FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

PhD Thesis

Stephan Sloth Lorenzen

Learning from Educational Data
Improving Methods and Theoretical Guarantees for Data Mining

Supervisor: Stephen Alstrup, Co-supervisors: Christian Igel, Mikkel Abrahamsen

August 2019 This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen.

Abstract. This thesis summarizes my PhD project. The structure of the thesis, and my project, follows the philosophy behind the Danish Center for Big Data Analytics driven Innovation (DABAI); in collaboration with companies, we develop solutions for educational data mining. Taking inspiration from the challenges faced, we define and investigate research problems within the areas of algorithms and machine learning.

During my project, I have worked with the Danish companies Clio and MaCom.

With Clio, the main objective has been to provide *teacher insight* about students in primary school. We do so through performance prediction in an online quiz system and by analyzing behavioral patterns observed in log data, in order to determine optimal study behavior.

With MaCom, we investigate methods for *detecting ghostwriters* in high school; external authors hired by students to write their essays. We extend this work to an analysis tool for *analyzing and tracking writing style changes* for high school students, providing insights for teachers.

Based on the problems faced while working with Clio, we develop novel techniques for improving *budgeted maximum inner product search*, an important algorithmic ingredient in many data mining methods.

Furthermore, we investigate *theoretical bounds for majority vote classifiers*, providing theoretical guarantees for the random forest classifier. While these bounds are often still too loose for practical uses, the area of research is important, as highlighted by our work with MaCom.

Finally, the thesis concludes with an overview of the company collaboration and a discussion of the challenges faced during the collaboration.

Abstract. Denne afhandling opsummerer mit ph.d-project. Strukturen af afhandlingen og mit projekt følger filosofien bag Danish Center for Big Data Analytics driven Innovation (DABAI): I samarbejde med firmaer, udvikler vi løsninger til dataanalyse inden for digital uddannelse. Baseret på de udfordringer, vi møder i dette samarbejde, opsætter og undersøger vi nye problemstillinger inden for algoritmik og maskinlæring.

I løbet af mit projekt, har jeg arbejdet med de danske firmaer Clio og MaCom.

Med Clio har vores hovedopgave været at finde metoder til at assistere folkeskolelærere, ved at give dem øget kendskab til deres elevers faglige evner og svagheder. Det gør vi ved at forudsige, hvordan eleverne klarer sig i online quizzer og ved, baseret på logdata, at analysere hvordan eleverne bruger systemet, og hvilken brug, der er optimal.

Med MaCom har vi undersøgt metoder til at spotte såkaldte *ghostwriters* i gymnasierne; eksterne skribenter hyret af studerende til at skrive deres opgaver. Disse metoder er blevet videreudviklet med henblik på at analysere og monitorere ændringer i skrivestil blandt gymnasieelever, som endnu en måde at danne indblik for lærerne.

Baseret på problemstillingerne ved Clio, udvikler vi nye teknikker, der forbedrer *maksimalt indre produkt søgning med begrænsede resourcer*, en vigtig algoritmisk ingrediens i mange metoder til dataanalyse.

Derudover undersøger vi teoretiske garantier for klassificeringsalgoritmer baseret på flertalsafstemninger, og vi præsenterer teoretiske garantier for random forest klassificeringsalgoritmen. Selvom garantierne endnu ikke er gode nok, er det et vigtigt forskningsområde, hvilket bekræftes i vores arbejde med MaCom.

Endelig konkluderes afhandlingen med en oversigt over samarbejdet med firmaerne, og en diskussion af de udfordringer, der er opstået under samarbejdet.

Contents

Chapter 1. Introduction

This thesis summarizes the work done during my PhD, starting September 1'st 2016 and ending August 31'st 2019. My PhD is funded by the ***Da****nish Center for* ***B****ig Data* ***A****nalytics driven* ***I****nnovation* (DABAI) [11] (described below in Section 1.1), and as such, the structure of my PhD follows the goals set out in this project. This has included a lot of work with different educational companies, as well as doing research based on problems identified with these companies; the overall structure is described in Section 1.2 below. The thesis consists of a collection of papers published during my thesis. The papers fall into two categories: *applied data mining* papers done in collaboration with the companies in the educational domain, and pure *research* papers. The papers and their main results will be introduced in the thesis, with the full papers being attached in Appendix A. A short overview of the included papers is given below with references to the appendix:

Applied data mining

A.1 S. Lorenzen, N. Pham, S. Alstrup: *On Predicting Student Performance Using Low-rank Matrix Factorization Techniques.* In *Proc. 16'th European Conference on e-Learning (ECEL)*, 2017.

A.2 S. Alstrup, C. Hansen, C. Hansen, S. Lorenzen, N. Hjuler, N. Pham: *DABAI: A data driven project for e-Learning in Denmark.* In *Proc. 16'th European Conference on e-Learning (ECEL)*, 2017.

