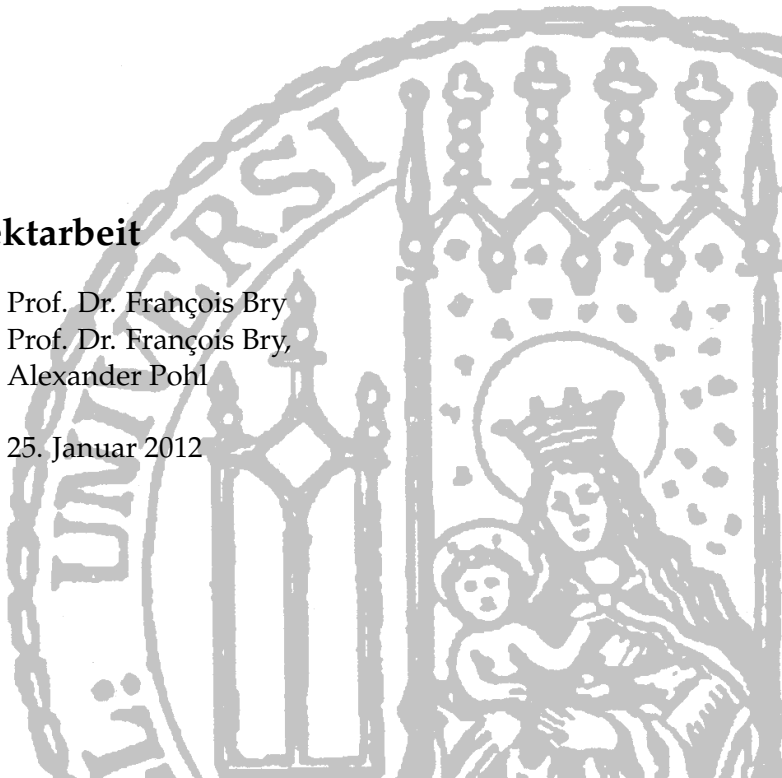
INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

PROCESSING RELEVANCE FEEDBACK IN BACKSTAGE

Julia Hadersberger



Projektarbeit

Aufgabensteller	Prof. Dr. François Bry
Betreuer	Prof. Dr. François Bry, Alexander Pohl
Abgabe am	25. Januar 2012

Acknowledgements

First and foremost, I would like to thank Prof. Dr. François Bry for offering this thesis and the advice he provided.
Besides, I would like to thank Alexander Pohl for his guidance during the development of my work and especially for his patience with me whenever I got frustrated. He was always a good listener and answered my questions promptly.
I want to express my eternal gratitude to my parents for their support and love.
Finally, I would like to thank Johann Kratzer, my eternal love and best friend. Completing this work would have been more difficult without his support and humour.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 25. Januar 2012 Julia Hadersberger

Abstract

The increasing number of students in one lecture does not only lead to overcrowded classrooms, but also to a unidirectional communication within it. For different reasons students often cannot ask a question about or comment on the lecture topic. Therefore, BACKSTAGE, a microblogging-based backchannel, was created that gives students the opportunity to post and answer to questions and comments. Furthermore, by reading the messages, the lecturer gets an impression of the speed and understandability of his lecture and receives feedback, which could help to improve his teaching. Due to the mass of messages in huge classes, it is not possible for the lecturer to read and answer all upcoming questions. Hence, some kind of filter is needed to reduce the number of shown messages and focus on the most important and popular ones.
Every participant is allowed to vote on those messages and thus collaborate with his fellow students to create a top-ranking, which is displayed separately in BACKSTAGE. Because the interests of the community will change over time, the top-ranking needs to make use of some aging process to include this alteration.
The purpose of this thesis is to give an overview of different rating, ranking and aging algorithms with their advantages and disadvantages. Furthermore, we want to develop our own prototypical algorithm for BACKSTAGE and discuss arising problems and ideas.

Zusammenfassung

Die steigende Anzahl der Studenten in einer Vorlesung führt nicht nur zu überfüllten Vorlesungssälen, sondern auch zu einer einseitigen Kommunikation innerhalb. Aus verschiedenen Gründen ist es Studenten oft nicht möglich Fragen zu stellen oder das Thema der Vorlesung zu kommentieren.
Daher wurde BACKSTAGE, ein Microblogging-basierter Backchannel, erstellt, das den Studenten die Möglichkeit bietet Kommentare und Fragen zu stellen und zu beantworten. Des Weiteren erhält der Dozent durch Lesen der Nachrichten einen Eindruck von der Geschwindigkeit und Verständlichkeit seiner Vorlesung, sowie Rückmeldungen, die helfen können sein Lehren zu verbessern. Aufgrund der Masse an Nachrichten in riesigen Klassen ist es dem Dozenten nicht möglich alle auftretenden Fragen zu lesen und zu beantworten. Deswegen wird eine Art Filter benötigt, um die Anzahl der angezeigten Nachrichten zu reduzieren und sich auf die wichtigsten und beliebtesten zu konzentrieren.
Jedem Teilnehmer ist es erlaubt für diese Nachrichten abzustimmen und somit mit seinen Kommilitonen zusammenzuarbeiten, um eine Spitzenrangliste zu erstellen, welche separat in BACKSTAGE angezeigt wird. Da sich die Interessen der Community im Laufe der Zeit verändern werden, muss die Spitzenrangliste Gebrauch von einem Alterungsprozess machen, um diesen Wandel zu erfassen.
Der Zweck dieser Arbeit ist es, einen Überblick über verschiedene Rating-, Ranking- und Alterungsalgorithmen mit ihren Vor- und Nachteilen zu geben. Des Weiteren möchten wir unseren eigenen prototypischen Algorithmus für BACKSTAGE entwickeln und die auftretenden Probleme und Ideen diskutieren.

Contents

CHAPTER 1

Introduction

Anyone who has graduated from university, has certainly experienced some lecture with a large or even an overcrowded classroom. In such cases students' interactive participation is inhibited and they take the more passive role of spectators. The reasons for this are often the fear of speaking in front of so many unknown people, of interrupting the lecture to ask questions and the so-called single-speaker paradigm of lectures [3] . Not only students suffer from less interaction, but also lecturers do seldom get feedback in form of questions or comments to see if the learning process works or if his students underachieve. The technical progress at universities makes it possible to use Web 2.0 applications, particularly social media, to overcome the communication gap between the lecturer and the audience. The advantages and disadvantages of Web 2.0 technologies in higher education were already part of earlier research work in which the conclusion was drawn, that universities should make use of it and research the tools with the best effect [15].
Therefore, the platform BACKSTAGE [5, 8, 14, 27], a microblog-based application, has been developed to give students and lecturers the opportunity to write and read short messages, for comments, questions and answers, during the proceeding lecture. Furthermore the lecture slides are simultaneously shown, quizzes can be conducted to see the status of learning progress and all events are saved to logs for future insight. BACKSTAGE aims at increasing the participation and awareness in class and motivating to reflective thinking and active knowledge construction.
It can be imagined that due to the great number of students even a mere chat communication is nearly unmanageable and that lecturers do not have enough time to read all messages. Although the quality of messages in average might be good, it can vary greatly [6]. Therefore, some kind of filter is needed to find the relevant messages, like interesting comments or important questions, so that the best of it can be shown to the lecturer and the audience. As BACKSTAGE is microblog-based and hence part of social media, it is obvious that ratings by users could provide the feedback to evaluate the content quality and detect the current topics and questions in the audience.

Because of their great dependency on context and purpose, one can find many rating and ranking algorithms in literature. In fact, those terms are often almost used the same way although, as we think, there is a salient difference: We understand rating to be a mean to assigning some value to an item to judge its importance. For example; a message could be rated "plus" or "minus" depending on if you like it or not. In such simplest form, with rating, it is voted for or against the relevance of an item. Ranking, in contrast, relates two or more items to each other by assigning a relative order to it, like item A is better than B and B is better than C. In its simplest form, the user expresses pairwise preference between any two items. Thus, while rating expresses quantitative difference between items, ranking is just relative and it cannot be seen wether item A might be ten times or just one point better than B. Those two terms are often mixed up because they influence each other in many cases. On the one hand it is possible to get a ranking by ordering items according to their rating values and on the other hand a ranking could imply rating values, for example the first item in the ranking gets the highest value.
In general, ratings and rankings are used to express quality of items and therefore they are used in a variety of fields of study, like economics, politics, mathematics, social studies, psychology and information sciences. In social media, for example, they are used to evaluate the relevance of news articles and searching results, the quality of a product, the trust of persons in a network or the expertise and knowledge of community members [7]. But they are also involved in university rankings, credit ratings or simple surveys.
In this thesis an overview of different ranking and rating possibilities will be given, but due to the wide scope of algorithms and alternatives there can be no claim of completeness. In fact ranking and rating are often inseparably interweaved with other fields of study and therefore some kind of completeness can never be reached. Instead of trying to create an impossible overall summary, here the most salient rating and ranking approaches, possibly of meaning for the project BACKSTAGE, will be presented.
Rating and ranking in BACKSTAGE serve various purposes. First of all it can be used to evaluate all messages and filter out irrelevant and point out relevant information, as a form of collaborative filtering. Second it can be feedback to each user who wrote a message to see if his comment or question is of importance. And third it can be feedback for the lecturer to recognize the deficits and interests of his students, which he can flexibly respond to during lecture.
It can be imagined, that the relevance of messages and their ranking does not only depend on the quality of the content, but also on its age. As the lecture proceeds, topics might change and some questions or comments might be mistimed and simply old. Therefore, some kind of aging is needed for BACKSTAGE to gradually reduce the importance of older and direct the attention to newer messages. The physical time is barely informative, because lectures can vary in progress. For example, introductory slides or simple images might be presented much quicker than a difficult mathematical formula. So aging should depend on something more individual, on a kind of perceived time, like the activity on the backchannel platform. When, why and how aging shall take place, gives rise to many questions that are dealt with in this thesis.

In this project thesis an overview of some literature about rating and ranking is given in Section 2. In Section 3, a closer look at the importance of aging is taken. Furthermore the requirements for BACKSTAGE are defined to design a prototypical algorithm afterwards. The results are discussed in Section 4, where ideas for future work and visions are also presented. Afterwards, in Section 5, conclusion is drawn.

CHAPTER 2

Rating and Ranking - A Selective Overview of the Literature

If you have a look at previous papers about rating and ranking, you realize that there are so many variants of rating in different fields of study, that there seems to be no universal solution for a problem and each project needs an individual and very specific design according to its own requirements.
As mentioned in the introduction, the terms rating and ranking are often used interchangeably and it can be very difficult to distinguish the two, especially when they are interleaved. Though, in our opinion a difference can be made: Rating terms the process of assigning some concrete value to a single item, while ranking relates two or more items to each other by specifying a relative order to it.
By rating a ranking can be determined, for example by arranging items according to their scores, for which those rating values have to be summed up in a way to get a single score. But it has to be noted, that the information about the distance between those items gets lost, for example how much better item A is compared to item B. Conversely, a rating can be determined by a ranking, if the items are assigned rating scores that preserve the ranking order. For example, the last item in the row gets the score 1, the second last 2 and so on. In this case of influencing each other the question appears, where the process of summing up or computing values to get a single score belongs to.
To get a better overview this section is divided into two parts. First of all the chapter "Rating" analyses the concrete process of voting for or against a message and the influence of different scales and systems on further calculation. The following chapter "Ranking" contains different ways to summarize those rating values and create an ordered list. Because of the abundance of different algorithms, this overview is restricted to those, which are relevant and interesting with regard to BACKSTAGE.

