarrasment [20].

One frequently used system - especially in North America - consists of using so called *clickers*, often synonymously used with the abbrivations *SRS* (Student Response System), which in turn are a very specific version of an *ARS* (Audience Response System).

The idea of such systems - regardless if used in a teaching environment or for other purposes (like townhall or corporate meetings) - is that every participant is handed one *clicker*. The presenter may then ask a question which the participants answer by pressing the corresponding button on their device. The presenter has a receiver plugged into his presenting computer and can visualize the results in near real time. As the data is aggregated, it is not possible to sort out a single participant, thus encouraging the public to participate without the above mentioned

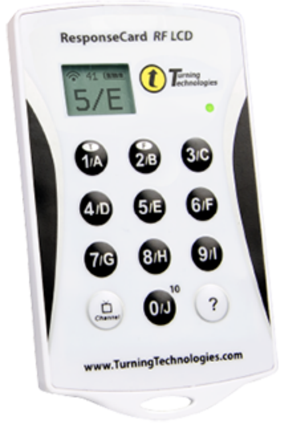


Figure 2.1: Example of a Turning Technologies Response Card [24]

fear of being called out for an incorrect answer[1].

A study by Martyn [12] showed the positive effects of using *clickers* in a classroom-setting compared to just implementing discussions among the students: the fact that "[...] many students are hesitant to respond to an answer until they know how others will respond" [12] is quoted as one the main reasons many abstain from participation - a theory Trees and Jackson [23] agree with.

In higher education where the exact number of participants is often not known beforehand, owning an equal number of *clickers* as students is not economically sane. Theft is also an issue. Universities require their students procure an answering device for themselves and that they subsequently register it with the university. While this removes the economic burden from the university's plate, it may still cause burdens to students

[1]For sake of completeness, it shall be mentioned that most *clickers* do indeed transmit a unique identifier (known as MAC-address) when sending their reply. Depending on the setup and preferences of the presenter, this MAC-address may be ignored or used to check against a predefined whitelist if the answer is to be counted or ignored. Manufacturers also offer a test-taking suite to universities and schools that enables students to pass their exams using their *clickers*: in this case the MAC-address is used as a unique identifier for the student and the entered answers. Of course, the selected answers of the students are not shown on the big screen.

with lower incomes: asking price for a *clicker* is around 40 USD.

The technical specification of such devices is also an issue. While early systems used infrared light to communicate, newer systems opted for using radio frequency transmission in license-free bands. Although RF communication is clearly a step ahead, it cannot be considered the perfect solution.

Infrared systems require a direct line of sight between the sending devices (the *clickers*) and the receiving device. In large lecture halls this might pose a problem that can only be combatted by using multiple receivers distributed across the room. This however means that either the installation is quite bothersome for an impromptu setup or that a fixed installation is necessary. Both these scenarios are inflexible as they require the system either be accessible at all times or in every room.

Contrary to that, radio frequency systems allow for a very compact receiver which does not need any fixed installation. As research by Travis Goodspeed has shown the most popular systems are built around the very well known and inexpensive Nordic Semiconductors nRF24E1 IC [7]. The used frequency-band by those devices is 2401 MHz to 2483 MHz. Unfortunately, another popular service is also using this 2.4 GHz-band: WiFi connections.

In times when WiFi was not as widespread as it is today, this may not have been a cause for concern - but today it is: using a RF-based SRS or ARS is just calling for trouble through interference - wether the WiFi is negatively impacting the SRS/ARS or the other way around. This has caused for a lot of universities to suspend usage of hardware-based SRS: McGill university of Montréal, Canada for example started asking its students and staff to return their (bought) clickers for recycling in exchange for a voucher for a free coffee [25]. (Keeping in mind the initial purchase costs, certainly an expensive free coffee.)

But SRS/ARS are not dead yet. In light of the omni-present cellphone and portable computer, manufacturers and universities have set out to reinvent old solutions. Basically every former manufacturer of hardware-based ARS is now also offering software-products that emulate the functionality of a *clicker* - offering this “new” solution to be hosted in the *cloud* or on premises.

Not a lot has to be said about those solutions as they - as mentioned above - emulate the usage of the well-known *clickers* on a personal device. Some may go the extra mile by not offering only a keypad for the answer but actually providing rich(er) content to be displayed unto the participants device - but the idea remains the same. One positive note should however be mentioned: for participating students, those solutions generally do not incure any further investments (of course apart from their devices, that most will already own).

Backstage, in its core, also implements such a solution when it comes to offering quizzes to the students. However it should be noted that *Backstage* is far more than just a simple interface to administer multiple-choice quizzes, as lined out in the introduction of this work and in the thesis by Alexander Pohl [17].

Weighing physical *clickers* against their virtual counterpart does indeed boil down to their accessibility as their functionality is the same for the most part. When deciding whether to use hardware or software-based systems, surrounding circumstances should be considered: in highly developed areas where every student is already equiped with a compatible device like a laptop or a cellphone, the usage of a computer-based ARS causes less costs to the individual student and allows for higher compatibility with other devices using the same frequency spectrum. For areas where personal computing devices are not yet that common, the dedicated clickers present themselves to be the better solution - even though initial purchases must be done. Seeing however as *clickers* are being phased out in major universities now, they might be offered a second life at a discounted price.

Regardless of the system that is used to incite students to participate more in lectures, interactivity should be regarded as a basic building block for every lecture and be used consistently.

Chapter 3

Basics and Concept

The basic idea behind this concept is to incite the students - no matter how many are participating in a lecture - to vote on submitted contributions by their fellow students to find the supposedly best solution to a given problem.

For the sake of brevity, this work will focus on the ranking of the solutions, which theoretically allows anything to be ranked: from program code, plain text, pictures or anything else that can be displayed to users.

Although there exists a multitude of different algorithms, this chapter will focus on the algorithms used by Round Robin tournaments, single- and double-elimination tournaments, as well as pairwise-matching. The latter category will be represented by the similar but distinctive algorithms ELO score and True Skill. This chapter will conclude with a selection of algorithms to by implemented as a means to concretize this paper's argument.

3.1 Terminology

While the upcoming algorithms speak about players who are to compete against each other, the described classroom setting of *Backstage* would have individual student's solutions compete against each other as the "players". Furthermore, a "round", "voting round", "encounter", or "voting" depicts the state of two solutions being put to the test against each other, while a "tournament" represents the entirety of all encounters.

Lastly, “staggered voting” represents a state where not all encounters can be played simultaniously but have to be broken down into multiple chunks which are then processed one after the other. In contrast, the “simultaneous voting” does not implement this limitation and allows all encounters to occur at the same time.

It should be noted again that the final result is not a selection of the player or student that is the best one, but rather the solution that is *percieved* to be the best one.

3.2 Round robin tournament

The first possible alogrithm - and possibly most evident one - is a round robin tournament. Such a tournament mode is used today for various soccer leagues around the world. The concept of the tournament is quite simple: every participant’s solution is contesting against every other solution.

Assuming a single-round tournament, n solutions will have to participate in

$$\frac{n}{2}(n-1)$$

individual encounters.

Assuming furthermore (and also for all other upcoming examples) $n = 300$ active solutions, this would lead to 44850 rounds that need to be played. It is obvious that this amount cannot be expected to be delivered in a university lecture’s timeframe.