A.3 S. Lorenzen, N. Hjuler, S. Alstrup: *Tracking Behavioral Patterns among Students in an Online Educational System.* In *Proc. 11'th International Conference on Educational Data Mining (EDM)*, 2018.

A.4 M. Stavngaard, A. Sørensen, S. Lorenzen, N. Hjuler, S Alstrup: *Detecting Ghostwriters in High Schools.* In *Proc. 27'th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2019.

A.5 S. Lorenzen, N. Hjuler, S. Alstrup: *Investigating Writing Style Development in High School.* In *Proc. 12'th International Conference on Educational Data Mining (EDM)*, 2019.

Research

A.6 S. Lorenzen, C. Igel, Y. Seldin: *On PAC-Bayesian Bounds for Random Forests.* In *Machine Learning, Volume 108, Issue 8-9*, 2019.

A.7 S. Lorenzen, N. Pham: *Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search.* In submission for *23'rd International Conference on Extending Database Technology (EDBT)*, 2020.

A short overview of all papers will be given in Section 1.2. A more thorough

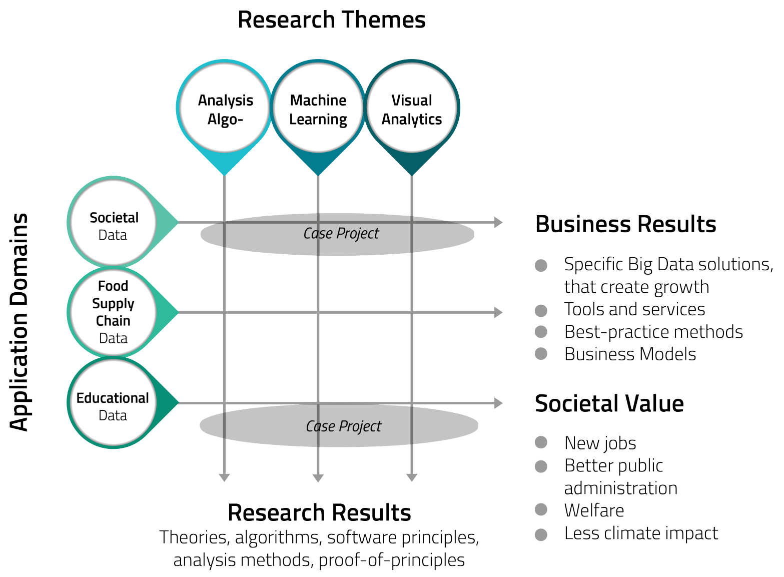


Figure 1.1: Overview of the research areas and the business cases of the DABAI project [11].

presentation and discussion of the **Applied data mining** papers is given in Chapter 2, with the exception of the paper *DABAI: A data driven project for e-Learning in Denmark*, which is discussed in Chapter 4. Chapter 3 presents and discusses the **Research** papers.

1.1. DABAI

The ***D**anish Center for **B**ig Data **A**nalytics driven **I**nnovation (DABAI)* [11] is a cooperation between the University of Copenhagen (UCPH), Aarhus University (AU), the Danish Technical University (DTU), and several companies and businesses in Denmark. The project is funded by Innovationsfonden.

DABAI consists of two parts: a practical/applied part, in which the universities help the companies solve their data analysis related problems, and a research oriented part, in which the universities, based on challenges faced in the practical part, formulate new theory and develop novel algorithms, which are then refined and publicized. Furthermore, each part is split into three. The scientific part is split between research in algorithms, machine learning and visual analytics, while the applied part is split between three different branches from Danish businesses: digital education, food supply chain and societal, see Figure 1.1.

UCPH is responsible for the work packages relating to algorithmic research and digital education, and thus this has been my focus in the PhD.

Three companies have been the main business collaborators in our branch: *Clio* [8], *EduLab* [15] and *MaCom* [29]. Of these, I have mostly worked with Clio and MaCom:

Clio [8] is the company behind the online learning platform with the same name, targeted at primary school. The system is used in most Danish primary schools, as well as other Nordic countries, having been used by more than 240,000 students. Clio consists of a large set of learning materials, including texts, exercises and auto-graded quizzes, and allows for the teacher to assign homework.

MaCom [29] supplies the lecture management system *Lectio* to more than 90% of Danish high schools. Lectio allows teachers to manage classes and teams and allows students to hand-in their assignments digitally. As a result, MaCom has access to a huge number of high school assignments written by Danish high school students.

As members of DABAI, Clio and MaCom formulated some high-level problems, which they or their users have encountered. These problems, described below in Section 1.2, have been the basis for starting the applied/practical part of my PhD, while some of the problems identified in this part, has formed (either directly or indirectly) the basis for the more theoretical part of my project. In short, my focus has been two-fold: assisting suppliers of digital educational tools with analyzing their data, and, based on the challenges faced in this task, to develop novel, efficient data mining/machine learning techniques with a focus on Big Data.