2.1 Rating

As mentioned above, rating is the process of assigning some value to a single item. It can be found in different situations and is done by diverse types of entities. An entity in this case could not only be an individual person, but also a company. For example, the simplest form of election is a situation, where different persons, thus entities, vote or rate someone out of a list. In contrast to some other rating situations voters are usually allowed to vote only for one item, which indirectly means all other items do not receive any votes of this person. Another example are financial service companies that issue credit ratings of corporations or even countries. In this case the entity represents a company and thus a few persons that evaluate the credit worthiness of someone and express the result as a single rating.
As it can be seen, rating is used for different purposes, therefore it is not surprising that context, purpose and users in the internet are also diversified. Rating can be found for example on commercial websites, where it is used to get and offer reviews of sellers or products. Even online communities and Web 2.0 applications are aware of the benefits of user feedback, which does not only give a hint where to find high-quality [6, 21], but also can regulate a whole system. Feedback can be used to not only avoid spam by weak or even negative votes, but also point out interesting content or report misuse by marking it [7].
A closer look at rating schemes and algorithms has to be taken, because rating should be used in BACKSTAGE to give users the chance to vote for posts to emphasise what is important to them. To enhance the comprehensibility the terms rating, rating scheme and rating system are defined in the following way: Rating names the combination of a specific *user* that rated one *item* with a specific *value*. The term rating scheme denotes, how the rating is designed for a specific system; for example which range is possible for a value - plus, minus, one to five stars, etc. - or if a user is allowed to evaluate every item and how often. In contrast to the design by a rating scheme, the rating system names the concrete implementation of a rating scheme. Basically rating schemes can be distinguished in two main groups, namely explicit and implicit rating.

2.1.1 Explicit Rating

Explicit rating denotes all algorithms that necessitate an intentional vote of a user, which means he is conscious of his evaluation [10, 20]. Popular examples of its use are recommendation systems on Netflix[1] or Amazon[2], where reviews of customers are used to suggest new products [20] or eBay[3], where users are able to rate a seller [18]. Another known system is Yahoo! Answers[4], which allows users not only to rate answers, but also vote for the best of it [1].
There are different ways to define the rating scheme or more precisley the rating values. To give users the possibility to express that they like a document or similar

[1] http:\\www.netflix.com
[2] http:\\www.amazon.com
[3] http:\\www.ebay.com
[4] http:\\answers.yahoo.com

by voting for it, is the simplest form of rating [20]. The computation with positive values is very easy and can be accomplished in no time, but in some cases the huge figures might be a problem for further calculation.
Other algorithms try to deal with this problem by computing normalized values, but this can be made a lot easier if users are only allowed to vote in a range from zero for dislike to one for like. The problem of distinguishing between a lack of information or really "negative" votes after computing for example the average value could be a great disadvantage of such a system in some cases [25].
A well known alternative, for example used at Amazon, is the five star rating scale, where users can choose between one to five stars. However, with such a rating scheme it can still not be distinguished between no or negative votes. It could be dealt with it by simply visualising how many votes came in, like it is done for example for Amazon reviews. It is noteworthy that, as a consequence of this solution, the responsibility to judge the better item lies to some degree with the user. If there is no further calculation, he has to judge on his own, if he prefers an item with a better overall rating but less votes or a medium overall rating with more votes.
A last one and often used form of explicit rating is the possibility for users to vote positively, neutral or negatively for something. This system is mainly known from eBay, where a seller can be rated after a transaction [18] and in different form from Yahoo! Answers, where you can not only vote positively and negatively for an answer, but also pick the best one or give stars for good questions [1]. There are many other projects where this kind of rating also takes place [7, 16]. This rating scheme seems to be better, because users can "punish" bad or spam messages by giving negative votes, which is easier to distinguish from no voting. But the possibility of negative votes complicates the computation and an element of risk remains that positive and negative votes might cancel each other out, which effects that no votes and balanced votes seem to be the same.
As it can be seen explicit rating can be an expression of trust, expertise or quality in general and forces the user to think actively about his judgement. But this could also be a disadvantage, as forming an opinion about an item can be seen as effort. If there is no kind of benefit or reward system, users might be uninterested in taking the time to explicitly rate something [24]. This could end up in some kind of spiral as users vote less, less feedback is gained, which leads to rare ratings and that causes dissatisfied users, who will not vote [10]. Furthermore using explicit rating gives also rise to intentional manipulation of the system [6], as it might be easier to see through the way the algorithm works.

2.1.2 Implicit Rating

In contrast to the explicit form, implicit rating extracts its information from different interactions or data, which is interpreted as a vote although the user might not be aware of it [10, 20]. The use of implicit rating can help to overcome the data sparsity, which is a big problem in the explicit rating.
The main difference between those rating algorithms is the source of the implicit vote, which obviously depends on the application in which rating is used. One basic version is simply counting the clicks on links, results or other objects, which

signal interest and therefore a positive feedback [1]. This can be applied to a variety of situations, for example the click on a link of a commercial website can signal interest in a product. Though, there could be also "misclicks" or even intentional "click spam", which influence the system the wrong way [6]. To eliminate such mistakes, it might be better to choose other more reliable clicks, like adding something to someone's favourites, printing or buying it. It is even possible, although quite complex to handle, to measure the time the user spends on a website or similar [24]. Giving an answer on a board or question-answer-portal can also be seen as a positive vote for the question, as it has to be good enough to attract the attention of the answering user, although some messages could also be seen as spam according to the above mentioned "misclicks". An imaginable example is an off-topic advertising link on a thread of a discussion board or entries that do no longer refer to the original topic. This could happen quite quickly, as human communication can be very excursive and one subject might lead to another during the discussion, which means that some of the following messages also do not handle the original topic.
Using the implicit form reduces the effort for the user to actively think about his judgement, but as he is unconscious of his decision, he also does not know, if he makes mistakes. Furthermore the disadvantage of probably needing more capacity to store and process all collected data is given. Advantageously implicit rating makes it, contrary to explicit rating, easier to hide parts of the system, so that the algorithm cannot be seen through that easily, as users often do not know which data takes influence.

2.2 Ranking

We define the term ranking as a relative evaluation to get an order of items. This order does not necessarily have to be, but also could be, strict, which means that there are no items with the same rank. Ranking can like rating be found in many situations. It is often used to show the best object, like university ranking, which is not strict, lists the most successfully working and best equipped universities in the world. Another example are music charts, where songs are ordered according to their success measured by their sales figures. But ranking is of course also used online, most known for ordering searching results according to their relevance [29] or for expert finding, which can be part of a discussion board or question-answer-portal, but also for listing the best players in a game [1].
As it can already be seen by the examples, there are different ways to get a ranking, although they all end up listing the best item first. But the term "best item" depends on the context; it could mean the item that fits best for one specific person, like it is done in recommender systems, or in a specific category, as for example music charts could be divided by their style. It could also depend on different variables, for example the university ranking is influenced by many different details of information.
In BACKSTAGE ranking shall be used to get the top-k most relevant posts, which are shown in a list to all participants, in particular the lecturer and the students. This can be seen as a special case of ranking, as the opinion of many students has

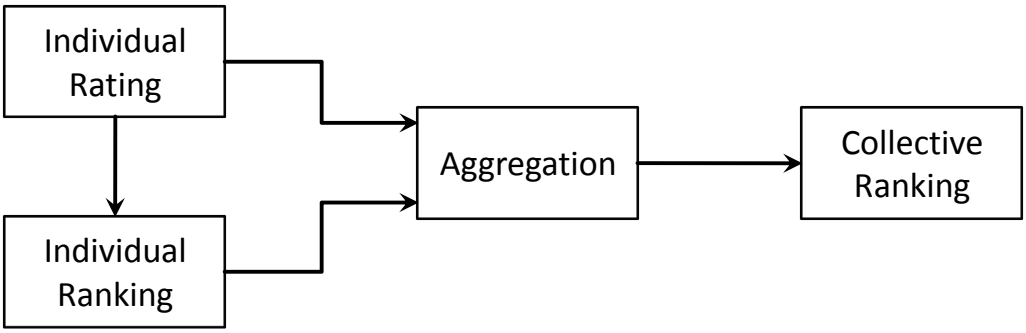


Figure 2.1: Basic ways to get a global ranking

to be collected and merged in a way to get one ranking. So it has to be noticed that a collective ranking requires some kind of aggregation. Keeping that in mind it can be thought of three basic ways to calculate a collective ranking, which can be seen in Figure 2.1. First, the users can individually rate different items and the ratings of all users are aggregated, so that each item has one collective rating value. The collective ranking is created by ordering the items to those values. Second, the individual rating can be used to create an individual ranking of all rated items for each user. Afterwards those individual rankings of all users are aggregated in a way they merge into one collective ranking. Third, the user can be asked to directly rank some or all items according to his preferences and afterwards the individual rankings of all users are aggregated to one collective ranking. The last mentioned way of calculating a collective ranking is a more complex and difficult challenge for users and it has to be considered that it might cost them much motivation.
To get an overview of the possibilities for ranking related to BACKSTAGE, the contemplated ranking algorithms are divided in three categories, namely simple mathematical solutions, eigenvector-based ranking and the special case of collaborative and content-based filtering.

2.2.1 Simple Mathematical Formulas

The most intuitive way to get a collective overall ranking is to use simple mathematical functions, which advantageously are often easy to use and cause only insignificant load on the system. Especially if the system is based on rating, mathematical solutions are not only an opportunity to directly calculate a ranking, but also for simple aggregation of the decisions of all users for any other ranking system. It is an interesting aspect that the solutions of this category can be further split into non-parametric and parametric formulas. Parametric algorithms in contrast to non-parametric algorithms make use of parameters that are externally set and therefore influence the computation significantly [2].

2.2.1.1 Non-Parametric Solutions

A non-parametric algorithm, thus a calculation without any external influence or set parameters, is desirable, because there are no unknown values that have to be set up in one or more experiments and the results of such a solution are predictable. As algorithms are even said to be parametric, just because there are weights used during the aggregation [2], simply summing up values, the exclusive use of the arithmetic mean or similar can be counted to non-parametric solutions. Because individual rating seems to be used more often as a basis than individual ranking, it makes sense to have a look at such ranking algorithms first.

As already suggested above in the context of explicit rating, simply summing up the single ratings for an item to get one aggregated rating for it, by which the items can be ranked, is a very simple solution that is easy to use [20, 21]. But there are different drawbacks, depending on which rating scheme is used: If the scheme is just designed to give positive votes or no votes, the figures could increase with every rating that comes in and therefore get very huge, maybe too huge for the implementation and further calculation.
On the other hand, if the scheme allows positive and negative ratings the problem of huge figures could still remain and even escalate on the negative side of the scale, but furthermore, as mentioned above, positive and negative ratings can cancel each other out [25]. If it is relevant for the context in which the rating scheme is used, this could be a real problem, as the ranking is created by strictly ordering to the summed up ratings. In the end no difference can be made between for example an item that received just a single rating and got the collective rating of maybe 1 and an item that already was rated many times, but the collective rating is at 1 for the moment, although it might already have been considerably in the negative or positive range of the scale. Another disadvantage of simply summing up ratings is the fact, that much feedback is necessary for an accurate ranking by unnormalized values [10].
A first wise solution to get rid of the huge figures is normalizing the values by calculating the arithmetic mean out of all ratings for one item and to rank all of them according to their average collective rating. This is a common solution that is used for example at Amazon to calculate the average contentedness of customers with one product or on eBay for the average reputation of a seller. Although the arithmetic mean can handle the problem of huge figures, it gives rise to another great disadvantage, as it is easier for new items to get a better collective rating than older ones. This is plain to see in the extreme example of item A, which is older and already had ten ratings, namely nine times five stars and one time four stars and of item B, which is new and had just one rating of five stars. Calculating the arithmetic mean for both items results in an average of 4,9 for item A and 5 for item B. If the average is not rounded up, item A will never be able to reach an average rating of 5 again, just because one person gave it only four stars. So it is easier for new items with less ratings to get a better average [21]. On the other hand it is also rather possible that new items get a worse collective rating than older ones, as their value relies on only one rating.