Of course it is possible to modify the initial position to provide a double-round tournament in which each solution will not only compete against all others once but twice. In this case,

$$n * (n-1)$$

individual encounters (89700) would need to be played. Even though (if n is a pair number, one solution has to wait out and is not matched with a counterpart) $\frac{n}{2} = 150$ can play simultaneously during the $(n-1) = 299$ rounds, even this tournament mode is not fit for our purpose.

3.3 Single-elimination tournament

Another possible algorithm to rank the solutions is a single-elimination tournament - known for its usage among major sports leagues such as the UEFA.

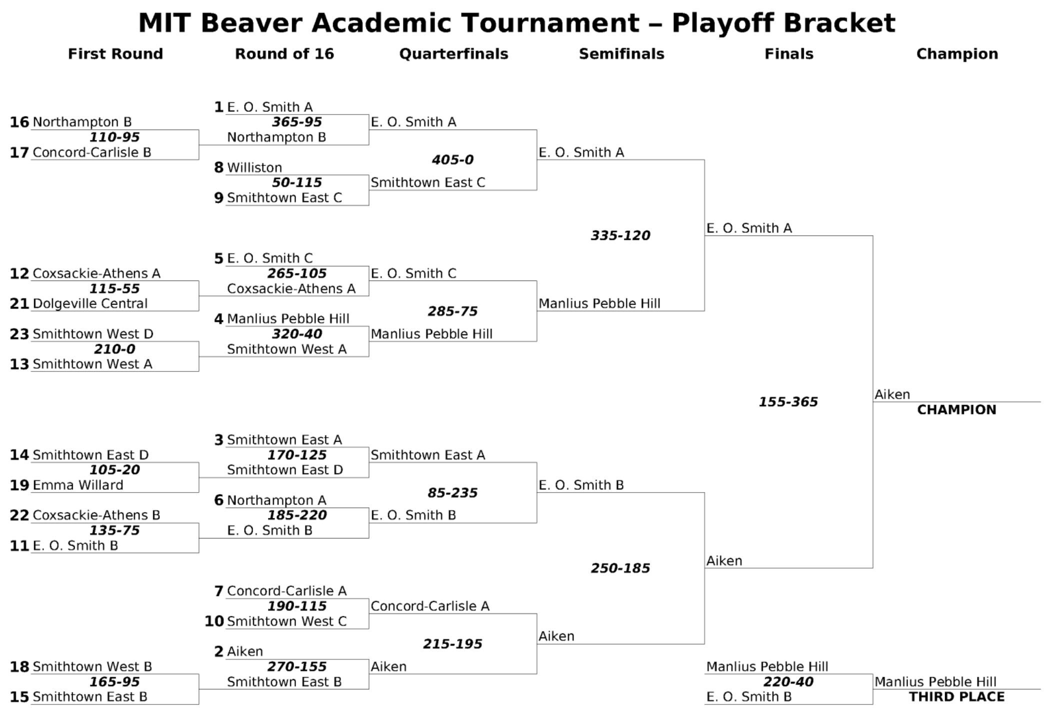


Figure 3.1: Example of a single-elimination bracket [1]

A single-elimination tournament is characterized by a number of solutions n competing against each other. The loser of a pairing is eliminated immediately.

The problem with using this algorithm is that it also scales poorly for a big amount of participating solutions as we face it: the number of required rounds for n solutions is $n-1$. At 300 solutions we would already require 299 voting rounds. Another major downside is that it cannot be run in an asynchronous way: it is impossible to offer a voting chance to every participant at the same time. It would be necessary to implement a staggered voting stage: staying with our example of 300 participating solutions,

$$\log_2 n = \log_2 300 \approx 9$$

staggered voting rounds would be necessary. This translates directly into a considerable amount of time that is necessary to complete those voting rounds. Of course, less than using a round-robin tournament but still considerable.

Those problems aside, this algorithm also has the major inconvenience of eliminating a solution from the tournament when it loses once. Of course this could be circumvented by, running a *best-of-n* mode for every round or even switching to a double-elimination tournament. This would however entail adding a substantial number of additional voting rounds.

3.4 Double-elimination tournament

A double-eliminiation tournament is quite similar to the above discussed *single-elimination tournament*, except that it offers greater flexibility. In a single-elimination tournament a participant's solution is eliminated right away after losing a pairing - even though it might have lost this encounter just by pure chance and not based on its quality. The double-elimination tournament is giving those losing solutions a second chance by letting them compete against each other in a secondary tournament bracket (prefixed with the letter L in figure 1.2). This allows a solution to still win the whole tournament if it has not lost more than once. This is a clear advantage over the single-elimination tournament, as it accounts for *good* solutions that might have accidentially judged a loser early on in the voting process.

But still, even this system has the major disadvantage of requiring staggered rounds to be played. At

$$2 * (n - 1) = 2 * (300 - 1) = 598$$

encounters that need to be played, it is also not assured, that this amount can be met in a classroom setting.

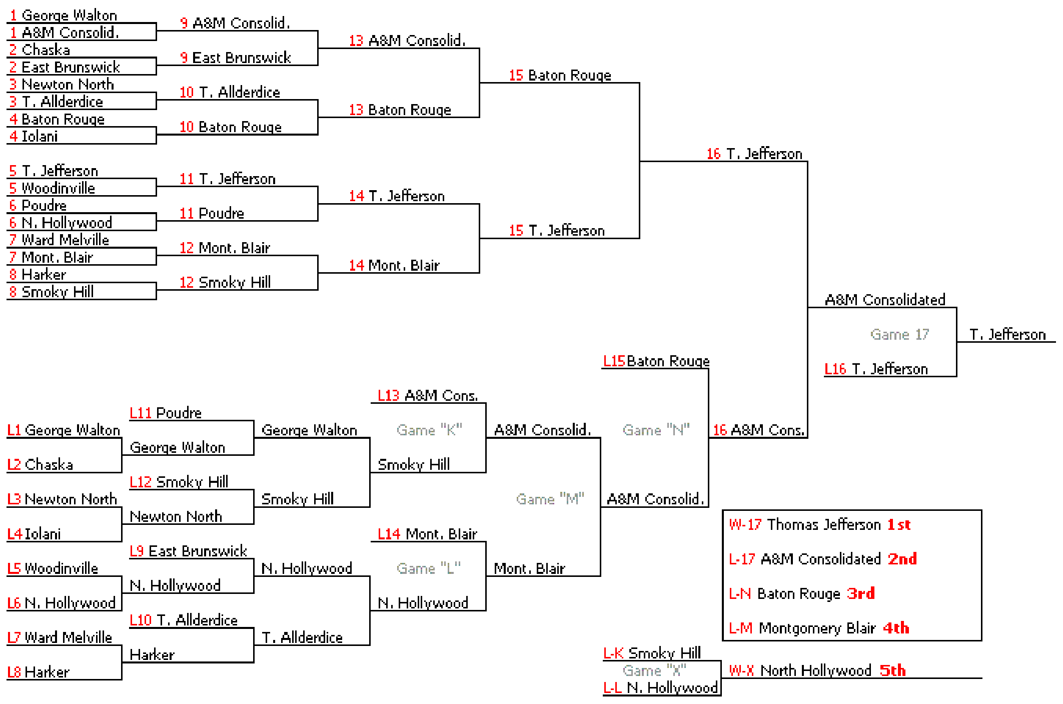
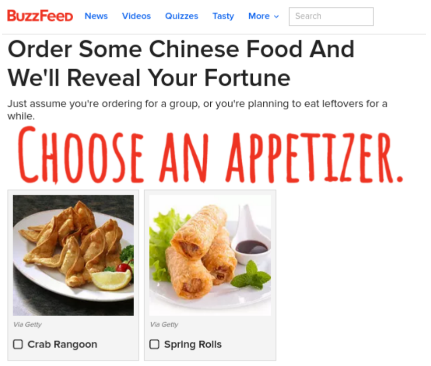


Figure 3.2: Example of a 10-entry double elimination bracket [15]

3.5 Pairwise Matching

Upon further research, Pairwise Matching - famously know for its usage on pages like “Hot or Not” or (now defunct) “babevsbabe.com”- was taken under scrutiny. Pairwise Matching is represented by having two randomly selected solutions being voted on. As a result, it is possible to rank all solutions by popularity.