1.2. PhD Overview

As mentioned, my PhD follows the project description for DABAI. It consists of two parts: working on data analysis problems with the companies Clio and MaCom, and, with the gained knowledge from these cases, working on some more general underlying problems.

My main focus has been data analysis, with an extra focus on large amounts of data. More and more data is created and collected every day; it is expected, that 175 zeta bytes will be generated in 2025, compared to 33 zeta bytes in 2018 [35]. This enormous increase in data comes with a huge potential, but in order to realize this potential, we need efficient algorithms. Furthermore, with machine learning used for assisting in making more, potentially important, decisions, we also need careful analysis of said algorithms in order to guarantee that they perform as expected.

These two questions: more efficient data analysis and obtaining performance guarantees, are the main topics of my project, and are inspired by my initial work with Clio and MaCom respectively.

1.2.1 Teacher Insight and Efficient Recommender Systems

Our main collaboration with Clio has been centered around their quiz system, in which a student takes a quiz and gets a score, which is then converted to a grade on the Danish grade scale. The system has been used for several years; with more than 240,000 unique users and more than 10,000 quizzes in the system, Clio has collected several million records of answered quizzes with accompanying scores.

The main task we have been working on in this domain, is to predict the score achieved by a student on an hitherto unseen quiz. Such kind of predictions may assist the teacher in several ways, for instance in selecting the correct quiz for the class or in differentiating assigned time for the quiz between different students.

Our initial approach was based on recommender systems, as used for example in the Netflix price competition [21]. In this setting, one may equate the score of the quiz with the rating of an item in the recommender system, and the skills of a student can be modeled in a similar fashion to the preferences of a user. Using matrix factorization techniques, we achieve a fairly high accuracy, as presented in [28].

As mentioned, the Clio system also contains a lot of other learning materials, such as texts and exercises. In order to improve our predictions and be able to provide even better feedback to the teachers, we apply again the matrix factorization techniques to a matrix containing temporal profiles for the students, i.e. activity profiles generated for students for a fixed period of time, consisting of aggregates over system accesses, time spent reading, etc. The output is a soft clustering of the activity profiles. Analyzing the clusters allows us to make qualitative statements about them, including which clusters are more likely to score better in quizzes. With development profiles for students in every time period, the method also allows us to track how the activity of students change during their time in primary school. This work was presented in [24].

Applying the methods and analyzing the large amount of data made available by Clio highlighted the need for algorithms with capacity for handling large amounts of data, especially as one requirement from Clio is that a quiz prediction query must be answered quickly. Specifically, Clio had demands on the amount of time allowed for answering a query; as such, the prediction algorithm required an explicit way to set the time allowed for a query. Hence we looked into the *budgeted maximum inner product search*. Maximum inner product search is a key component of many recommender system algorithms, such as item similarity search and matrix factorization, and indeed in many other machine learning algorithms. The budgeted version accepts an explicit budget on the number of computations allowed, at the cost of higher failure probability (i.e. not getting the actual maximum inner product), giving us an explicit way to limit the amount of time needed. Our work on this topic is presented in [27], currently in submission for EDBT 2020.

Our data analysis tasks related to predicting scores and analyzing student behavior is described in more detail in Section 2.1, while the follow-up work on improving recommender systems is described in Section 3.1.

1.2.2 Writing Style Analysis and Trusting Machine Learning

The focus with MaCom has been on *student writing style* in Danish essays, with the end goal being to provide insight to the teacher about the development of the writing style of a student. This is a natural area of interest, as developing the writing style is a main task in Danish secondary education. However, our initial research into tracking writing style was motivated by the increase in use of *ghostwriters* in Danish high school. In this setting, a ghostwriter is a person (likely to be external to the high school), who is hired by a student to write an assignment for them. It may be an older high school student, a university student or a teacher, and the ghostwriter may be hired through online services. In the literature, the problem is also known as *authorship verification.*

The use of ghostwriters has become an increasing problem in Danish high schools, especially for Studieretningsprojektet (SRP), a large, interdisciplinary, written assignment, which high school students have to complete during their third and final year, and which counts double towards the grade point average [47]. We investigated several approaches to solving the ghostwriter detecting problem based on MaCom's textual data. The best results were achieved using a *Siamese neural network* to encode and compare texts, beating results obtained by the more traditional authorship verification methods. These methods were initially tested as part of two master's projects, before the results were finalized in our paper [42]. The results were encouraging and paves the way for implementing the software, adding to the plagiarism detecting software already in place in Lectio.

From the models trained to solve the ghostwriter problem, we also made observations about the development over time of writing style among high school students. This resulted in a spin-off study of writing style development, which was published at the international conference for *Educational Data Mining* in 2019 [25]. This study investigates the overall patterns for development of writing style in high school and correlates different development profiles with development in writing quality and can potentially be used to detect suboptimal development. Furthermore, our models allowed us to analyze how similarities between different students change over time, providing insights about the general development trends during high school.