Depending on the context there are many other simple formulas possible to achieve an aggregation and hence a ranking. For example ranking users according to their expertise in a question-answer-portal can be easily handled by calculating the number of questions, which were answered by one user, believing that more answered questions relate directly to a higher expertise. But it might be better to count the number of askers one user helped, because it can be said, if he was possible to answer this question, his knowledge might be greater than that of the questioner. Seeing that a user helped ten persons is more meaningful than seeing that he has answered ten questions, maybe even from the same person. But this simple solution has also an evident drawback: It cannot be distinguished between a user who answered only simple questions of lay people and a real expert that answered the same number of questions of other experts with a high knowledge [30].

A collective ranking can also be calculated out of individual rankings, which a comparison-based method is often used for. Two or more items are shown to the user, who has to choose which one he prefers in the case of two items or even to order them, if there are more items. One solution to keep fairness, is comparing an item to another one as long as it wins and as long as there are other items it can be compared to. Out of this knockout tournament an individual ranking can be derived, but it is also possible to use this method irregularly and accordingly on demand, to refresh scores of items. Therefore not necessarily each user has to be asked to give a whole ranking, but can be asked to rate specific items that were chosen by any mechanism. Of course this mechanism does not have to be explicit, it can also be solved implicitly. For example if the user browses through a website and can choose between two links to maybe different topics his selection can be seen as a preference of this link in contrast to the other one. The advantage of such a ranking method is the less need for feedback to get an accurate overall ranking [10].

The so-called Hasse method is a further non-parametric ranking algorithm that can handle items with two or more categories and values by which they can be ordered. It is known for scientific rankings, for example of chemicals with different properties, like their flammability limit or similar. If every property of item A is better than that of B, it is clear that A has to be ranked higher than B. But as an item could be better in one, but worse in another property compared to a second item, it is hard to say in which way they should be ranked, so they are termed incomparable.

This comparison can be shown in a graph, the so-called Hasse diagram, where the vertices represent the items and an edge between two items their comparability to each other. The graph is created the way that higher ranked items are on top and lower ranked ones follow downwards, so that it is easier to see the partial order in it. On the one hand the advantage of the Hasse method is that different sources of information are not merged by aggregation, but on the other hand there is only a partial and not a total order of the items and no estimation can be made between two items that are not connected, which means they are incomparable.

Of course the Hasse diagram can be a basis to read off each possible total ranking for all items and afterwards it can be counted for each item how often it is placed

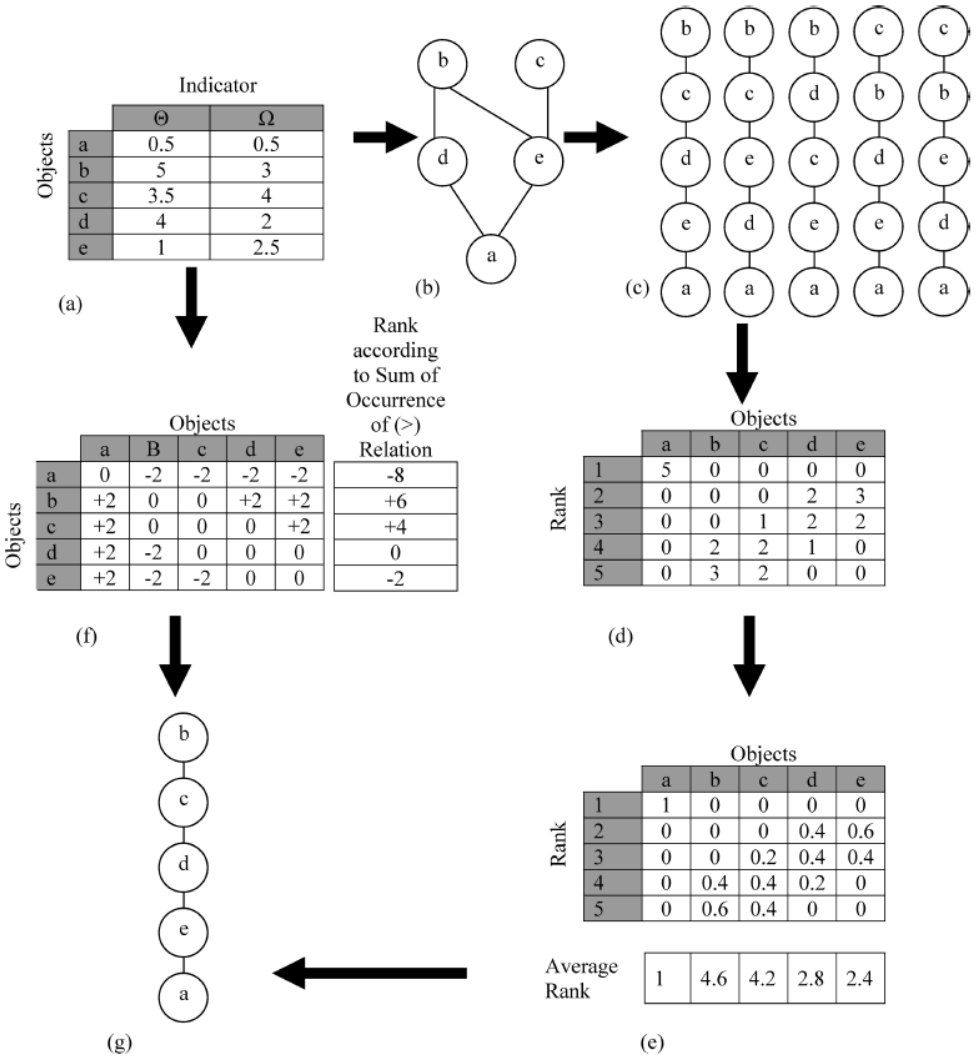


Figure 2.2: Example for ranking by the Copeland and Hasse method [2]

in which position in those rankings. It is remarkable that this can be seen as a rating once again, calculated out of different individual rankings. Those ratings are aggregated by calculating the arithmetic mean for each item and afterwards they can be ordered by those average ratings [2].
Another possibility for a non-parametric ranking is the Copeland method, where items are ranked according to their so-called Copeland score. This is the number of wins minus the number defeats of an item in pairwise comparison to all alternatives. Each time one item is better than another you add +1, each time it is worse -1 and 0 if they are incomparable. It could happen that two items are incomparable because the value by which they are compared is identical or, as mentioned in relation to the Hasse method, if they are compared by different properties and an item is better in one but worse in the other.
A square matrix with rows and columns presenting the items can show all comparisons at once and the Copeland score for each item can be read off by summing up each row. The ranking is created by ordering according to the Copeland score, reminding the lower it is, the more often the item lost, the lower in the ranking it has to be. Although this is an easy way to calculate a ranking it has to be noticed, that there is a remaining possibility of identical ranks, hence Copeland scores, and therefore no unique ranking can be guaranteed [2].
An example for the use of the Hasse and Copeland method can be seen in Figure 2.2 on page 12. The illustration (a) shows the data, hence five items a, b, c, d, e with their two properties. Out of the Hasse diagram in (b) five possible combinations are shown in (c). The counted positions in each possible ranking (d) are used to calculate the average in (e), which leads to an overall ranking, shown in (g). On the other side the data in (a) is used to build the comparison matrix and calculate the Copeland score in illustration (f), which leads to the same overall ranking in (g).

2.2.1.2 Parametric Solutions

As parametric ranking algorithm, each of the above mentioned non-parametric solutions in combination with weights or other factors is imaginable. This opens up many new ranking methods by for example just assigning weights to specific users, to limit or boost their influence. It is also possible that a part of an already created formula takes too much influence and therefore it is reduced by an external set factor. This can be very helpful, if it is proven in experiments that the results are like intended, but it also seems to be arbitrary, as there is often no other justification for setting this factor than that the authors wanted it that way. To give an impression what parametric solutions could look like some examples are presented in this chapter, but there are a lots other possibilites for combining factors, weights and different formulas.

The Backchan.nl project [16], as a parametric example that aggregates individual ratings to a collective rating by which the items are ranked, is very similar to BACKSTAGE and also based on public messages that can be voted for. The rating scheme consists of explicit up- and down-votes, where each user is treated equally. Part of the formula for the overall score of a post, by which they are ranked in the end, is

among other things a so-called vote factor of a post that is represented in Equation 2.1.

$$voteFactor(p) = \frac{kU_p}{U_p + D_p} + \frac{U_p + D_p}{\overline{U + D}} \tag{2.1}$$

U_p reprents all up- and D_p all down-votes for p, while $\overline{U + D}$ represents the arithmetic mean of all positive and negative votes during the whole session. The first fraction of the formula represents how well a post has been voted for and gets higher with more positive votes, while the second fraction appeals to how much in the whole context has been voted for this post, regardless of positive or negative. The second term is targeted on bringing up posts that are controversial and therefore interesting to read. The external set variable k is used to regulate the influence of the first term, hence how important positive votes are [16]. Although this formula seems to be very easy to understand and use, it is not evident why the factor k is necessary and which value it should have. But it is remarkable, that the above mentioned problem of positive and negative values cancelling each other out is countervailed by the second term, which grows with every vote for this post.

Another example of the same ranking category is the Real-Life-Rating algorithm, which includes the factor of friendship and expertise. Especially for Web 2.0 applications the influence of social networks is not negligible, in fact they can even push irrelevant content [20], and the opinion of friends and experts is particularly appreciated [1, 21, 23]. This elaborate algorithm takes care of many relevant aspects, but the more accurate an algorithm is, the more complex it gets. Therefore this chapter will not include each formula and aspect in detail, but just the most important and basic facts that are relevant with regard to BACKSTAGE.
The Real-Life-Rating algorithm is related to an application, where all users are allowed to publish resources as authors, which can be rated explicitly by other users. For this rating the expertise and friendship relation of the evaluating user to the author is taken into account. The received ratings for a resource affect in a further calculation the expertise level of the author for the corresponding domain. All new users, who have not published any resources yet, start with the minimum value of expertise and therefore do not have any advantage in contrast to older members. Furthermore, users are motivated to publish resources, to improve their expertise level and get more influence. The expertise level is separately treated for different domains, as being an expert for example for biology does not imply being an expert for computers. The system needs to provide the possibility to assign a resource to one or more domains.
The rating of a user is weighted by his expertise level for the according domains and his friendship weight to the author. The basic idea behind the friendship term is that friends evaluate subjectively and therefore better than other users, so the friendship weight is calculated by the intensity and distance of both users and the evaluator has more influence if he is seen as more objective.
A resource is ranked by the so-called Bayesian Weighted Rate Calculation, which is based on Bayesian Rating. This is method avoids the above mentioned problem that new items with a few excellent respectively few very negative votes are of-

ten higher respectively lower ranked than older items that have many really good votes. The trick is that the more ratings an item has, the more weight should be applied to them. If an item does not have enough ratings to be believable, it should be closer to the average rating of all items. The formula for Bayesian Rating can be found in Equation 2.2, where *avg_num_votes* stands for the average number of ratings of all items that have been rated yet, *avg_rating* for the average rating of all resources' ratings and *this_rating* and *this_num_votes* for the rating and number of ratings of the actual item. The important factor is the average number of votes, because the higher it is the more ratings are needed to influence the formula.

$$BR = \frac{avg\_num\_votes * avg\_rating + this\_num\_votes * this\_rating}{avg\_num\_votes + this\_num\_votes} \tag{2.2}$$

$$BWR(R_j) = \frac{avg\_num * neutral\_value + RN(R_j) * WR(R_j)}{avg\_num + RN(R_j)} \tag{2.3}$$

The final formula, the Bayesian Weighted Rate Calculation, shown in Equation 2.3, is the applied Bayesian Rating. The average rating is substituted by the *neutral_value*, which is for example in a one to five stars rating scheme the three. The number of ratings for a resource R_j is represented by $RN(R_j)$ and the term $WR(R_j)$ denotes the already weighted rating for it, the calculated average of all user ratings in combination with their user weights for this resource.
To close the circle, the new expertise level for the author in a specific domain is calculated by the average rating of all his resources in this domain [21]. This algorithm seems to be very elaborate and robust, but it is also very specific and requires different preconditions, like the possibility to measure friendship or the fact that this has to be even important for the further calculation.