Order Some Chinese Food And We'll Reveal Your Fortune

Just assume you're ordering for a group, or you're planning to eat leftovers for a while.

CHOOSE AN APPETIZER.

Figure 3.3: Example of pairwise comparison

The paper by Hacker and von Ahn [8] explores the usage of various pairwise matching algorithms in a similar setting. Two of the presented algorithms in said paper will be further explored in this work: ELO Score and TrueSkill.

3.5.1 ELO Score

The ELO Score has its origins in the chess world, where it is used to rank chess players according to their strength.

The basic idea behind this measure is to be able to rank any amount of solutions according to their strength - the higher their score, the better they are. Furthermore, by having two solutions contest against each other, their score is adjusted up or down - but in accordance to their respective strength [5]. In other words: if a solution with a higher score wins against a solution with a lower score, the win is not surprising and the winner is awarded a smaller amount of score points. However, if a weaker solution wins against a stronger solution, it gains a lot more score points. The same is valid the other way round for the losing scenario.

In order to calculate the point variance of each solution - whether losing or winning - one thing needs to be defined first. For starters, every solution in a tournament has a score R - even if they have not played yet. As such, it should not be 0 but rather an approximation of the actual strength of a player. In real-life chess, this is much less of an issue, as experienced players can give their opinion on a new player. In virtual spaces this is a lot more difficult, as there is no judging entity. Therefore, online chess clubs tend to offer new participants either a choice of various categories (new player, debutant, amateur, etc.) or just attribute a fixed number (often in between 1200 and 1800). Overall ELO scores range from under 1000 for amateurs to over 2500 for a select few grandmasters, 2882 being the recorded highest active score of Magnus Carlsen in 2014 [3].

Either way, a chess amateur with a more or less random assigned R value might experience within the first few games a substantial loss (or win) of played games until he has reached his adequate ELO score.

It should be noted that the ELO score is a relative measure and not an absolute one: it ranks the participants in relation to each other - it cannot be used as an absolute measure across people that have never interacted with each other.

In our case, we can profit from this property by assigning all solutions a fixed but random starting value of $R > 0$, as the result will only be a ranking within the participating solutions.

In a first step, the expected points for a pair of solutions is calculated using the following formula:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

where E_A are the expected points for the solution A and R_A and R_B are the current ELO scores of solution A and B, respectively. The divisor of 400 should be ignored, as it yields no importance or difference to this thesis' usecase[1]. As the accumulative amount of points to be distributed between two participants (before weighing them using the solutions individual score) is one, the expected points for R_B are $1 - R_A$.

In a second step, the new ELO-score of both solutions is calculated. This is done using the current score of the participants R, a fixed scaling-factor k, a fixed value S depicting one of the three possible outings (win, loose, tie) and the expected points E from step one using the following formula:

$$R'_A = R_A + k * (S_A - E_A)$$

The selection of the scaling-factor k is of importance in the chess world, as it dictates how much of in impact an individual game has on the score of the participating players. While, in general, $k = 20$ is being used, it can be reduced down to $k = 10$ for very strong players ($R > 2400$) or even incremented to $k = 40$ for amateur players with a score $R < 2300$. As our scores are not used over a longer period of time, an arbitrary value can be chosen.

The factor S depicts for each participant the outcome of the played round from *their own point of view*. The winning party is attributed $S = 1$, while the losing party is attributed $S = 0$. Only in case of a tie, both may claim $S = 0.5$.

[1]It has been chosen by the inventor of the ELO scoring system, Mr. Arpad Elo, to allow the compatibility and comparability of ELO scores and the anciently used chess scoring system by Kenneth Harkness

One important take-away of this scoring algorithm is, that it allows for an infinite number of pairings to be evaluated at the same time for an arbitrary amount of time. Contrary to the other algorithms that have been provided so far, ELO score does not necessitate the employment of a round-based system as two random solutions can be matched against each other without interfering with other concurrently running matches.

However, by using the described scoring mechanism, a definitive answer of how well deserved the score for a certain solution is cannot be given. At a certain time of observation - in our case: the moment the winner is to be announced - a weak solution might have just won by chance against a strong solution and thus be at the top of the score. Seeing that the voting which solution is better (read as: which solution is winning) is not only based on facts but also on individual preferences, a certain amount of randomness has to be assumed.

3.5.2 TrueSkill

TrueSkill is based on the ELO scoring mechanism, but additionally introduces a variance feature to judge the robustness of a solution's score. The latter has been introduced by Microsoft for their XBOX Live gaming network. While a low variance signifies that the current score is most likely not a pure result of chance but rather that it was earned with a number of consecutive wins, a high variance signifies exactly the opposite.

Figure 1.4 illustrates the TrueSkill system as used by Microsoft: μ signifies the average skill (or score) of a player (25 in this example), while the *degree of uncertainty* (or variance) is signified by σ. Accordingly the skill of the player in Figure 1.4 should be $25 \pm \sigma = [22; 27]$. However Microsoft is using the shaded area of the graph to predict (in this example) a skill of 15 to 20 - well below the average skill less the uncertainty σ. This is due to the fact, that Microsoft is ranking its players lower than they actually are - a precaution that we can ignore for our project. At this point, it is not known why Microsoft opted to display such low rankings.

Furthermore, the variance is also used in the calculation of the score points that are awarded or deducted after a win/lose. This way, the impact on a strong player losing once against a weaker player is not ruining his score. Would that strong player however lose more often, not

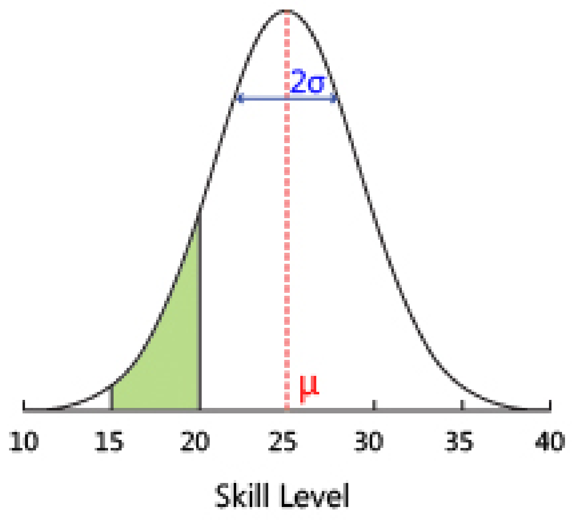


Figure 3.4: Belief curve of the TrueSkill ranking system[13][14]

only his score would be corrected downwards but his variance would get corrected upwards as well.