Working on the initial project to detect ghostwriters presented us with another problem: being falsely accused of cheating in the SRP can have serious consequences for a student. As such, we have to be very confident in the accuracy of the algorithm (which, even then, should not be used by itself, see also the discussion in Section 4.2). Our proposed method allows for limiting the number of false accusations by adjusting a threshold, but this in itself gives no theoretical guarantees.

This indirectly motivated an investigation into such theoretical guarantees in the form of *PAC-Bayesian bounds* on the risk of a classifier. However, our study in this regard is not directly applicable to the final ghostwriter software, as we investigated PAC-Bayesian bounds for *majority vote classifiers*, specifically random forests. We did evaluate random forests for the authorship verification task, but the final model found was not very successful and was abandoned in favor of the deep learning approach.

However, our results on applying these theoretical guarantees to random forests trained using *bagging* showed, how good guarantees can be had "for free", in the sense that no validation set is required (this extends to any algorithm trained using bagging) [26]. Furthermore, our experiments indicated the strength and usefulness of some of the state-of-the-art bounds in this setting. And while not applicable in our current ghostwriter detection approach, a future model is likely to be based on an ensemble, in which case the bounds may be applicable.

The ghostwriter project and the writing style analysis are described in more detail in Section 2.2. The work relating to PAC-Bayesian bounds for majority vote classifiers are described in Section 3.2.

1.3. Road map

My thesis is structured according to the DABAI setup. Thus, Chapter 2 gives an overview and discusses the new results achieved in educational data mining. The section will discuss the results from [24, 25, 28, 42], including a few other relevant results.

Chapter 3 presents the new theoretical and empirical results obtained, motivated by our work in educational data mining. The results from [26,27].

Chapter 4 discusses our collaboration with the companies, the challenges faced, and the potential for the companies and for society in terms of improved learning and evaluation of students, drawing on the conclusions of [1].

Finally, Chapter 5 makes some concluding remarks on the thesis.

Chapter 2. Data Analysis in Online Education

This section discusses the results obtained in educational data mining and analysis during our collaboration with Clio and MaCom.
Section 2.1 describes the research done in collaboration with Clio, which is focused around providing teacher insight, through predicting student performance and by profiling the students in terms of performance and behavior in the system. The section presents our results in predicting quiz scores [28], and our investigation of the correlation between performance and behavior [24]. The section is similar to the content of those papers and may contain passages from the papers without quotation.

Section 2.2 discusses the results obtained with MaCom, which, as mentioned, is focused on the writing style of the students. The results regarding detection of ghostwriters is described and discussed [42] (including some results from an earlier master's thesis on the subject), before the spin-off study regarding tracking of writing styles [25] is presented. The section is similar to the content of those papers and may contain passages from the papers without quotation.

2.1. Providing Teacher Insight in Primary School

Insights about students' skills and study habits can be very valuable for teachers in primary school. While most teachers probably know their students well, the teachers knowledge will be "local" in nature, limiting itself to the students and material, that the teacher knows. Hence, new and improved insights may be found by analyzing student data across classes and even across schools; data, such as that available to Clio.

2.1.1 Predicting Student Performance

One way to provide insights to the teacher is through performance prediction. Thus, we have looked into predicting the grades of students in the Clio quiz system. Our work is presented in the paper *On Predicting Student Performance Using Low-rank Matrix Factorization Techniques* [28], see Appendix A.1.

The problem of predicting the performance of a student (as the answer to a question, the score on a quiz, a final grade, or some other metric) has seen a lot of interest in the educational data mining community [14, 31]. There are several motivating factors for measuring and understanding the skills of a student. It is valuable insight to the teacher, and may help the teacher provide individualized learning (in terms of allotted time and difficulty of learning material), or identifying at-risk students early.

Predicting grades in the Clio quiz system has been a primary objective in our work with Clio. The system contains quizzes consisting of various question types such as multiple choice, pair matching, or fill-in-the-blanks. Each quiz has a maximum number of points obtainable, and, on answering a quiz, a student is awarded some number of points between zero and the quiz maximum. The points are then converted to a grade on the Danish 7-grade scale. This is the grade we wish to predict and present to the teacher. However, for the work discussed here, we have instead normalize the points to a score in $[0, 1]$; converting back is easily done.

The data made available by Clio consists of answered quizzes and access logs for different learning materials, such as texts read, videos watched and exercises performed, making the data suitable for the task.

In [28], we consider only the historical quiz data for the prediction, i.e. letting $S = \{s_1, s_2, ..., s_n\}$ denote the set of students and $Q = \{q_1, q_2, ..., q_m\}$ denote the set of quizzes, we are given an incomplete matrix $\mathbf{X} \in [0, 1]^{n \times m}$, where $\mathbf{X}_{ij}$ is equal to the score obtained, if student i has taken quiz j, and missing otherwise. The objective is to determine the values of the missing entries in $\mathbf{X}$.