The last example shows the combination of individual ratings and rankings that are aggregated all at once to get a collective rating and ranking. Because, as already mentioned above, the influence of social networks is very important, it also has to be considered that members who are more active in a community have a greater influence on people reading their messages. This varying influence was also considered in different formulas and for example used in a ranking algorithm for microblog search [23]. Therefore different properties - the so-called FollowerRank, LengthRank and URLRank - of an item are calculated and added together, to determine the best results for a specific query in a microblog.
The FollowerRank is defined by the indegree divided by the sum of indegree and outdegree of an author, where the indegree denotes the number of people who follow this author and the outdegree is the number of people this author is following. The indegree and outdegree of a user can easily be determined, if the system uses an interface that supports the observing of specific persons. The indegree by itself can be seen as an indicator for quality, assuming that users follow authors because of their good messages. Usually, however, active members do also have followers for reasons other than their messages' quality. To account for this, the indegree is divided by the sum of in- and outdegree.
The LengthRank is the comparison by percentage of this message to the longest

message within the results. The last term, URLRank, is set to a positive constant c if the message contains a link and to zero if not, assuming that links to websites lead to an interesting content and are not used for spam [23].
This algorithm seems to be simple and powerful, but only if the adequate interface already exists to collect the data for the FollowerRank. Furthermore, it can be criticized that the length of a message and the containment of a URL is not necessarily a sign of quality.
This algorithm is an interesting example for the smooth transition between rating and ranking, as defined in this thesis. The name of the three properties - FollowerRank, LengthRank, URLRank - which are aggregated to get a collective ranking give the impression that they are three individual rankings. But a closer look at all three components shows that FollowerRank and URLRank are in fact independently calculated or set values without any comparison to other items and therefore are more ratings than rankings. In contrast LengthRank can really be seen as an individual ranking, because this item is compared to another one and a relative evaluation of the length is made. So this algorithm can also be seen as a special case where different properties of items are evaluated and aggregated as individual ratings as well as individual rankings.

2.2.2 Eigenvector-Based Ranking

It is quite intuitive to think of a graph-based solution for ranking, not only because a ranking can be shown in a graph, but also the social network and the process of voting for someone or something can be visualised. In fact graph-based algorithms for ranking are very popular due to HITS and PageRank, which is used in the Google search engine. As all algorithms presented in this chapter are based on the so-called eigenvector and PageRank is one of the most famous of it, it might be helpful to first have a look at this special algorithm. Though, the mathematical underpinning of eigenvector-based ranking is rather elaborate, PageRank makes use of a very intuitve model.

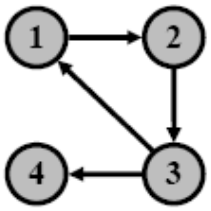
2.2.2.1 PageRank

As PageRank is an algorithm for searching engines in the world wide web, the directed graph $G = (V, E)$, which it is based on, consists of webpages as a set of vertices $V = (v_1, v_2, \ldots, v_n)$ and links between those sites as a set of directed edges E [19, 29]. The dependency of vertices and edges can be easily represented by a $n \times n$-adjacency-matrix, where the entry a_{ij} is 1 if there exists a link from webpage v_i to webpage v_j and 0 otherwise. The $outdegree(v_i)$ is the number of edges coming out of node v_i, also called forward links, which means in concrete the number of hyperlinks on the webpage v_i. In contrast the $indegree(v_i)$ is the number of edges coming into v_i, so-called backlinks, and represents the frequency of other webpages linking to v_i [13, 19, 26].
The basic idea of PageRank is that each page is more relevant, the more other relevant pages link to it [26]. So it is not just a matter of the indegree, but it has also to

be differentiated between links of webpages of good standing or links of any other webpage. So the PageRank Score of a webpage represents its importance and can be seen as the sum of the PageRank Scores of those webpages that link to it [26]. The according recursive formula can be found in Equation 2.4 and shows in the first part a general value each page has and in the second part that the PageRank value depends on the PageRank values of other pages.

$$PR(v_x) = \frac{1-d}{n} + d\left(\frac{PR(v_1)}{outdegree(v_1)} + \ldots + \frac{PR(v_n)}{outdegree(v_n)}\right) \tag{2.4}$$

For PageRank all websites consisting in the world wide web are represented by an adjacency-matrix called Web Hyperlink Matrix, where the 1 for an existing link from webpage v_i to v_j is replaced by $1/l_i$, with l_i denoting the number of links on webpage v_i, so the $outdegree(v_i)$ [29]. This normalization has to be done, to avoid the immoderate influence of webpages with dozens of links on it in comparison to other good webpages with only a few links [11]. An example for an webgraph and its Web Hyperlink Matrix H can be seen in Figure 2.3.

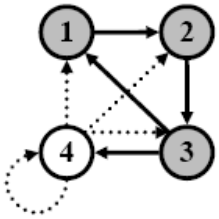


$$H = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2.3: Example of a webgraph and its Web Hyperlink Matrix H [29]

The PageRank algorithm and the Web Hyperlink Matrix can be explained very intuitively by the random surfer model, which states that the probability a random surfer following a link depends on the number of links available on this webpage [26]. This means that the Web Hyperlink Matrix shows the probability of the random surfer to get from one page to another [29]. Because most of the webpages in the world are dead ends or so-called dangling nodes, i.e. documents or webpages without any outgoing links, it has to be taken account of the possibility that the random surfer might be stuck in it [13, 26, 29]. Those websites can be easily identified in the Web Hyperlink Matrix, as the whole row is filled with zeros, because the outdegree is zero if there is no probability to get from this page to another.
The solution for this problem is the imagination that the random surfer in the case of a dead end types in any url address he wants to, which is represented by filling the row with $1/n$, where n is the number of webpages in the whole world wide web [29]. In case of the simple example in Figure 2.3 node 4 can be identified as a dangling node because the fourth row of the matrix is filled with zeros and the problem is taken care of by replacing all zeros in this row by $\frac{1}{4}$, as $n = 4$ in this simplified webgraph. That means if the random surfer is stuck in a dead end, he can get to any page in the web with the same probability by just typing in its address. The mathematical solution to this problem is the formula $S = H + dw$, where d is a

vector identifying the row of the dangling nodes and w is a vector filled with the above mentioned $\frac{1}{n}$. The according example to the earlier webgraph H can be seen in Figure 2.4 including the webgraph S after it is fixed.



$$S = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Figure 2.4: Fixed dangling node by $S = H + dw$ and its according webgraph [29]

The last mentioned situation leads to the thought that the random surfer might type in an address at any time, although there might be links on the actual webpage [11, 29]. To involve this case, the so-called Google Matrix G was designed, calculated by the formula $G = \alpha S + (1-\alpha)\vec{1}v$, where α and v are predefined. The damping factor alpha has to be in the range of $0 \leq \alpha < 1$ and is usually set to $\alpha = 0,85$. The so-called personalization vector v can also be of different type, but the most common and intuitive one is to set it to $v = \begin{pmatrix} \frac{1}{n} & \frac{1}{n} & \dots & \frac{1}{n} \end{pmatrix}$. The symbol $\vec{1}$ represents a column vector filled with ones [29]. So the first part of the Google Matrix formula represents the random surfer following the link structure with probability α, while the second term symbolizes the random surfer choosing a random webpage by typing in any address with probability $1-\alpha$ [13, 29].
The aim of the whole calculation is to get one so-called PageRank vector p where the i^{th} entry represents the PageRank score of webpage i. In general in the equation $Ax = \lambda x$, where A is a given square matrix, the scalar λ is called eigenvalue of A, if there exists a non-zero vector x, the eigenvector, that fulfils die equation [29]. Not every mathematical detail is explained here, only the basic facts that are necessary to know, as any further explanation or proof can be consulted in almost every comprehensive book of linear algebra. If A is a stochastic matrix, that means the entries of each row sum to 1 and all entries are positive, the dominant eigenvalue, that is the existing one with the largest absolute value, is always $\lambda = 1$.
As G is a transition matrix and therefore also a stochastic matrix, the equation for the eigensystem can be rewritten by $Gp = \lambda p$ and as $\lambda = 1$ be shortened to $Gp = p$. This means, by given G it is searched for the unique solution, the PageRank vector p, so that any applying of the transition matrix G does not change the values of the PageRank vector and therefore represents a steady state [13, 29]. Due to the very sparse Web Hyperlink Matrix H, where most entries are 0, the so-called power method to solve this equation converges quite rapidly to p [29]. In the end the results for a specific search term are listed by their rank, which means their PageRank scores.
So the important point of the PageRank algorithm, next to its very intuitive explanation by the random surfer model, is that reducing the problem to an eigenvector calculation makes it in contrast to the vast number of websites in the world very

easy to rank all of them. Therefore, it is not surprising that some authors used it as a basis for their own algorithms even in a different context, as this idea cannot only be applied to webpages.

2.2.2.2 Similar Ranking Algorithms

Many situations can be represented by a directed graph. The nodes can be users and the edges symbolize interactions or relationships between them, like giving an answer, trust or friendship. On the other hand the nodes could also stand for items, like threads on boards, messages or any other data users offer to a community [1]. In this case the edges could also display the timeline or structure of answers or the coherence of different data, like similar topics [30].The variety of algorithms deduced from the idea of PageRank is very huge, therefore only some of those are represented here.

The so-called EigenRep algorithm was designed to calculate the trust among users in a peer-to-peer network. The associated graph shows the trust by edges between the interacting users shown by nodes. After each transaction the downloading user i has the opportunity to vote positively or negatively for the providing user j. Their relationship can be calculated by $s_{ij} = sat(i,j) - unsat(i,j)$, where $sat(i,j)$ counts the number of positive votes of user i for user j and $unsat(i,j)$ the negative ones. The difference of both, s_{ij}, symbolizes the contentedness and experiences of user i with user j. In Equation 2.5 the values are normalized to get relative values for better comparisons. Furthermore, $max(s_{ij},0)$ is used to keep the values in the non-negative region, as s_{ij} could also be negative. As a consequence once again it is not possible to distinguish between a provider someone had negative experiences with and a provider someone never had a transaction with. But by keeping the values between zero and one it is possible to make use of the PageRank algorithm. The $n \times n$ matrix C can be build, where n is the number of users in the peer-to-peer network and each entry c_{ij} represents the relative trust between user i and j. Now it is searched for a so-called trust vector $\vec{t}$, which can be seen as a global reputation vector for all users, so that t_j represents how much the system trusts user j. This can be computed corresponding to PageRank above [18].

$$c_{ij} = \frac{max(s_{ij},0)}{\sum_j max(s_{ij},0)} \tag{2.5}$$

It is noteworthy that the matrix C is once again very sparse, as most users might never had any transaction together and furthermore any negative trust is also reduced to zero. So the calculation of the trust vector can be done very quickly.