As TrueSkill is a proprietary algorithm, there are no publicly accessible resources that describe the exact math behind the algorithm. However, there have been some publications by Microsoft Research that offer a general overview as well as a multitude of graphs that allow for the reconstruction of the algorithm. Furthermore, as this bachelor thesis is discussing informatics rather than mathematics, no further evaluation of the math involved in TrueSkill will be done. Interested parties are invited to consult Herbrich et al. [9] for an academic publication or Minka [14] for a more accessible publication on the subject. For the means of this work, the knowledge that TrueSkill works in a manner similar to ELO Score shall suffice.

It also should be noted that an open-source library by the same name[2] will be used for this bachelor thesis "as provided". No further attempts of verification of the math involved in TrueSkill in general or employed by said library in particular has been made.

In conclusion, one can assume that TrueSkill would make another good candidate to rate solutions obtained by student input, as it incorporates all the features of the above mentioned ELO Score, as well as

[2]https://github.com/freethenation/node-trueskill

improves on it by providing a certain robustness against random results that are purely based on chance.

3.6 Conclusion

All algorithms that have been presented in this chapter are likely candidates for evaluating large numbers of solutions. However, once the algorithms are compared by their qualities, this is no longer the case.

Algorithm	Concurrency	Staggered	Required rounds
Round robin tournament	little	yes	44.850
Double Round robin tournament	little	yes	89.700
Single-elimination tournament	little	yes	299
Double-eliminiation tournament	little	yes	598
Pairwise Matching[3]	unlimited	no	none

Table 3.1: Comparision of voting algorithms for 300 participating solutions

Of course the statement that no rounds are required for ELO Score and TrueSkill are to be taken with a grain of salt: if there are no comparisions at all, the resulting order would be somewhat random as all solutions still have the same starting score. But it does however signify, that contrary to the other algorithms, there is no mininum amount of comparisions to generate a list: missing just one single round in the other four algorithms inhibits the creation of the resulting list.

Looking at the requirement of the algorithms to be executed in a staggered way, also eliminates the non-pairwise comparision algorithms. Given the limited amount of time during a lecture, it is an absolute requirement to be able to start and stop the voting-phase at any point in time.

Lastly, having only a limited amount of concurrency during the voting-rounds leaves a number of students without any actual voting to do while others have to vote right from the start. Providing unlimited concurrency all students can participate right from the beginning.

[3]ELO Score and TrueSkill are Pairwise Matching-algorithms and thus represented by this line

In conclusion, only ELO Score and True Skill have shown after the preliminary study that they are fit candidates for the implementation during this thesis. As both algorithms can be run concurrently on the same data sets, collecting the resulting data out of both algorithms and comparing the results should offer an interesting insight: should the list of the best/-most popular solutions be the same, one can assume that chance (in regards to just having won an important tournament just before the closing of the voting phase) has no important influence. On the other hand, there is still the claim by Hacker and von Ahn [8] that TrueSkill only produced accurate results after a multitude of tournaments had been completed - their paper found it to be a few thousands. It shall be seen, if this applies also to the data sets produced by students in our classroom-setting.

Chapter 4

Implementation

This chapter is focused at the details of implementing the described interactive unit into the *Backstage* system. For an overview of the architecture of the system, a more detailed description of the structure of courses and units, and how interactivity is being added, please refer to the appendix.

4.1 Phases of the voting interaction

As mentioned earlier, the interaction is separated into multiple distinct phases, which shall be discussed in this chapter.

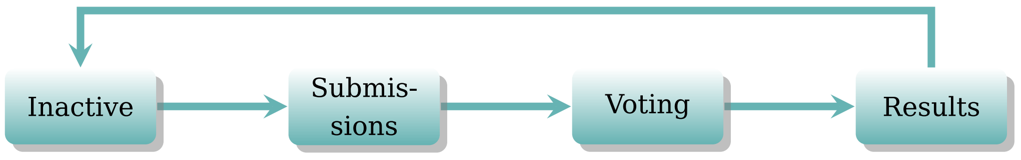
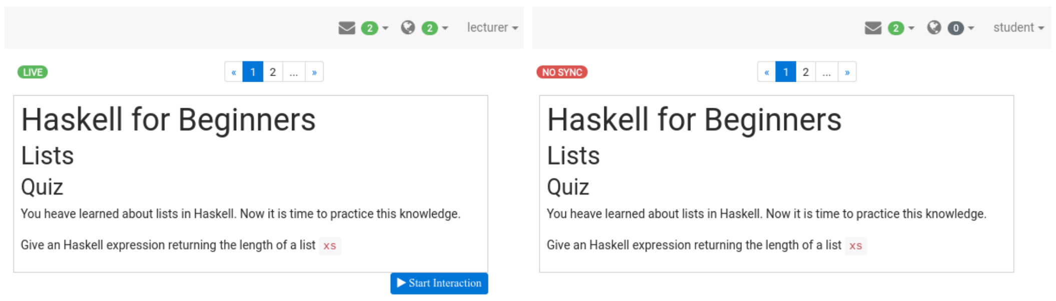


Figure 4.1: Overview of the different phases of the voting interaction

4.1.1 Inactive

If an interaction has never been run, its state is considered "inactive". In this state, students have no possibility for interaction with the current slide which has the interaction attached to. As the interaction itself is



Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.
Give an Haskell expression returning the length of a list `xs`

Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.
Give an Haskell expression returning the length of a list `xs`

Figure 4.2: Screenshot of lecturer and participant view of the unstarted interaction

independent of the context, it does not carry a question that needs to be answered by itself - the instructions and/or question must be part of the unit that is displayed at this point.

If however one or more runs have already taken place, the results of previous executions can be viewed.

In either case, the lecturer is provided with a button to start the interaction. Upon clicking said button, a broadcast is emitted to the connected clients.