The problem can be viewed as a collaborative filtering problem, which is more often associated with rating of items by users, such as movies [21] or music [20], but with a rating substituted for a score, and user preferences substituted for student skills. Thus, we consider methods from collaborative filtering for solving the problem; specifically, we consider *matrix factorization (MF)* [22]. Using MF, the given matrix $\mathbf{X}$ is factorized, so that we obtain $\mathbf{Q} \in \mathbb{R}^{n \times k}$ and $\mathbf{P} \in \mathbb{R}^{k \times m}$ for some parameter k, such that $\mathbf{X} \simeq \mathbf{QP}$. The parameter k describes the number of *latent variables*. In problems relating to movie rating, these variables have been linked to user preferences, while the natural link in our case is student skills.

Aside from considering unrestricted MF, we also consider *non-negative matrix factorization (NMF)* in which $\mathbf{Q}$ and $\mathbf{P}$ must be non-negative, due to it being naturally regularized by the non-negative constraint, as well as the intuitive interpretation of the solution found [21].

As $\mathbf{X}$ is incomplete, the factorization is computed by an *Expectation-Maximization (EM)* procedure [28, 40], in which new solutions $\mathbf{U_i}, \mathbf{V_i}$ are found (expectation step) based on $\tilde{\mathbf{X}}_{i-1}$, obtained by imputing missing values of $\mathbf{X}$ with the best previous solution $\mathbf{U}_{i-1} \cdot \mathbf{V}_{i-1}$ (maximization step). Given imputed matrix $\tilde{\mathbf{X}}_{i-1}$, the solution $\mathbf{U}_i, \mathbf{V}_i$ is obtained by a *truncated (non-negative) single value decomposition* [3].

The results obtained using NM and NMF are compared to a baseline, where the prediction $B(i, j)$ for student i taking quiz j is computed as a simple linear combination of the global mean score μ, the *student bias* α_i for student i, and the *quiz bias* β_j for quiz j [21, 28]:

$$B(i, j) = \mu + \alpha_i + \beta_j,$$

where the student bias is given by $\alpha_i = \mu_{s_i} - \mu$, μ_{s_i} being the student mean

Method	k	**RMSE**
Baseline	-	0.1768
MF	1	0.1751
NMF	2	**0.1744**

Table 2.1: Results obtained using matrix factorization compared to the baseline.

score, and similar for the quiz bias β_j. This baseline is also used to impute the initial matrix $\tilde{\mathbf{X}}_0$ for the EM procedure.

The data supplied by Clio was cleaned for the experiment, consisting of removal of incomplete and duplicate answers, and answers from test users, leaving a dataset with approximately 167,000 students, 6,290 quizzes and 1.4 million answers. However, many of the students and quizzes were *inactive* in the quiz system, in the sense that they had only few answers. For instance 40% of students had answered less than 5 quizzes. Thus, the data was densified by continuously removing students and quizzes with less than 15 answers, leaving us with a final dataset consisting of $n = 1141$ students and $m = 245$ quizzes. Table 2.1 presents the results obtained on this dataset, where the error is computed as the *root mean square error* between $\mathbf{X}$ and the found solution $\mathbf{U} \cdot \mathbf{V}$. For regular MF, $k = 1$ was found to be optimal, while $k = 2$ was found to be optimal for NMF [28]. As can be seen, NMF achieves the best result.

It is worth noting, that while NMF does not beat the simpler baseline by much, the model can be used to provide insight about the students. The fact that k is small for both MF and NMF, i.e. we get a low dimensional factorization with only few latent variables, indicates that the performance of a student on a given quiz is not well described by many different types of skills. This was unexpected, as we had hoped to observe at least a few different skills. The results is further confirmed by computing the eigen spectrum of $\mathbf{X}$ [28].

The results presented in [28] and discussed above were all performed on a small, densified subset of the quiz dataset. Further experimentation showed, that the methods would usually degrade on more sparse data. As can be seen from Figure 2.1, most users are not very active, with more that 40% answering less than 5 quizzes (from more than 6,000 quizzes in total), meaning that the data set is indeed very sparse. Furthermore, the matrix factorization techniques were also limited by available memory and computational power. Hence, we implemented hybrid versions, which applied (N)MF to the most active users and quizzes, while using the baseline for less active users and quizzes, allowing Clio to obtain predictions for all students. The results for the hybrid NMF are shown in Table 2.2. While these results have