Another example is the ExpertiseRank algorithm that is used to find experts in a question-answer-community or on a board. In this case the graph consists of users as nodes and who got an answer from whom as edges. The advantage of such a graph is that it can be assumed that the person, who answered a question, has

more knowledge and expertise than the questioner. Furthermore, it is possible to see similarities and where users have the same interests. But as the graph alone does not include, if someone answered only simple questions of a newbie or very hard questions of another expertise, the PageRank algorithm was adopted to it to overcome this problem.

$$ER(A) = (1-d) + d(\frac{ER(U_1)}{C(U_1)} + \ldots + \frac{ER(U_n)}{C(U_n)}) \tag{2.6}$$

As PageRank, ExpertiseRank can be defined recursively by the formula given in Equation 2.6. $ER(A)$ denotes the ExpertiseRank of user A and depends on the ExpertiseRanks of those users U_1 to U_n he answered questions for. The damping factor d is the same as the α mentioned above in the original PageRank algorithm and is also set to $d = 0,85$. As usually, the ExpertiseRank of the different questioners are normalized, in this case by dividing their rank by $C(U_i)$, which is the total number of users that already helped user U_i. The ExpertiseRank denotes in fact the relative knowledge of one user in comparison to all the other users by analysing the question-answer network [30]. Once again it can be seen that the graph and matrix in this case might be very sparse, as there might be only some users who regularly answer questions and most of the nodes are users who search for help.

There are many more similar examples that are based on the idea of PageRank. On the one hand the eigenvector-based algorithms can be explained quite intuitively by the random surfer model in most cases, no matter what kind of nodes and edges are given. On the other hand, although the computation is not as complex as it seems on first sight, the use of the eigenvector problem needs some preconditions, like positive entries in the matrices, to work. Furthermore, the so-called powermethod, used for computing the eigenvector of PageRank, and other similar methods require the graph to be strongly connected. This condition is in practice often not fulfilled. Depending on the concrete situation of use this could be hard to fulfil and therefore the eigenvector-based idea not as effective as wanted.

2.2.3 Special Case: Collaborative and Content-Based Filtering

Looking for scientific articles about rankings, another field of research turns up, the so-called collaborative and content-based filtering, which are often used in recommender systems. Those are systems mostly used in e-commerce to provide recommendations of new products to customers, based on their taste and profile [28]. This topic is treated separately from other ranking algorithms, as it is a special case, where each user is treated individually and recommendations are made according to one's own profile. The upcoming problem of data sparsity, because one user has too less ratings to recommend a new item, is often solved by involving the profiles of other users to compare preferences and find good recommendations. In contrast to this, BACKSTAGE shall create a list with the best evaluated items and therefore make a recommendation for all users at once disregarding the individual preferences. Indeed, the purpose of BACKSTAGE is to keep the audience of a lecture

together and not to forward its partioning. As recommender systems also use ratings to create a ranking of items, they shall be mentioned here shortly. Although they cannot be used exactly in the same way, they might give interesting ideas for BACKSTAGE. Despite other specific definitions, the two groups collaborative filtering, also sometimes called social filtering, and content-based filtering are the most prominent categories [4, 12, 22, 24].

Collaborative filtering uses some profile, for example purchase or rating history of a user, to find other users with similar taste. This provides the opportunity to predict a users preference for an item he has not evaluated yet, by comparing the similar users' opinion about it [4, 9, 22, 24, 28]. It works by first finding the so-called nearest neighbours, those whose ratings for items are the most similar. Afterwards a weighted combination of the neighbours' ratings is used to compute the likely preference for the unseen items and recommend the best of it [4, 22, 28].
Advantageously collaborative filtering can recommend items without knowing or analysing their content. Therefore, it can also suggest items, which might not be interesting according to the historical profile on first sight, but were liked by the nearest neighbours and thus are still interesting for the user [22]. Although collaborative filtering works very well if one user has only less ratings [4], it suffers from less ratings of all users in general during the start-up. This situation is called sparsity or cold-start problem and describes a phase of too less data for recommendations usually in the beginning, until a sufficient number of rates and users is reached [9, 22, 28]. Furthermore, it is a disadvantageous system for new items, as they will not be recommended to anybody unless at least one person rated it. This situation is called first-rater problem. Additionally, users with an exceptional taste might have no sufficient nearest neighbours for a good comparison [4, 22, 24].

On the other hand content-based filtering works solely with the properties of items and the knowledge of the taste and context of the user to recommend him new items. The taste or interests, retrieved by a given user profile or by analysing historical ratings of the user for items, are compared to the properties of all unknown items and the best matches are recommended to the user [4, 22, 24, 28]. Of course as a consequence all items have to be tagged with some description or keywords, which have to be added manually or, depending on the type of items, could be analysed and retrieved automatically.
The problem with content-based solutions is, beside the necessary descriptions, the sparsity of data for comparison, if the user has no sufficient profile or never rates any items [4]. But in contrast to collaborative filtering it solves the first-rater problem and can also recommend to users with an unusual taste or profile.

As it can be seen both methods have their advantages and disadvantages, hence it is not surprising that there are many experiments with combining both techniques in a hybrid model to eliminate the negative and boost the positive sides [4, 9, 22, 28]. At first sight, the content-based filtering seems to be of no interest for BACKSTAGE, as the analysis of the content is not envisaged so far. Morover it will be hard to manage, as every lecture consists of other important keywords

depending on the topic. Furthermore, not only this method, but also the collaborative filtering works with individual profiles for each user and therefore might be inappropriate for BACKSTAGE, where no comprehensive profiles exist. But a closer look can bring up new ideas, which will be mentioned later in chapter 4.2, and highlight possible problems that should be taken care of. For example the cold-start problem could really be a demotivating factor for users in the beginning of a lecture.

CHAPTER 3

Aging

In the context of rating and ranking some authors describe the so-called "rich gets richer phenomena". It states that items which are abundantly rated and are displayed on the top part of a ranking get even more ratings, as they attract more easily the attention of the user [10]. Additionally to this problem, a pure rating and ranking algorithm could cause some other difficult situations in regard to BACKSTAGE. One general problem is the standstill of the top ranking: If an item was rated extraordinarily well it stays on top of the ranking until it either is replaced by other items, which are better rated, or is intentionally voted down by negative ratings. Another more specific example is an ongoing lecture with many students, where most of them voted for an interesting item, so that it gets on top of the list with a high number of positive ratings. It might be the case that about half of the class leaves the lecture early for example, because of writing a test in another lecture. The remaining half of students will hardly have the chance to vote as often as the full class did and therefore never kick the top item from first place.
In general any system that shows a dynamically changing ranking – and in BACKSTAGE the topic might change with the progress of the lecture – without including any kind of aging will sooner or later fail, because it creates more or less a steady list that will only change in a few extraordinary cases. A steady list might only be interesting in case of showing the best items overall without any time limit, for example the "best rated movies ever" in a movie database. Hence, aging is a necessity in regard to BACKSTAGE, as a dynamically changing ranking is desired, to keep comments and questions in context of the progressing lecture and direct the attention to newer ones. Only a few aging algorithms are included in the mentioned literature so far, but a look at it can show up advantages and disadvantages of different systems and ideas for new ones.
Therefore, this chapter will first present three different aging algorithms. In the next section the requirements for BACKSTAGE are defined, so that afterwards a prototypical algorithm can be designed according to them.

3.1 Review of Aging Algorithms

As most of the rating and ranking applications found in the mentioned literature do not consider the notion of time, aging algorithms are quite rare.
One solution can be found in [12] in the context of collaborative filtering, where the authors mentioned that the taste of a user might change over time. Therefore, older ratings for items shall weigh less for the filtering than newer ones. They included a time-dependent function $f(t)$ in their preference prediction algorithm, where t is the concrete moment in time when the prediction was done and the opinion of a user for an item was created. As $f(t)$ is supposed to form some weight, it has to lose value over time. Therefore, it is a monotonic decreasing function between zero and one. Two predestinated mathematical functions for this case are the exponential and the logistic function, which have similar graphs.
The authors defined the time function as $f(t) = e^{-\lambda \cdot t}$, where λ denotes the decay rate by $\lambda = \frac{1}{T_0}$. As known from the modelling of nuclear decay processes the half-life parameter T_0 defines how long it takes, to reduce the weight by half. Thus, the lower T_0 is, meaning the time to reduce the weight by half is shorter, the higher the decay rate λ and the faster $f(t)$ decreases, which will graphically result in a steeper curve. As already mentioned in [12], the selection of T_0 is a fundamental aspect and depends on the concrete context of the application.

Another quite similar use case in the context of social tagging sites is the freshness of items, not by their date of creation but by regarding how many tags were added to it recently [17]. The described algorithm computes the best and freshest results according to a query based on tags. For each tag in the query an overall score out of different weighted factors is calculated. This, however, is not explained in detail here. We rather focus on the aging aspect here.
Therefore, the timeline is divided in discrete and equally spaced time slices, where m counts the number of time slices. To get the time-dependent score of one factor the sum over all existing time slices is determined. For each time slice, starting with index $s = 1$ and ending with the current time slice $s = m$, the specific factor score for this combination of item, user and tag is computed and afterwards weighted by a^{m-s}, where the decay factor a lies between zero and one. Scores of older time slices get less weight than scores of newer time slices. The newer the time slice and therefore the score, the smaller the exponent $m - s$, the higher the whole multiplier, so that this score has more influence. On the other hand the older the time slice, the higher the exponent, the lower the whole multiplier, which will reduce the influence of that score on the whole sum to a minimum [17].
Once again this system depends on choosing the right value for a, as the higher a is, the faster the decay and the more important is the freshness.

The interesting project Backchan.nl, which is already referred to in the chapters above, uses a so-called age factor (Equation 3.1). Like in the previous aging algorithm the notion of age is independent from the creation time of a post, but relies on the last votes it received.
The age of a vote v for a post p is defined as the subtraction of its timestamp $t_{p,v}$

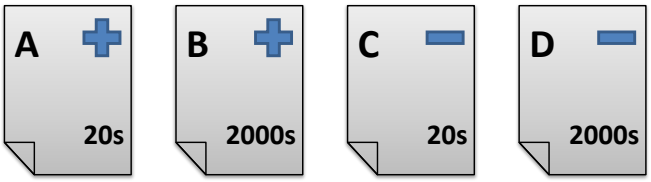
from the current timestamp t_{now}. The authors decided to only take the last five votes into account for computing the average age of votes received by the item and furthermore make use of the arithmetic mean. This average age is related to some time constant τ, which the authors predefined to be $\tau = 10^4$, but also pointed out that τ has to be chosen according to the specific situation [16].

$$ageFactor(t_p, t_{now}) = \frac{\overline{t_{now} - t_{p,v}}}{\tau} \tag{3.1}$$

At the first sight this formula seems to be quite intuitive, but a closer look reveals several problems. In contrast to the both aforementioned algorithms the age factor does not definitely lie between zero and one. Although the division by τ might work fine in most cases, there is a remaining possibility that the average age of the last five votes for a post might be older than 10^4 seconds (i.e. approximately 3 hours), such that the age factor could be larger than one. This is not necessarily a big drawback, but it should be recognized that there is no upper bound.
The whole formula of the Backchan.nl project seems to be inscrutable in general, as much important information is missing in the paper. The authors use the vote factor formula, which was already mentioned in the "Parametric Solutions" chapter in the context of ranking (also cf. Equation 3.2), and multiply it by the age factor formula of Equation 3.1 to get the overall scoring for each item. Already at this point crucial information is missing: Is the ranking determined by odering the items according to their scores in ascending or descending order? One might think that a closer look at the formula might help. But actually it reveals another oddity that is never explained in the paper: the confusing interplay between the vote and age factor.

$$voteFactor(p) = \frac{kU_p}{U_p + D_p} + \frac{U_p + D_p}{\overline{U + D}} \tag{3.2}$$