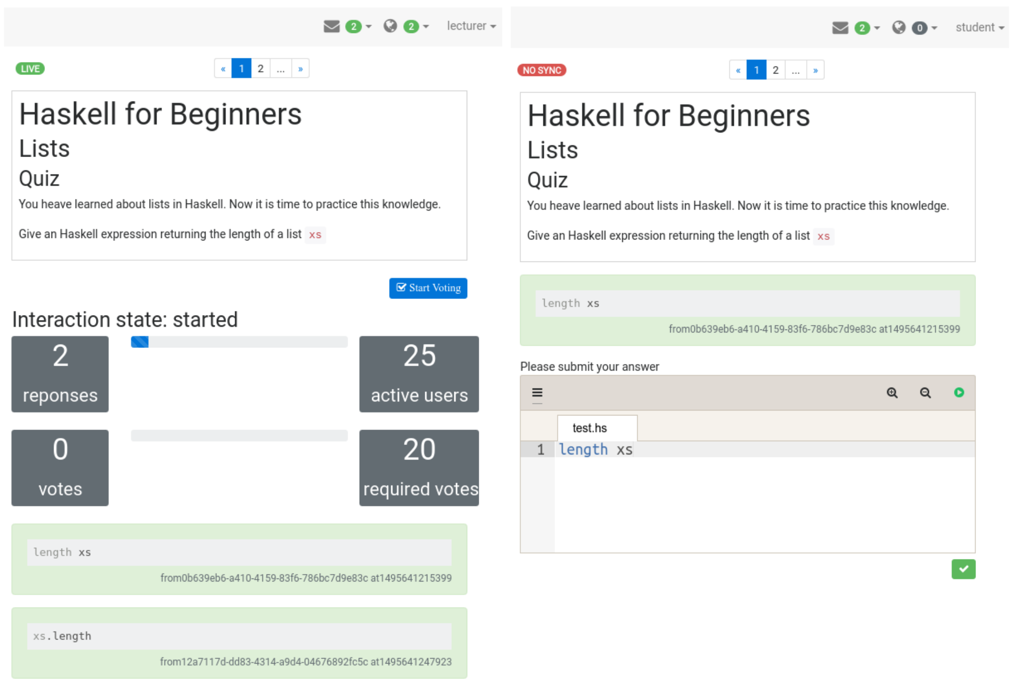
4.1.2 Submission

Upon receiving the broadcast, the interaction part of the application is displayed instead of - if applicable - the previous results of executions that have already taken place.

Participating students are provided with means to submit their response: depending on the configuration of the interaction, this might be a simple input field, a textarea, a syntax-highlighting editor or even a component like a graphical *Blockly* editor[1].

Students may now prepare their responses and submit them. As this interaction is completely modular, submitted answers may even be validated before allowing it to enter the pool of submitted answers.

[1] "The Blockly library adds an editor to your app that represents coding concepts as interlocking blocks. It outputs syntactically correct code in the language of your choice." [2]



Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.
Give an Haskell expression returning the length of a list `xs`

Interaction state: started

2 reponses	25 active users
0 votes	20 required votes

```
length xs
```
from0b639eb6-a410-4159-83f6-786bc7d9e83c at1495641215399

```
xs.length
```
from12a7117d-dd83-4314-a9d4-04676892fc5c at1495641247923

Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.
Give an Haskell expression returning the length of a list `xs`

```
length xs
```
from0b639eb6-a410-4159-83f6-786bc7d9e83c at1495641215399

Please submit your answer

```
length xs
```

Figure 4.3: lecturer and participant view of a quiz evaluation in submission state

Once an answer has been submitted, that very same submission will be displayed to the student as a visual indicator that it has been successfully received.

Depending on the concept of the lecture, the lecturer may opt to allow students to either edit and thus overwriting their own previously submitted solution or to collect more than one submission per student. In the latter case, each submission of the student equals a new, additional entry that will be subjected to the voting mechanism later on. Depending on the chosen input method, solutions may be validated for correctness before being accepted.

Upon receiving and validating students' responses, the server will also add a few selected attributes with a default value to every submission in order to prepare for the upcoming voting. These additional attributes are:

- *ELO score rating*: 25
- *TrueSkill rating*: 25

- *TrueSkill variance*: 8.33 and
- *Rounds*: 0

These attributes are used to store and bootstrap the values for the ELO score and the TrueSkill score, that are calculated during the voting phase. The rounds field is a simple counter, which reflects the number of votes a submission has received. The starting values of 25 and 8.33 are chosen according to the starting point of the Gaussian distribution used by the *True Skill* algorithm [11].

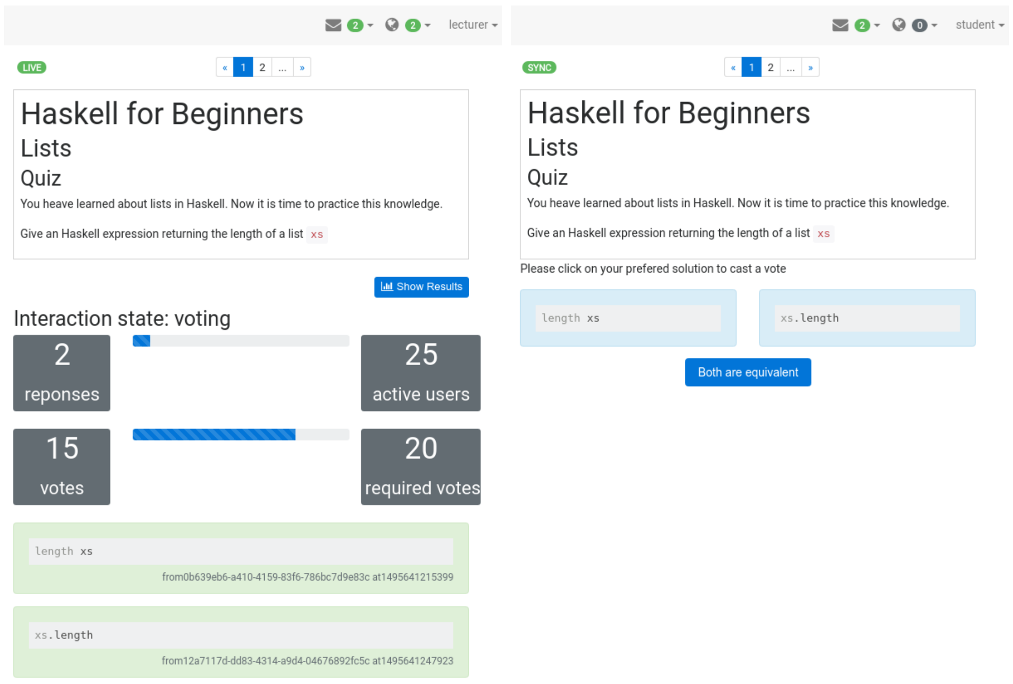
Lecturers are not provided with a possibility for input - however they are provided with a chronologicaly ordered list of all submitted responses of the participating students. They are given the power to end the submission process and start the voting.

Alternative views for the lecturer may include statistical data to facilitate his decision when to initiale the next stage of the voting process. Such data may include the number of submitted solutions, the number of students participating in this lecture, and the amount of submission per minute.

4.1.3 Voting

The voting process takes place in a pairwise comparison-like process. To achieve this, each participant is presented with two randomly sampled submissions from the database, without regards on the current standing of each submission or the author. By clicking onto one of the presented submissions, the student is expressing his personal preference for the selected one. In case the selection of a personal favourite is not possible - regardless if this is based on the believe that both submissions are equally good or that they are actually identical - the participant may press a third button underneath the solutions which flags both submissions as alike.

In technical terms, regarding the ELO score and TrueSkill rating, each click will cause the server to update the above mentioned ratings for each one of the two participating solutions: the clicked one is the winner in the calculation of the new rating, while the other one is the loser. Should the solutions however be flagged as equal, a tie is recorded.



Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.

Give an Haskell expression returning the length of a list `xs`

Interaction state: voting

2 reponses

25 active users

15 votes

20 required votes

```
length xs
```
from0b639eb6-a410-4159-83f6-786bc7d9e83c at1495641215399

```
xs.length
```
from12a7117d-dd83-4314-a9d4-04676892fc5c at1495641247923

Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.