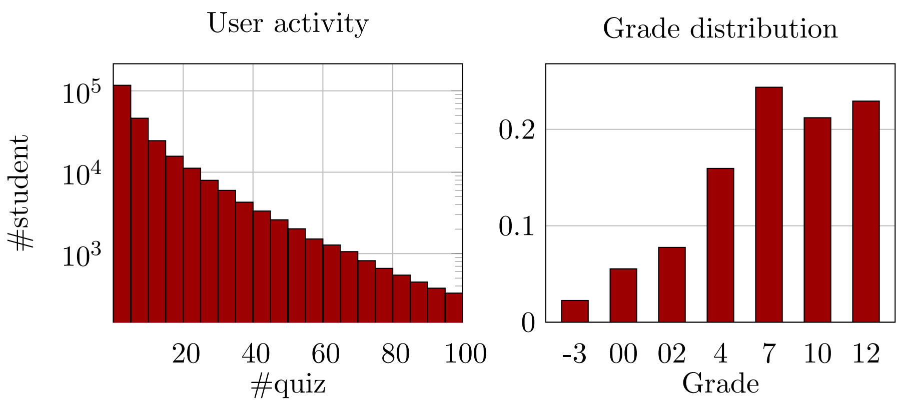


Figure 2.1: Statistics for the Clio dataset. **Left:** distribution of students according to activity. First bar is students with 0-5 answered quizzes, second bar is 5-10, and so on. Note, the y scale is logarithmic. **Right:** distribution of grades in the dataset.

RMSE	**Acc@1**	**Acc@2**	**Acc@3**
0.1933	0.4061	0.5674	0.8338

Table 2.2: Results obtained on the full Clio dataset using the hybrid NMF ($k = 1$), combining NMF with the baseline. The table contains the RMSE and accuracy when predicting one (Acc@1), two (Acc@2) or three (Acc@3) grades.

not been publicized, the method was incorporated in the final prediction framework developed with Clio, as described in Section 4.1. Hence, the table also includes the accuracy Acc@x for predicting the correct grade when allowing the algorithm to return a set of x grades, as these are relevant to the deployment of the solution. We determined $k = 1$ to be optimal for NMF in this extended dataset.

We also continued experimentation with other approaches. One approach considered was based on quiz similarity, and predicted scores as a linear combination of the student's score on other quizzes, weighted by the quiz similarities. This has been done successfully in other recommendation tasks, but unfortunately we were unsuccessful in making it work well for the Clio case. However, as a result of our experimentation with that method, we started looking into efficiently computing the *budgeted maximum inner product*, as described in Section 3.1.

A drawback of the solutions presented in this section, is the fact that they rely only on the data from the quiz system, while much more data is available in the Clio system, such as time spent on the quiz and access logs for other learning materials. Thus, we started investigating the gains from using the activity log data from the system as well.

2.1.2 Tracking Student Behavior

As a further mean to provide teacher insight and in order to hopefully improve the quiz grade prediction, we investigated the activity log data provided by Clio. In our paper, *Tracking Behavioral Patterns among Students in an Online Educational System* [24] (see Appendix A.3) we investigate these logs. The main objective was to determine optimal student behavior, where optimality was considered as improving grades obtained in the quiz system. Furthermore, we also investigated the relation between subjects and complexity of exercises, where complexity is measured by Bloom's taxonomy.

For both experiments, we defined a set of non-negative activity measures, for example *hours active in the system between 8AM and 4PM*, *hours spent reading texts*, etc. We then traversed the log and computed these measures for fixed time periods, i.e. for every period, an *activity vector* consisting of these measures was computed for every active student within that period. The length of the period was set to 1 week. Processing the log in this way, we obtained a large matrix $\mathbf{X} \in \mathbb{R}_+^{n \times m}$, where each of the n rows corresponded to the activity vector with m measures for an active student in some time period. Thus, each student would have several rows; one for each period, in which they were active.

A soft clustering of the activity vectors were constructed using non-negative matrix factorization, i.e. we obtain student matrix $\mathbf{U} \in \mathbb{R}_+^{n \times k}$ and cluster matrix $\mathbf{V} \in \mathbb{R}_+^{k \times m}$, such that $\mathbf{X} \simeq \mathbf{U} \cdot \mathbf{V}$. The student matrix $\mathbf{U}$ describes which clusters a student in a given period belongs to, while $\mathbf{V}$ describes how the clusters relate to the measures. Thus, the rows of $\mathbf{U}$ for a single student allowed us to track how the student changes clusters, while $\mathbf{V}$ allowed us to analyze the clusters.

As mentioned, we performed two different experiments with two different sets of activity measures [24]. The first experiment, trying to determine optimal student behavior with respect to quiz grade obtained, is shortly summarized here. By use of the 'elbow method' [45], we determined $k = 5$ to be optimal. The transposed cluster matrix $\mathbf{V}^T$ from the experiment is shown in Figure 2.2.

The cluster matrix shows the correlation between each of the activity measures and the performance, given by the average score (bottom row). One can make the following observations:

– Students working with science (C_1) and societal (C_3) subjects appear to benefit more from reading texts.