To understand this problem, a closer look at the vote factor formula has to be taken. The first term rewards the positive votes for a post and ranges between zero and one, if k is ignored, which means in case of $k = 1$. As posts with a high number of up-votes should get a high ranking, it can be said that the bigger the first term, the better the post or otherwise the closer to zero, the more negative votes the post received. The authors state that the parameter k is used to create a balance between the first and the second term. Actually, the word balance in this case is quite critical, because if k symbolizes some kind of weight for the first term, the counterweight of the second term, something like $k - 1$, is missing. In the formula above k does not influence the second term and therefore just regulates the impact and importance of the number of positive votes in the first term.
Furthermore, the paper does never state which value for k is suitable, nor in which range it lies, hence it could even be negative. If we at least assume that k is a positive constant, $k = 1$ means the up- and down-votes are treated equally, if $k > 1$ the weight of the up-votes is boosted. On the other hand if $k < 1$ the weight of the down-votes is somehow boosted, as the impact of the up-votes is reduced. It can be concluded, that the first term of the vote factor formula lies between zero, if the



	vote factor	age factor	final score
A	$k * \frac{100}{100} + \frac{100}{100} = 1 * 1 + 1 = 2$	$\frac{20}{10000} = 0,002$	$2 * 0,002 = 0,004$
B	$k * \frac{100}{100} + \frac{100}{100} = 1 * 1 + 1 = 2$	$\frac{2000}{10000} = 0,2$	$2 * 0,2 = 0,4$
C	$k * \frac{0}{100} + \frac{100}{100} = 1 * 0 + 1 = 1$	$\frac{20}{10000} = 0,002$	$1 * 0,002 = 0,002$
D	$k * \frac{0}{100} + \frac{100}{100} = 1 * 0 + 1 = 1$	$\frac{2000}{10000} = 0,2$	$1 * 0,2 = 0,2$

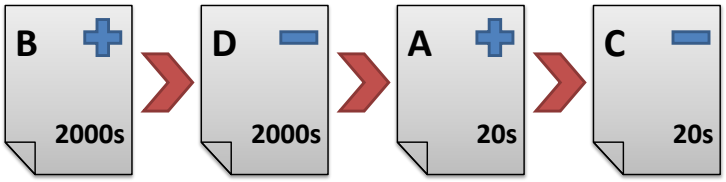


Figure 3.1: Example for calculating the ranking of items A, B, C and D according to the algorithm of Backchan.nl project

post got no up- but maybe down-votes, and the constant k, if the post got only up- but no down-votes.
Furthermore, as already mentioned, the higher the value for the first term, the better it ranking should be.
The second term of the Equation 3.2 rewards the voting activity around a post. The term yields zero, if the post received no votes at all and one, if it exactly coincides with the average voting activity of all posts. Accordingly the term lies between zero and one, if the voting activity is below average, and above one, if the voting activity was above average. Once again it can be seen that the higher the value for the second term, the better it should be ranked. Thus, it can be concluded that the whole vote factor formula results in a high ranking by a high value.
As the authors state that newer posts should replace older ones, one might assume that the age factor formula would also push up young posts with a high value and pull down older posts with a lower value. But in fact it works just the other way round, the lower the numerator of the fraction, i.e. the younger the post, the lower the result of the whole quotient. Conversely, the higher the numerator, i.e. the older the post, the higher the value of the fraction. Hence, the lower the value for the age factor, the younger and hence the better is the post.
The critical point is reached when vote and age factor are combined by multiplication. For a good ranking the first factor should be very high and the second very

low. However, both values are compensating for each other and the overall ranking becomes quite confusing.
The example in Figure 3.1 illustrates the complexity of this inconsistency by using four extremely opposed posts. A ranking for the posts A, B, C and D shall be created, where post A and B got only positive votes and in contrast C and D only got negative votes. The constant k is set to $k = 1$, as we do not know the exactly value and this seems to be the neutral one that does neither boost positive nor negative votes. To keep it simple, the number of votes for each post is set to 100 and even the average voting activity is set to 100. Two of the posts, A and C, are very young with an average age of 20 seconds for the last five votes and the other two posts, B and D, are very old with an average age of 2000 seconds.
As can be seen in Figure 3.1, the resulting ranking for this example would be B, D, A, C, if it is ranked according to the highest value. The best item, B, has in fact many positive votes but is also very old and cannot be kicked out of the list by a younger one. The second item in the ranking, post D, is even a negatively voted item that reaches the top only because of its age. The young post with the highly positive votes, A, which was assumed to be quite on top of the list, stays at third place. Ordering according to the lowest value instead would result in the ranking C, A, D, B, where the youngest posts are rewarded, but negatively voted posts are placed before positive ones. It seems that in the first case the age factor has a significant influence on the whole formula and rewards the older posts, regardless if their votes are positive or negative. In the second case the order by age seems to be correct, but negative votes are preferred.
Although the idea of the Backchan.nl project is very good and the formula for the votes seems to work fine, the aging formula and the lack of information make this approach quite inexplicable.

In summary it can be seen that all three aging algorithms depend on the real time, which is included by concretely determining the half-life parameter, dividing the time line into slices or simply counting the seconds of age. In contrast to the Backchan.nl project, the first and second algorithm use their aging formula as some kind of weight in their whole algorithm, which reduces the value and therefore the importance of an item over time.

3.2 Algorithm Requirements for BACKSTAGE

A look at the mentioned papers about rating and ranking reveals the necessity and importance of defining requirements for a project before designing an according algorithm. It is obvious that the algorithm depends on the concrete use case and the desired properties it has to fulfil.
The requirements found in the literature can be grouped in four categories. First, user focused aspects, like the rating shall be easy to use [21] or that latency should be low, so that the influence of a users vote can be seen immediately, which will keep the motivation high to make use of the rating [10].

Second, general and administrative aspects, for example a self-policing design, so that the users themselves decide which content is right or wrong and no intervention by an admin or similar is necessary to avoid spam or misuse [18]. Furthermore, the algorithm has not only to result in an accurate ranking, but also should already work with less feedback [10]. In general, like all algorithms, it should be very efficient [21].
The third category, rating and ranking related requirements, includes very specific aspects in relation to the concrete purpose of the algorithm. Fairness of the rating and ranking algorithm is one general example [10]. Another more precise requirement states that highly positively voted items should be high in the ranking, while on the other hand items, which are highly controversial, should also be rewarded to some kind, but not as much as positive ones [16].
The last category of requirements is related to the aging and dynamic aspect. Besides the popularity, which means the positive interest and hence positive votes, the freshness of items, the influence of time and the dynamic changes coming along with it should be regarded [17, 21]. This involves not only the fact that maybe newer and fresher items should replace older ones in the ranking, but also that items which constantly get rated over a long period of time should somehow be rewarded for it [16].

Most of the mentioned requirements are also suitable for the BACKSTAGE application. We want the algorithm to handle explicit positive and negative votes, which could not only express if a user agrees, maybe even has the same idea or question, or disagrees with a post, but also if it is appropriate or not. Using this kind of rating scheme we also rely on the belief that the community will regulate itself and do not take into consideration any mechanism for preventing spam. As we want the algorithm to be very efficient in regard to the computation time and as it should already work properly with only a few votes, we search for a very simple algorithm in contrast to the literature. Like in the above mentioned requirements we want a descending order where highly positively voted posts are on top and controversial posts get a better ranking than negative ones, but not better than positive ones to keep fairness. The last category of requirements, the aging and dynamic, is crucially important for BACKSTAGE, as we want a constantly changing top-k list, where newer items can replace older ones in the ranking and regularly voted on items stay higher ranked than ones which seem to be out of the focus. All these requirements should be in one's mind, designing the according algorithm.

3.3 Prototypical Algorithm for BACKSTAGE

It makes sense to first design a simple prototypical algorithm for BACKSTAGE, to see if a simple solution might work too. If the experiments with this prototype show any difficulties or drawbacks, the algorithm can be redesigned, improved or expanded anyway.

As mentioned above the rating scheme we want to use consists of an explicit plus or minus vote. Each vote has the same weight to keep fairness between the users, in this case a plus vote will count $+1$ and a minus vote accordingly -1. The voting score for one post can be computed by summing up the value of all votes it got. As each user is only allowed to vote once for each post, it can be guaranteed that the score of a post is limited by the number of participants in this lecture. Of course this will result in the fact that posts with no votes and post with balanced votes look the same in their scores, but as we definitely want the possibility to vote negatively for a post this seems to be inevitable. The individual rating will be aggregated by summation and result in a collective ranking. If the aging was disregarded, the collective ranking would be created by an descending order according to the voting score for each post.

As already demonstrated the aging cannot be neglected. A first simple idea to measure the age of a post would be computing the current timestamp minus the timestamp of creation for that post. But this solution would be inappropriate, as posts which are regularly voted on would age in the same speed as posts which are disregarded by the community. This does not meet our requirements mentioned above.
As a consequence it would make sense to rely the age of a post on the votes it got. By this solution the aging depends on the attention a post gets and it will age at the moment it gets out of focus. The first idea might be to calculate once again the current timestamp minus the timestamp of the last vote the post got, but this could lead to some awkward situations. For example, if many students voted on a post a long time ago, so that it is quite old, and one individual student would vote for this post now, it will suddenly be very young again. This does not seem to be a very representative solution.
Although calculating the arithmetic mean out of all votes seems to solve this problem, it still disadvantageously gives much power to outliers. Many votes at the same time would be needed to assure that the age of this post is some kind of robust.
To overcome these difficulties the median is used to compute the average age for a post. The median of a frequency distribution is the exact value where as many values lie beneath the median as above it. This means, the median divides the distribution in half. Its value is often used as a form of average, because the median considers the distribution of the different values and hence is very robust to outliers.
Therefore, the number of votes for one post is counted and added one, to include the creation time of a post, which seems to be useful in case a post might get no attention in the beginning but in the end. This number is divided in half to determine the specific vote that's age is used as the age of the post at this moment. Where necessary decimals are rounded down. This solution is much more robust to outliers and highlights the moment of most votes for this post.
The difference between the arithmetic mean and the median can be seen in an example in Figure 3.2. The stars above the timeline symbolize the moments votes for that post came in, regardless of the value it got, except the first star two minutes

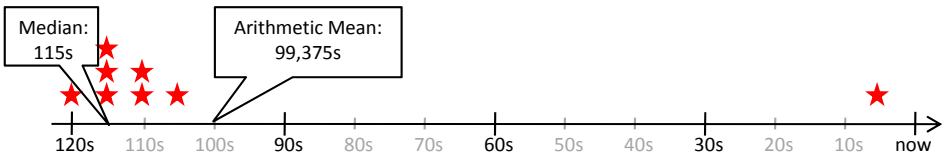


Figure 3.2: Example for the different results of arithmetic mean and median in the same situation

ago, which stands for the moment of creation of this post. The calculation of the arithmetic mean in Equation 3.3 results in 99,375 seconds as average age for the whole post. The relevant point of time for the median is computed by $\frac{8}{2} = 4$, which means that the fourth timestamp, thus 115 seconds is used as the age for the whole post. Both ages are also marked in the figure and show that already one outlier can move the average considerably using the arithmetic mean, while using the median is quite robust.

$$AverageAge(p) = \frac{120s + 3 \cdot 115s + 2 \cdot 110s + 105s + 5s}{8} = \frac{795}{8} = 99,375s \tag{3.3}$$

The next question that comes to one's mind is, when the refreshing of the top-k list and thus the computing of the new scores and ages shall take place. Computing all values for each post each time a new vote comes in seems to be too much overhead, especially in lectures with hundreds of students. Determining some kind of time period for refreshing is necessary, but very difficult. Regardless which time period is chosen, the problem stays the same: If the period is chosen too long and many votes come in during that time, the top-k list might radically change in the next refreshing step. On the other hand, if the period is chosen too short much overhead can be created if nothing happens on the backchannel during that time. It does not seem to make sense to depend the refreshing on the real time, instead it should be based on the activity on the backchannel during the lecture.
As a very simple and direct solution a post counter was chosen, which initiates the refreshing calculation every n posts. The parameter n has to be chosen accordingly to the size of the audience or maybe also to the speed of the lecture. This "logical clock" ticks along with the traffic on BACKSTAGE and seems to be a very good solution for fluctuating classes. Furthermore, it is more expressive, as the physical time does not state anything about the lecture progress. For example, in two minutes the lecturer could click through ten introducing slides with pictures and no one posts a question or on the other hand he could show only one slide with a highly complicated mathematical formula in the same time and many questions come up. Neither the physical time nor the number of slides can appropriately react to the traffic on BACKSTAGE, as it highly depends on the audience.