Give an Haskell expression returning the length of a list `xs`

Please click on your prefered solution to cast a vote

```
length xs
```

```
xs.length
```

Figure 4.4: Screenshot of lecturer and participant view of the voting interaction

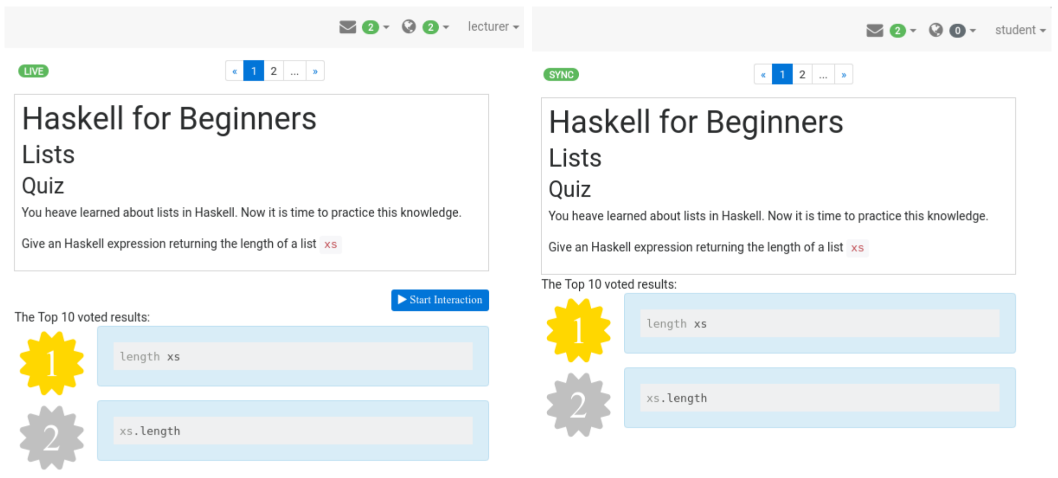
With each click, the student is presented a new set of two randomly sampled submissions from the database. This solution has been conciously taken over the possibility to have the “winning” solution stay in place and only replace the “losing”, other one. The reasoning behind this is to encourage an even inclusion of all submissions across the voting-process. Having the winning solution stand in place would dramatically increase its *rounds* number.

The lecturer is not provided with a possibility to participate in the voting. However, he is presented with an updated list of solutions and their standing. As before, he is provided with a button to conclude the voting phase of the interaction and proceed to the display of the results.

Additional visualisations for the lecturer may also include, as before, statistical counters like the amount of submitted votes, a quote regarding the submitted votes and the submitted responses and the amount of votes per 10 seconds. If desired, the lecturer may even be provided with a list of preliminary results, that is updated in near real-time.

4.1.4 Results

The display of the results is the stage, when the interaction has finished. As such, it is in fact exactly the same as the first stage of the interaction, *Inactive*.



Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.
Give an Haskell expression returning the length of a list `xs`

The Top 10 voted results:
1 `length xs`
2 `xs.length`

Haskell for Beginners
Lists
Quiz
You heave learned about lists in Haskell. Now it is time to practice this knowledge.
Give an Haskell expression returning the length of a list `xs`

The Top 10 voted results:
1 `length xs`
2 `xs.length`

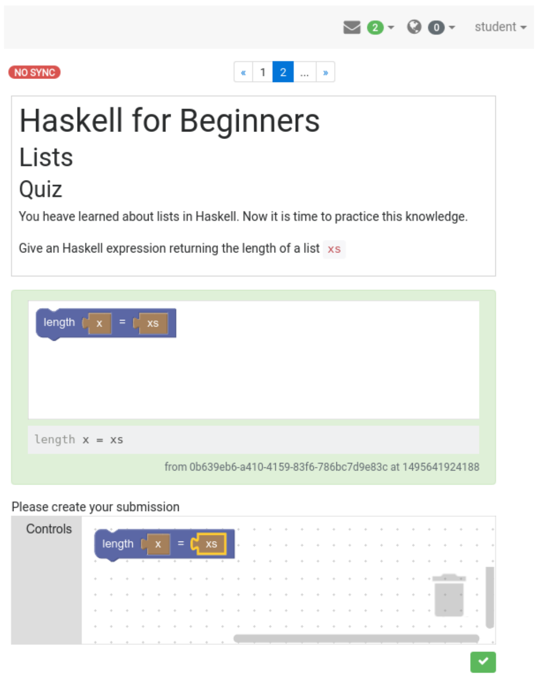
Figure 4.5: Screenshot of lecturer and participant view of the interaction's results

Displaying the final standing of the votes in the current interaction is done automatically, based on the algorithm defined in the interaction meta data (either ELO score or TrueSkill). The very same list is displayed to students and lecturer alike, barring the difference, that the lecturer may restart the interaction again.

In order not to overload the display, the lecturer may choose an arbitrary number of results to be displayed in the results screen. This number is also defined in the configuration data of the execution (see appendix).

4.2 Caveats

While the whole interaction has been designed to be as modular as possible, right now it cannot use each and every existing input method out of the box. This is due to the fact, that every input method has it's own data structure that needs to be taken care of - when saving the data as well as for displaying the data.


NO SYNC

« 1 2 ... »

Haskell for Beginners

Lists

Quiz

You heave learned about lists in Haskell. Now it is time to practice this knowledge.

Give an Haskell expression returning the length of a list `xs`

```
length x = xs
```

from 0b639eb6-a410-4159-83f6-786bc7d9e83c at 1495641924188

Please create your submission

Figure 4.6: Example of a voting interaction with the Blockly editor

As such, some amount of work is required to adapt this interaction to new input methods. For this thesis, the syntax-highlighting editor *Knoala* and the visual programming editor *Blockly* have been integrated. They share a majority of common code and are just separated by a few statements when it comes to rendering and processing data.

Chapter 5

Conclusion

This Bachelor's thesis explores the possibility of creating and integrating a new kind of interactive unit into *Backstage 2.0*, a digital backchannel for large scale lectures. This new interactive unit enables participants to submit self generated content and to vote on it collectively, thus allowing to rank the submissions by popularity.

Upon exploration of various possible tournament modes, it was made apparent that well known tournament modes, as used in sports, are not appropriate for ranking large numbers of responses. For this reason, the rating of participants' submissions has been selected to be done through pairwise comparison - a mechanism that has already been proved to be effective (compare [8]). More specifically, the ELO Score and TrueSkill rating have been adopted for this project.

In order to allow for a flexible system, the interaction has been designed to be universal and allow the processing of arbitrary input. Like this, anything can be made the subject of a voting-interaction: simple text inputs, visual programming language samples (like *Blockly*) or even pictures.

Future work on this subject can include a multitude of new approaches: while this approach was angled towards an anonymous voting and ranking interaction, it could easily be modified to present each participant with a single submission and ask them for a direct feedback, which - in turn - can be provided to the original submitter. Another interesting angle might be the usage of the voting results to create personalized exercises for every participant.

As of now, the flow of the interaction is strictly following the state-machine: Every execution of the interaction uses each stage exactly once. For future research it might be of interesst to leave this path and allow the lecturer to freely select the state of the interaction. Assuming a hypothetical scenario in which the students are not agreeing with the final results of a voting - either because of lack of participation or because of a lack of understanding of the underlying concept of the question asked - the lecturer may offer additional voting time or even offer to submit additional solutions to the problem.

In an academic context, the gathered data may also be used for statistical purposes. Knowing which student submitted which solutions and knowing how well they were regarded after voting on them, this data could be used for predictions of the general understanding of the lecutre material by the student and his potential success in the upcoming examination period. While this might be used to reassure students with a positive track record that they are well prepared for their exams, it might also serve as a motivational tool for other students that will probably not do so well. By tying the individual voting rounds to the subject areas of the lecture, it should even be possible to provide personalized guidance to the students, which area of the subject needs extra attention and training.