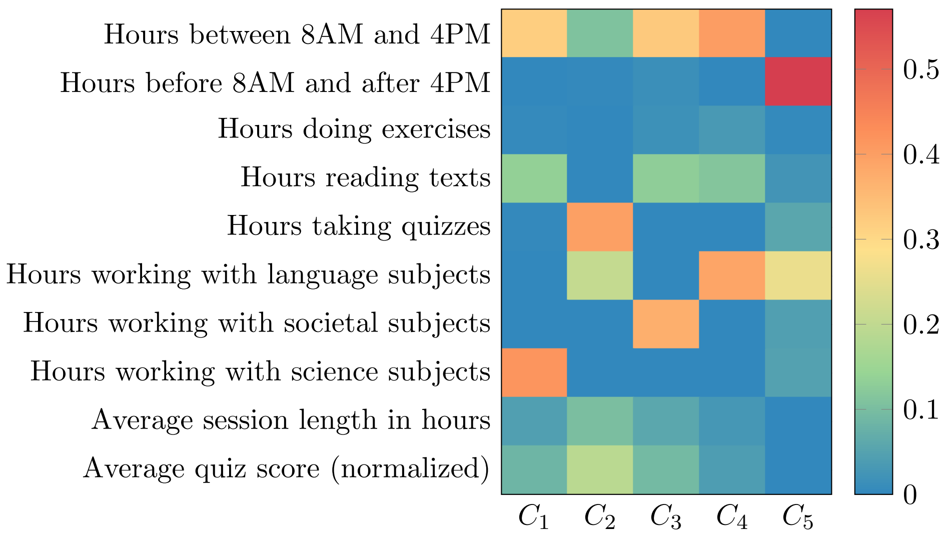


Figure 2.2: The cluster matrix obtained, transposed.

– In contrast, students working with languages (C_2) benefits more from taking quizzes, while less from reading texts (C_4).

– Working outside of school hours (4PM to 8AM) (C_5) does not seem to have any effect on performance.

A full description and analysis of the five clusters are given in the paper [24].

As mentioned, we can track how individual students moves between clusters by inspecting the corresponding rows of $\mathbf{U}$. Single students cannot be highlighted due to privacy concerns, but Figure 2.3 shows instead the average cluster membership for each period. From the figure, C_1, C_2, C_3 and C_4 appear correlated at the system-wide level. This is due to these clusters being dependent on the general activity in the online system, i.e. drops in membership occur during school vacations. C_5 appears to be the exception, as it is the “work from home” cluster.

Regarding the use of log data for predicting quiz grades, the experiment presented here does show a correlation between student activity in the system and performance, indicating that including activity measures will improve the score prediction. However, while we did experiment with several methods for prediction, which included the log data as input, we did not managed to beat the methods based solely on the historical quiz data.

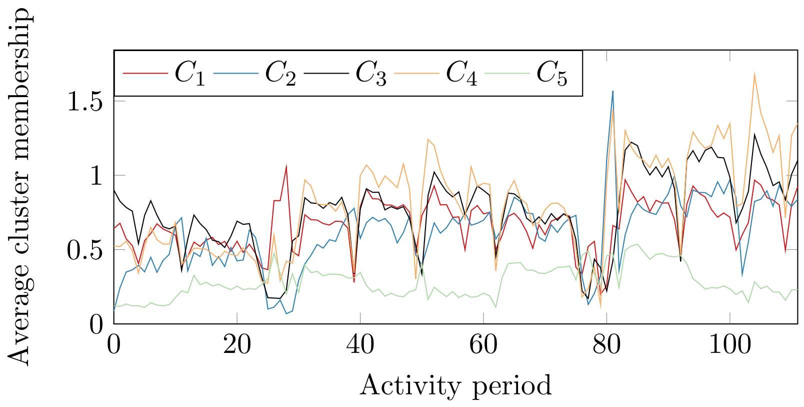


Figure 2.3: Average cluster membership in each period.

2.2. Writing Style Analysis in High School

Writing style, and especially good writing style, is a hard concept to define. Some studies link writing style to readability [33], i.e. "simple" writing style may be first grade level, while "better" writing style may be at a ninth grade level. In discussing writing style, we are not thinking of grammatical errors, but rather vocabulary, sentence structure, etc. Such stylistic markers provide a finger print for the author, allowing us to identify the author by the writing style. Furthermore, tracking changes in the writing style can be used as valuable feedback for teachers.

2.2.1 Detecting Ghostwriters

Our initial work with MaCom dealt with the problem of *detecting ghostwriters*. The hiring of ghostwriters for large written assignments has been an increasing problem in Danish high schools. A large part of the grade point average for a Danish high school student is made up of grades based on written home work, especially the *Studieretningsprojekt (SRP)*, an interdisciplinary paper the students have to complete in their final year, the grade of which counts double towards the grade point average.