To combine the aging and scoring for a post a totally different idea than those in the

mentioned literature came up. As the distance between two posts in the end is not relevant for BACKSTAGE, it does not matter if it gets lost during the calculation. The whole algorithm should yield in a top-k list, regardless of the fact if post A is ten times better than B or just one vote better. The only aspect that matters is A being better than B. Therefore, two lists were created, both containing all posts that at least got voted once. In one list the posts are sorted by their age with the youngest first, while the other list is sorted by the score of the posts with the most positively voted first. To get the overall rating score for one post, the number of its position in the aging list is multiplied by the number of its position in the score list. An example would be a post A, which is in place nine of ten in the aging list and comes in second of ten in the score list; hence, in comparison to the other rated posts it is quite old but has also a very good evaluation. As the number of its positions are multiplied, the rating score would result in $9 \cdot 2 = 18$.
The collective rating is created by an ascending order of the rating scores, because a low rating score symbolizes a low and therefore good position in one or both lists. Instead of the multiplication a simple summation was also imaginable, but sums have a higher probability of resulting in the same value, although using totally different summands. Like the result 9 can easily be reached by summing $1+8$, $2+7$, $3+6$ and $4+5$, whereas it is only the result of the multiplications $1 \cdot 9$ and $3 \cdot 3$.

So in summary, the plus and minus votes are immediately added to the current score of a post, but the whole refreshing and aging function is not initiated until a specific number of posts come in. In that case, the age of all posts that received votes during the last time tick is recalculated using their median ratings. As the median of posts which did not receive any further votes does not change, it is not necessary to consider their recalculation. The aging and score list are created by sorting all posts in two ordered lists, according to their increasing ages and according to their increasing scores. For each post its position number in the first list is multiplied by the position number in the second one and saved as its rating score. In the end all posts are sorted by their rating score in an ascending order to get the collective ranking and the top-k entries can be easily displayed.

In order to prevent spam messages and keep the anonymous and self-policing structure of BACKSTAGE, it could be an idea to motivate the students to use the system seriously by introducing user weights, to maximize the influence of credible and minimize it for incredible users. A system like this does of course relies on the assumption that most of the students participating in BACKSTAGE are credible and would denounce spam messages by giving negative votes for it. As a total circumvention of spam is not possible, but if the students know that user weights are used to regulate the influence of ratings, they will probably be motivated to get a good user weight to maximize their influence and therefore behave well. A possible algorithm and usefull approach for user weights will be shown here.
The user weight should influence the significance of a users rating, by minimizing the influence, if the user is incredible and maximizing it, if the user behaved well in the past. Each user weight is calculated by the arithmetic mean of the ratings a user received for his posts, regardless of the age of the post. If most of his

messages, questions or answers were of interest for the community and got voted positively, his weight will be positive and vice versa for mostly negative messages. For example user A is the creator of three posts, one got only positive votes and has an average score of $+1.0$, the second and third post received mixed votes and have an average score of $+0.6$ and -0.1. The user weight lies generally between $-1 \leq w(u) \leq 1$, as the extreme cases consist of only positive or negative votes for the posts of a user, which will emerge as $+1.0$ or -1.0 in the average ratings. Hence, the user weight $w(u_A)$ of A will be $\frac{1.5}{3} = 0.5$. As the weight could also be negative, a multiplication of one's rating value by his user weight is not useful. Therefore, the weight is rather incorporated by adding it to or subtracting it from the rating value. Because user A has a positive weight, his ratings should be boosted, hence, if he votes a post positively his sent-in rating score will be rating value $(+1)$ plus user weight $(+0.5)$. As a result, user A adds $+1.5$ to the score list of the post he voted positively for. On the other hand if he votes negatively (-1) it makes no sense to add the positive user weight $(+0.5)$, as this will reduce his influence (-0.5), instead the user weight is subtracted (-1.5). In case of a negative user weight, imagine user B having a weight of $w(u_B) = -0.5$, the algorithm will work the same way. If user B votes positively $(+1)$, his weight is added and his influence reduced $(+0.5)$ and if he votes negatively it is subtracted $(-1 - (-0.5))$, which will also result in less influence (-0.5).
The overall score, regardless of the age, for the posts is once again calculated by the arithmetic mean out of all weighted ratings. As the average could now be between -2 and $+2$ normalization is needed, otherwise weighted vote and post scores might escalate. For example, if the average of a post could be $+2$, the weight of a user could be $+2$, hence, his positive vote would be $+3$ at most, which closes the loop as the average rating score of a post could now be between -3 and $+3$. It can be thought of normalizing the values by either dividing the weighted vote or the user weight by two.
To keep fairness, new users will start with a weight of zero, which disadvantageously gives incredible users the possibility to refresh their weight by just registering a new account if the system permits it. It seems that some kind of identification is inevitable, for example, allowing just one account per enrollment number. On the other hand it could also be an advantage to allow new accounts, because students who have learned to behave get the possibility to start over new. Advantageously this suggested user weight algorithm is kept very simple and could be easily extended with further subtleties. It fulfils the basic requirements of reducing the influence for incredible and boosting it for credible users, while working without any further preconditions. Of course the research of literature and experiments are needed to confirm this idea.

The pseudo-code of this algorithm can be seen on page 33. It has to be noticed, that the top-k list does not contain any post which has not received any votes yet. Therefore, especially in the beginning of a lecture, it could happen that negatively rated posts are also shown in the top-k list, while ignored posts do not appear. At first sight this simple idea seems to work right, but some experiments during lectures of different sizes will be necessary to evaluate its performance.

Algorithm 1 Rating with User Weight

```
1: procedure POSTRATING(v, u, p)                    ▷ user u gives post p vote v
2:     amr(u) ← arithmetic mean of ratings for u's posts
3:     w(u) ← amr(u)/2                              ▷ normalization of user weight
4:     if v == −1 then
5:         weighted_v ← v − w(u)
6:     else if v == +1 then
7:         weighted_v ← v + w(u)
8:     end if
9:     add weighted_v to ratings of p
10:    calculate arithmetic mean of ratings for p
11:    save as score of p
12: end procedure
```

Algorithm 2 Refreshing Top-k-List

```
1: procedure REFRESHING(n)
2:     if incomingpostcounter mod n == 0 then    ▷ refresh every n incoming posts
3:         get freshRatedPosts
4:         calculate median age for posts
5:         save as age of posts
6:         get ArraySortedByAge                  ▷ Array of all posts sorted by age
7:         get ArraySortedByScore                ▷ Array of all posts sorted by score
8:         for i = 0 to i = CountArraySortedByAge do   ▷ Search for all posts in first
   array the same post in second array
9:             search for same post in ArraySortedByScore
10:            multiplicate positions in both arrays and save as rating
11:        end for
12:        get ArraySortedByRating    ▷ Array containing all posts sorted by rating
13:        show top − k − list
14:    end if
15: end procedure
```

CHAPTER 4

Discussion

As the algorithm designed in the chapter before is only a simple and prototypical solution, it is obvious that it has some deficits and drawbacks. Solving these problems will inevitably lead to a more complex algorithm and system design, as further mechanisms need to be integrated. On the other hand the possible solutions reveal further ideas for the future development of BACKSTAGE.
Therefore, this chapter will first summarize the deficits of the designed algorithm in order to provide possible solutions. Afterwards, as a result of it, some visionary ideas will be developed and described.

4.1 Limitations of the Designed Algorithm

Having in mind the mentioned literature and the drawbacks of different algorithms, it can easily be noticed that some deficits also appear in the prototypical algorithm of Section 3. The cold-start problem can also be found here, although our algorithm does not directly concern the scientific field of collaborative filtering. Though, in the beginning of a lecture there might be not enough ratings by the users to fill up the top-k list and due to the sparse number of votes for each post, they might also be not reliable enough. The top-k list will be completed if at least k different posts have got any votes, which is another disadvantage, as also negatively rated posts could appear in it. This seems to be analogue to the so-called first-rater problem, as posts which did not receive any ratings cannot appear in the top-k list.
Although the cold-start problem is first and foremost a problem of the community, which does not produce enough ratings, the algorithms differ from each other in how well they manage this situation. If it is not possible to encourage the audience to produce more ratings, the algorithm has to take care of this problem. Therefore, this problem is well-considered in the scientific field of collaborative filtering and should be mentioned here, though the cold-start might be only a minor problem in

BACKSTAGE.
To solve this problem, on the one hand negatively voted posts can be excluded from the top-k list, which will result in a clean version of it, but could also protract the cold-start problem. On the other hand posts with no votes could be included with a score of zero and their age of creation to at least display neutral instead of negative posts. Though, this does not necessarily avoid negative posts in the top-k list, as the number of positive and not voted posts could be lower than k and therefore the list will be filled up with negative posts.
Solving the cold-start problem could also eliminate the first-rater issue in our case. As already mentioned in chapter 2.2.3 content-based methods can be a solution, by analysing the content instead of or additionally to using ratings. The content analysis can be designed to be very simple or extremely specific. For example, it can be thought of a simple algorithm where posts which include a question mark get boosted, as this message will probably contain an interesting question, if questions are to be promoted independently of their endorsement by the student community. To improve this algorithm, a minimum number of characters could be defined, so that questions without any context, like "why?" or "what?" are eliminated. In contrast to this simple solution, the extremely specific version could make use of predefined keywords, to find interesting messages. These keywords could be typed in manually as a kind of catalogue or for example filtered automatically out of the slides for the lecture. Generally content-based solutions for BACKSTAGE have a major drawback, as the content changes with each lecture and lecture topic, therefore an analysis will be difficult to manage. Although the content analysis also yields false-positives, it could at least help to reduce the time of the cold-start and eliminate the first-rater problem in first place. It could be an option to use the content-based method additionally and permanently to the ratings. On the other hand the mentioned Bayesian Rating could also be applied here, by using the content-based method in the beginning and rely on the ratings, if a certain and reliable number is reached. Introducing implicit feedback and ratings could also help solving these problems. It can be a possibility to get earlier more ratings, although the user does not know about it. This idea is once again considered in the next chapter.

Another problem that can be easily remedied concerns the "logical clock". Although this idea is very appealing, some experiments have to be done to assure its efficiency and to find the right configuration. In the described algorithm of Section 3, the clock ticks every time a certain number of posts have come in. This seems to be a reasonable solution, as the frequency of posts is a good indicator for the comprehension and speed of the lecture. If the students understood everything the lecturer said, there will be probably less posts than if things remain unclear and questions arise. Refreshing the age and the top-k list more often in critical phases seems to be rational, as the focus of the students probably lies on the new and actual questions, which should get the possibility to enter the top-k list. However, this solution is already very specific, because originally the clock should been dependent on the activity on the backchannel. The term "activity" could be defined in different ways, for example, the frequency of new posts, votes or both. Depending

on the context it can also be defined very specific, like counting the frequency of new posts, which are no answers to others and therefore are probably totally new questions or comments. In another sophisticated solution, the clock could depend on the usual activity on the backchannel. If usually x posts come in per minute, the clock should tick every $y \cdot x$ times, where y can be selected in the beginning of a lecture. For example, in case of choosing $y = 3$ the clock will tick every three minutes, if the activity on the backchannel is as usual, faster if there is more and slower if there is less activity.
The algorithm mentioned above disadvantageously does not tick if many votes but no posts come in. This could lead to the fact that the votes might completely change the top-k list, but the change is not displayed until the next post is received. In an extreme case it could happen that from one moment to another the whole top-k list is substituted by new items, which could confuse the audience. Therefore, the "logical clock" should be customizable, so that the lecturer can choose before starting the backchannel, on what to depend the aging and changing on the backchannel. This might not only be a good solution because of the different tastes of the lecturers, but also because the clock can be adjusted for different class sizes. Some experiments in smaller and huge classes are necessary to evaluate the efficiency, advantages and disadvantages of all possible options.