Going further, this system may also be used outside of the classroom to allow for a multitude of other, non-academic usages. In current times where so called *fake news* are gaining more and more traction, the system could be used to allow people to vote on contradicting news items.

Either way, pairwise comparision has proved to be a valuable measure with numerous usages in and outside the classroom.

Chapter 6

Appendix

All concepts explained herein were mediated by Sebastian Mader through personal communcation.

6.1 Architecture

While the original *Backstage* started out with the backend written in *Grails*, it changed soon to the *Play!* framework [17]. While *Grails* employs *Groovy* as their language of choice (which in turn is a derivative of *Java*), the *Play!* framework uses *Java* and *Scala* as its programming language. As those languages share a common denominator, the switch from *Grails* to *Play!* was not an issue. For storage, *MongoDB*, a NoSQL database, was used.

As *Backstage 2.0* is a completly new project with no ties to the original application, the chosen software stack is considerably different. This, however, only applies to the technical aspects of *Backstage* and not the teaching concepts.

Seeing that the focus of Backstage is - to a certain extent - synchronised interaction with multiple participants, a lightweight and easily synchronisable framework had to chosen. To combat the unnecessary transfer of large amount of data and the need to precompile content that is to be displayed, a concious choice was made to provide a web application, that runs in the browser and is only fed with a lightweight stream of data. This permits also to keep the server load low, as only JSON fragments have to be computed and sent to the clients.

The frontend framework of choice is *React*, a JacaScript library that has been mainly developed and published by Facebook (published in 2013). As part of the *dogfeeding*[1] process that those two companies adhere to, *React* is heavily used on their own websites and optimized for high traffic loads. As *React* is a JavaScript library for creating user interfaces, the resulting product is a single page web application.

Using *React* provides however a big advantage: Using *NPM*, the *Node Package Manager*, one can choose from an abundance of prepackaged JavaScript libraries, that can easily be included into one's project.

The frontend however, does not directly communicate with the database: it is doing this trough the exchange of *JSON* messages with a server, also written in JavaScript. These messages are passed back and forth using *WebSockets*.

All server client communication is handled by *socket.io* - an event-driven, bi-directional API, written in JavaScript. As it features the same syntax for server- and client-side operations, it is easy to integrate in all parts of this project.

Looking at the used storage backend, the database, we have stayed with the conecept of a NoSQL database. However, *RethinkDB* has been elected to replace the former *MongoDB*. It is a document oriented database and also stores all received data internally as *JSON* documents.

6.2 Flow

As mentioned before, *Backstage 2.0* is a *single page web application*, featuring data exchange with the backend server using *JSON* messages. In oder to keep multiple participants synchronised or to signal an event to a group of users, it is not necessary to keep track of every individual user's session. By joining an activity in a lecture, like for example clicking the square button of the "Reading Assignment 01" in Figure A.1, a client is joining a stream. Events will be dispatched from the server and are

[1] *Dogfeeding* is "an expression that comes from the idea that companies should eat their own dog food, or use their own products." [22]

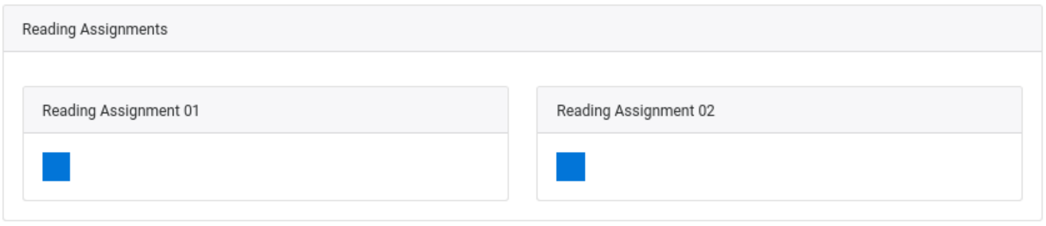


Figure 6.1: Example of a stream selection

addressed to a stream. This addressing will resolve automatically into the relevant clients and send them the event. Such events may range from simple events like turning to the next or a specified slide, starting interactions, or creating annotations or comments.

6.3 Courses and units

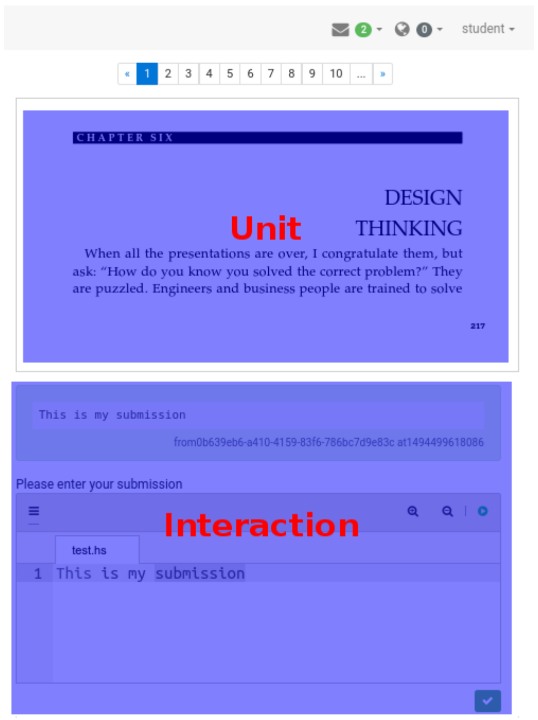
The main containers in *Backstage* are courses: Those represent a collection of different course materials which are provided to the students - either all at once or in smaller increments.

The provided course materials inside a single course are called units. Those units can take different forms like images, texts with markup, PDF documents that have specifically been converted for display in the application, or a compound consisting of a set of units in form of a graph. A proper analogy would be, that the units represent single slides in a presentation. The presentation - the collection of different slides/units - by itself is considered to be a stream.

6.4 Interactions

Interactions can be understood as an optional addon to a unit to which it can be attached.

This separation of interactions and units permits for greater flexibility: In theory, a single interaction may be shared and used across multiple units without the need for duplication - thus keeping the database clean of redundant content and improving user experience. For example,



CHAPTER SIX

DESIGN
THINKING

When all the presentations are over, I congratulate them, but ask: "How do you know you solved the correct problem?" They are puzzled. Engineers and business people are trained to solve

217

```
This is my submission
```

from0b639eb6-a410-4159-83f6-786bc7d9e83c at1494499618086

Please enter your submission

```
This is my submission
```

Figure 6.2: Layout of a unit with an optional interaction

the same interactions can be reused during a lecture to verify the understanding of a presented concept and again towards the end of the course for exam training purposes.

In order to facilitate such reruns of an interaction, the interaction model stores a list of foreign keys linking to executions. Every linked execution in return stores the parameters of a run of the interaction including the answers provided by the participants.

```json
{
  "creatorId":  "d0bc533d-f0dc-4401-b5fa-00646422af2a" ,
  "data": {
    "algo":  "ELOrating" ,
    "editor":  "knoala" ,
    "limit": 10
  } ,
  "description":  "Description for votingInteraction1" ,
  "executionIds": [
    "a0127b2e-d278-4979-b354-c495bdb462e4" ,
    "7a65b0b9-47b8-41af-97a7-62b61a7ea18a"
  ] ,
  "id":  "7a7491e4-8d9c-4a69-a493-5ed32c435fb8" ,
  "settings": { } ,
  "tags": [ ],
  "title":  "votingInteraction1" ,
  "type":  "voting"
}
```

Listing 6.1: Definition of an interaction with executionIds

```json
{
  "creatorId":  "d0bc533d-f0dc-4401-b5fa-00646422af2a" ,
  "data": {
    "algo":  "ELOrating" ,
    "editor":  "knoala" ,
    "limit": 10
  } ,
  "id":  "a0127b2e-d278-4979-b354-c495bdb462e4" ,
  "responses": [
    {
      "ELOrating": 25 ,
      "TSrating": 25 ,
      "TSvariance": 8.333 ,
      "createdAt": 1493393448063 ,
      "files": [
        {
          "name":  "test.hs" ,
          "value":  "This is a solution"
        }
      ] ,
      "id":  "147ea2aa-ecf5-437b-8f72-466789c731e0" ,
      "rounds": 0 ,
      "status":  "success" ,
      "userId":  "0b639eb6-a410-4159-83f6-786bc7d9e83c"
      } ,
      {
        "ELOrating": 25 ,
        "TSrating": 25 ,
        "TSvariance": 8.333 ,
        "createdAt": 1493393450840 ,
        "files": [
          {
            "name":  "test.hs" ,
            "value":  "This is another solution"
          }
        ] ,
        "id":  "06d8e1b5-4d3e-48fb-bf50-a3defe6621cb" ,
        "rounds": 0 ,
        "status":  "success" ,
        "userId":  "0b639eb6-a410-4159-83f6-786bc7d9e83c"
      } ,
    } ,
  ] ,
  "state":  "finished" ,
  "streamId":  "f84caf78-1c28-4272-82fe-23813709b140" ,
  "type":  "voting" ,
  "unitId":  "40f8d7c5-0bb4-4ec0-9be3-2425ade039b8" ,
  "versionId":  "04565996-9e00-4494-a5e2-42cd4d646eba"
}
```

Listing 6.2: Definition of an execution as definied in Listing 4.1

List of Figures

List of Tables

List of Listings

Bibliography

[1] Wikimedia Commons / User Aerion. Example of a single-elimination bracket, 2007. URL `https://commons.wikimedia.org/wiki/File:Mitbat-2007-bracket-large.png`.

[2] Google Developers. Blockly, 2017. URL `https://developers.google.com/blockly/`.

[3] World Chess Federation. Carlsen, Magnus NOR FIDE Top Chess Player, 2017. URL `http://ratings.fide.com/top_files.phtml?id=1503014`.

[4] H. Fend. *Geschichte Des Bildungswesens: Der Sonderweg Im Europäischen Kulturraum*. VS Verlag für Sozialwissenschaften, 2007. ISBN 9783531900476. URL `https://books.google.de/books?id=AA6ByCVrOxMC`.

[5] Dr. Jim Fox. Details and Comments Regarding the Elo System, 9 2009. URL `https://www.chess.com/blog/JollyPlayer/details-and-comments-regarding-the-elo-system`.

[6] HG Frost. Observations on a great occasion. *Adult Education*, 37 (5):283, 1965.

[7] Travis Goodspeed. Reversing an RF Clicker, 7 2010. URL `http://travisgoodspeed.blogspot.de/2010/07/reversing-rf-clicker.html`.

[8] Severin Hacker and Luis von Ahn. Matchin: eliciting user preferences with an online game. In Jr. et al. [10], pages 1207–1216. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518882. URL `http://doi.acm.org/10.1145/1518701.1518882`.

[9] Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill$^{\text{TM}}$: A

Bayesian Skill Rating System. In Schölkopf et al. [19], pages 569–576. ISBN 0-262-19568-2. URL `http://papers.nips.cc/paper/3079-trueskilltm-a-bayesian-skill-rating-system`.

[10] Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg, editors. *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, 2009. ACM. ISBN 978-1-60558-246-7.

[11] Heungsub Lee. TrueSkill - Rating, the modell for skill, 03 2013. URL `http://trueskill.org/#rating-the-model-for-skill`.

[12] Margie Martyn. Clickers in the classroom: An active learning approach. *Educause quarterly*, 30(2):71, 2007.

[13] Microsoft Research / Tom Minka. 7belief curve of the trueskill ranking system.

[14] Microsoft Research / Tom Minka. TrueSkill Ranking System, 2016. URL `https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/`.

[15] Wikimedia Commons / User Miraceti. Example of a 16-entry double elimination bracket, 2004. URL `https://commons.wikimedia.org/wiki/File:NSB-doubleelim-draw-2004.png`.

[16] Erik Paulsen, Edvin Bru, and Terje A. Murberg. Passive Students in Junior High School: The Associations with Shyness, Perceived Competence and Social Support. *Social Psychology of Education*, 9(1): 67–81, 2006. ISSN 1573-1928. doi: 10.1007/s11218-005-1365-y. URL `http://dx.doi.org/10.1007/s11218-005-1365-y`.

[17] Alexander Pohl. *Doctoral Thesis, Institute for Informatics, Fostering awareness and collaboration in large-class lectures. Principles and evaluation of the Backchannel Backstage.* Universitätsbibliothek der Ludwig-Maximilians-Universität, 2015. URL `https://edoc.ub.uni-muenchen.de/19012/`.

[18] Alexander Pohl, Francois Bry, and Yingding Wang. About Backstage. URL `http://backstage.pms.ifi.lmu.de`. Accessed: 28.10.2016.

[19] Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors. *Advances in Neural Information Processing Sys-*

tems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006, 2007. MIT Press. ISBN 0-262-19568-2. URL `http://papers.nips.cc/book/advances-in-neural-information-processing-systems-19-2006`.

[20] Silke Schworm and Frank Fischer. *Academic Help Seeking*, pages 282–297. Göttingen u.a., 2006. ISBN 3-8017-1813-1978-3-8017-1813-8.

[21] Bayerische Staatskanzlei. Art. 3 - Freiheit von Kunst und Wissenschaft, Forschung, Lehre und Studium. *Bayerisches Hochschulgesetz (BayHSchG)*, 05 2006. URL `http://www.gesetze-bayern.de/Content/Document/BayHSchG-3`.

[22] The New York Times Staff. Google Appears Closer to Releasing Its Own Phone. *The New York Times*, 12 2009. URL `https://bits.blogs.nytimes.com/2009/12/12/google-appears-closer-to-releasing-its-own-phone/`.

[23] April R. Trees and Michele H. Jackson. The learning environment in clicker classrooms: student processes of learning and involvement in large university level courses using student response systems. *Learning, Media and Technology*, 32(1):21–40, 2007. doi: 10.1080/17439880601141179. URL `http://dx.doi.org/10.1080/17439880601141179`.

[24] LLC Turning Technologies. ResponseCard RF LCD. URL `https://www.turningtechnologies.com/response-solutions/responsecard-rf-lcd`.

[25] McGill University. Return your clickers starting September 14, 2016, 9 2016. URL `http://www.mcgill.ca/polling/channels/news/return-your-clickers-starting-september-14-2016-262580`.