Like every quite anonymous social system, BACKSTAGE could also be of interest for spammers. As already mentioned in relation to the requirements, a central management is not desired, BACKSTAGE should rather be rational, because the goal is to build a community. Hence, there will be no or little verification by someone and if the self-policing will not work, it could happen that spam messages, like for example a joke posted by someone, gets boosted by intentional positive votes. However, off-topic messages and jokes shall not be completely removed from the system; as such posts every now and then can convey humanity and ease the communication on the backchannel. Although some spam messages could help to promote the activity on BACKSTAGE, too much of them or even offending ones can ruin the communication.
To avoid situations like these, it could help to abstain from the anonymity, as the possible identification of offenders will probably inhibit people to abuse the system. On the other hand it could also inhibit students from asking questions, because the anonymity gives them the power to defeat their fear of speaking in public. Of course it can be thought of using nicknames instead of real names to facilitate the identification if possible on the one hand and guarantee anonymity on the other hand, like it is done in BACKSTAGE. But even if the nicknames of spammers are banned from the system, they could try to create a new account maybe even with another fake name.

A last and minor problem appears if the multiplications of positions in the score and aging list produce the same result twice or more. This does not necessarily have to be a problem, as in some cases no strict order is needed. However, to avoid this problem it can be thought of introducing a priority for one of the lists. For example, post A with place 4 in the aging and 5 in the score list would produce

the same overall score as post B with places 2 and 10. To get a unique solution the score of a post could be given a higher priority than its age and therefore A could be ranked higher than B, because of its better score. It is understood that the priority could also be set for the age instead of the score.
Generally, it can be thought of giving a weight and counterweight to both lists to increase the influence of one factor and decrease it for the other. This is another property that might be a matter of taste and could be set by the lecturer. Therefore, experiments will be necessary to analyse possible advantages and disadvantages of different set parameters.

All in all the designed prototypical algorithm for BACKSTAGE could be a great basis for further development, as it is a totally different approach for a rating and ranking system. After a few experiments in different class sizes, it could be extended by the mentioned methods to minimize its drawbacks and deficits.

4.2 Future Work

Analysing the disadvantages and shortcomings of the prototypical algorithm gives rise to new ideas and visions for the further development of BACKSTAGE.
Generally, it can be thought of extending the whole BACKSTAGE application with several new ideas. For example, giving users the possibility of marking favourite posts or even save them for their personal account, opens up the opportunity to recommend them other interesting messages they might have missed. This would be an extension with respect to the already mentioned field of content-based and collaborative filtering.

Regarding the rating, ranking and aging aspect, BACKSTAGE should be extended by a implicit rating system. Additionally to the explicit rating the above mentioned marking and saving of posts could be counted as a positive vote for a message, as this signals interest for the post. Even answering a post can be evaluated the same way, as it symbolizes either a user with the same opinion or question or a user who respects the comment as noteworthy. Of course some answers can be negative and without much context, like "stupid comment" and therefore should not be counted as a vote for the original post. On the other hand it also means that the answering user was at least interested enough in it to spend some time on typing an answer, therefore, this post might still deserve some attention. A further way to get implicit ratings is comparing the words of the message with keywords, for example taken from the slides. A match probably symbolizes a question or comment in context of the lecture, which is desired.
Implicit rating advantageously could help to solve the cold-start problem, as more ratings come in earlier. Furthermore, it helps to stabilize the whole rating and ranking system, as the users will have to know, how the system and algorithm work to misuse it. Explicit rating algorithms can easily be seen through, while implicit rating algorithms could hide their sources of ratings.

There are many new ideas for BACKSTAGE, solving known deficits or refining the system. Each new vision is not only a possible improvement for the application, but also a new challenge, as some research is necessary. Advantages and disadvantages of different versions and properties have to be deliberated to realize an idea.

CHAPTER 5

Conclusion

This thesis gives an overview of different rating and ranking algorithms found in literature and furthermore highlights advantages and drawbacks of different possible solutions. By accurately defining the terms "rating" and "ranking", it is not only possible to structure the algorithms according to their properties, but also to describe the intertwining and confusion of both terms. The overview discloses an unlimited number of possibilities to design and create rating and ranking algorithms. It can be concluded that each individual application needs its own and very specific algorithm design.
The important role of an aging process and different algorithms making use of some form of time decay is presented. The literature shows that most aging algorithms depend on the physical time and are not of interest for BACKSTAGE. Instead a "logical clock" was designed, which depends on the activity on the backchannel. This timer is not only very reliable, but also flexible as the parameters can be changed.
After defining the requirements for BACKSTAGE, an according prototypical algorithm was designed, which integrated a complete different approach for rating, ranking and aging. The deficits of the prototype were mentioned and possible solutions discussed. In the end the concrete design gave ideas for visions and future work.

In summary this thesis shows that it is hardly possible to design a complete simple rating, ranking and aging system without any deficits. On the other hand resolving these problems will inevitably lead to a complex algorithm. Nevertheless, the new idea for rating and ranking as well as the logical clock are promising possibilities for BACKSTAGE. Future experiments are necessary to prove the efficiency of the algorithm, while further developments of the project might turn BACKSTAGE into an outstanding application for education.

Bibliography

[1] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne, *Finding high-quality content in social media*, Proceedings of the international conference on Web search and web data mining (New York, NY, USA), WSDM '08, ACM, 2008, pp. 183–194.

[2] G. Al-Sharrah, *Ranking using the copeland score: A comparison with the hasse diagram*, Journal of Chemical Information and Modeling **50** (2010), no. 5, 785–791.

[3] Richard J. Anderson, Ruth Anderson, Tammy Vandegrift, Steven Wolfman, and Ken Yasuhara, *Promoting interaction in large classes with computer-mediated feedback*, April 04 2003.

[4] M. Balabanović and Y. Shoham, *Fab: content-based, collaborative recommendation*, Communications of the ACM **40** (1997), no. 3, 66–72.

[5] Daniel Baumgart, Alexander Pohl, Vera Gehlen-Baum, and François Bry, *Providing Guidance on Backstage, a Novel Digital Backchannel for Large Class Teaching*, Education in a Technological World: Communicating Current and Emerging Research and Technological Efforts, Formatex, Spain, 2011, p. in press.

[6] J. Bian, Y. Liu, E. Agichtein, and H. Zha, *Finding the right facts in the crowd: factoid question answering over social media*, Proceeding of the 17th international conference on World Wide Web, ACM, 2008, pp. 467–476.

[7] Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha, *A few bad votes too many?: towards robust ranking in social media*, Proceedings of the 4th international workshop on Adversarial information retrieval on the web (New York, NY, USA), AIRWeb '08, ACM, 2008, pp. 53–60.

[8] F. Bry, V. Gehlen-Baum, and A. Pohl, *Promoting awareness and participation in large class lectures–the digital backchannel backstage*, Proceedings of the IADIS International Conference on e-Society, Avila, Spain, 2011, pp. 27–34.

[9] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi, *Short and tweet: experiments on recommending content from information streams*, Proceedings of the 28th international conference on Human factors in computing systems, ACM, 2010, pp. 1185–1194.

[10] Anish Das Sarma, Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy, *Ranking mechanisms in twitter-like forums*, Proceedings of the third ACM international conference on Web search and data mining (New York, NY, USA), WSDM '10, ACM, 2010, pp. 21–30.

[11] C. Ding, X. He, P. Husbands, H. Zha, and H.D. Simon, *Pagerank, hits and a unified framework for link analysis*, Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2002, pp. 353–354.

[12] Y. Ding and X. Li, *Time weight collaborative filtering*, Proceedings of the 14th ACM international conference on Information and knowledge management, ACM, 2005, pp. 485–492.

[13] A. Farahat, T. LoFaro, J.C. Miller, G. Rae, and L.A. Ward, *Authority rankings from hits, pagerank, and salsa: Existence, uniqueness, and effect of initialization*, SIAM Journal on Scientific Computing **27** (2006), no. 4, 1181–1201.

[14] Vera Gehlen-Baum, Alexander Pohl, and François Bry, *Assessing backstage–a backchannel for collaborative learning in large classes*, Proceedings of the 14th International Conference on Interactive Collaborative Learning (ICL 2011) (Piestany, Slovakia), 2011, pp. 154–160.

[15] Gabriela Grosseck, *To use or not to use web 2.0 in higher education?*, Procedia - Social and Behavioral Sciences **1** (2009), no. 1, 478 – 482, World Conference on Educational Sciences, Nicosia, North Cyprus, 4-7 February 2009 - New Trends and Issues in Educational Sciences.

[16] Drew Harry, Dan Gutierrez, Joshua Green, and Judith Donath, *Backchan.nl: integrating backchannels with physical space*, CHI '08 extended abstracts on Human factors in computing systems (New York, NY, USA), CHI EA '08, ACM, 2008, pp. 2751–2756.

[17] Wenyu Huo and Vassilis J. Tsotras, *Temporal top-k search in social tagging sites using multiple social networks*, 2010.

[18] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina, *The eigentrust algorithm for reputation management in p2p networks*, Proceedings of the 12th international conference on World Wide Web (New York, NY, USA), WWW '03, ACM, 2003, pp. 640–651.

[19] J.M. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM (JACM) **46** (1999), no. 5, 604–632.

[20] Kristina Lerman, *Advances in web mining and web usage analysis*, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 77–96.

[21] Mateusz Marmolowski, *Real-life rating algorithm*, 2008.

[22] P. Melville, R.J. Mooney, and R. Nagarajan, *Content-boosted collaborative filtering for improved recommendations*, Proceedings of the National Conference on Artificial Intelligence, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002, pp. 187–192.

[23] R. Nagmoti, A. Teredesai, and M. De Cock, *Ranking approaches for microblog search*, 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE, 2010, pp. 153–157.

[24] D.M. Nichols, *Implicit rating and filtering*, Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering, Citeseer, 1997, pp. 31–36.

[25] S. Nobarany, M. Haraty, and D. Cosley, *Geputtis: General purpose transitive trust inference system for social networks*, AAAI Spring Symposium on Social Information Processing, AAAI, 2008.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking: Bringing order to the web.*, (1999).

[27] Alexander Pohl, Vera Gehlen-Baum, and François Bry, *Introducing Backstage – A Digital Backchannel for Large Class Lectures*, Interactive Technology and Smart Education **8** (2011), no. 3, 186–200.

[28] Y.Y. Shih and D.R. Liu, *Hybrid recommendation approaches: collaborative filtering via valuable content information*, System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, IEEE, 2005, pp. 217b–217b.

[29] R.S. Wills, *Google's pagerank*, The Mathematical Intelligencer **28** (2006), no. 4, 6–11.

[30] J. Zhang, M.S. Ackerman, and L. Adamic, *Expertise networks in online communities: structure and algorithms*, Proceedings of the 16th international conference on World Wide Web, ACM, 2007, pp. 221–230.

List of Figures