The use of ghostwriters have seen an increase in the last few years, with new websites offering ghostwriter services appearing [47]. Thus, together with MaCom, we started investigating the problem. MaCom, through Lectio, already provides tools for checking for *plagiarism*; however these tools only detect plagiarism when passages of text are copied from other, known sources. Since a ghostwriter will produce new, original and unseen text, the plagiarism tools will not detect this kind of cheating.

Hence, in [42] we looked into detecting ghostwriters by analyzing the writing style used in texts handed-in by students. The reverse problem is known in the literature as *authorship verification*: Given author α with

Method	**Type**	**Accuracy**
Distance (V)	Author specific	0.629
SVM (V)	Author specific	0.598
Distance (A)	Semi-generalizing	**0.726**
SVM (A)	Semi-generalizing	0.642
Random forest	Generalizing	0.670

Table 2.3: Results obtained in the first master project [19]. (V) denotes the verification version of the method, while (A) denotes the attribution version.

known texts $t \in T_\alpha$ and unknown text x, determine whether α is the author of x [41].

Furthermore, in the case of our work with MaCom, we had access to not only the texts of α, but indeed an entire corpus of texts, T, i.e. texts $\overline{T_\alpha} = T \setminus T_\alpha$ not written by α are also available. These can be utilized as examples of different writing styles, when training a model. In our case, we considered only Danish essays; MaCom supplied us with access to approximately 130.000 such essays from approximately 10.000 students.

The project was started as a masters project [19] (co-supervised by me), in which the master student compared some traditional methods with some state-of-the-art methods for authorship verification. In the project, *author specific* models (using only texts from T_α) were compared to *(semi-)generalizing models*, which utilize some or all of the data from T. Specifically, the student tested outlier methods based on distance computation between feature vectors constructed from the texts, as well as SVM based outlier detection. These methods were further modified by sampling "false authors" from T and solving the related problem of *authorship attribution* [18, 41]: selecting the correct author from a set of authors, and answering the verification query positively, if the correct author was identified. This trick allowed for including more data from T, providing examples of different writing styles. Finally, the methods were compared to a fully generalizing model, the generalizing forest [32]. The methods were tested on a balanced dataset, and the results are shown in Table 2.3.

As can be seen from the table, the (semi-) generalizing methods perform better. Motivated by these results, the project was continued in another master thesis [43] (again, co-supervised by me), which investigated deep learning based, generalizing approaches for authorship verification. Our final approach, described in the paper *Detecting Ghostwriters in High Schools* [42] (see Appendix A.4), is based on the ground work from this thesis.

Our approach for solving the authorship verification consists of two steps:

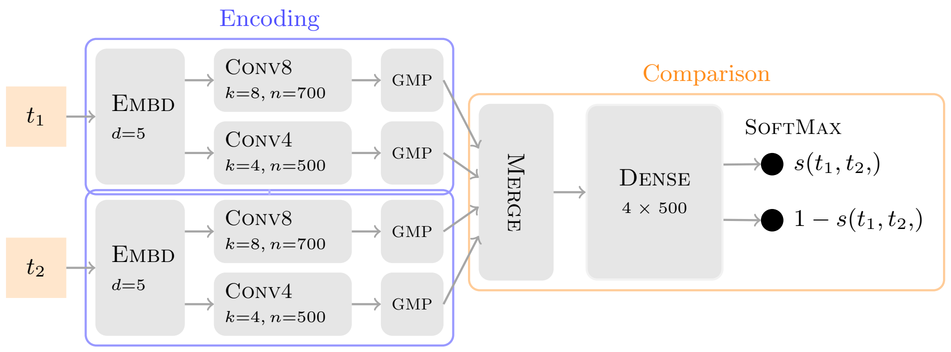


Figure 2.4: The Siamese network used to compare writing style [25, 42].

a text similarity computation using a *Siamese neural network* and a combination phase, in which several similarity computations are combined. A Siamese neural network consists of two equal *encoder* networks with shared weights, which encode two inputs (in our case, two texts), before the encodings are compared in a final *comparison* part of the network (see also Figure 2.4). This approach to computing text similarity (for instance in terms of writing style) is well known in the literature [36, 39]. However, our study is the first in the domain of high school essays and using Danish texts, and as such, a new network structure was needed, as shown in Figure 2.4.

The network solves the problem of writing style similarity between texts: given two texts $t_1, t_2 \in T$, it computes $s(t_1, t_2)$. Now we consider an author α with texts T_α and an unknown text x. For each $t \in T_\alpha$, we compute the similarity $s(x, t)$. A final similarity between x and the author α is then computed as a weighted average of these similarities, taking into account the hand-in time of each known text t relative to x. A decision is made by comparing the computed, final similarity to a configurable threshold.

This approach is generalizing, as no model needs to be trained for each specific author; rather the network and combination strategy are trained on the entire training set, while a separate set of authors is set aside for evaluating the final model. Another advantage of the network, is that it learns the important features in the convolutional layers of the encoding sub-network during training, compared to many of the traditional approaches which require excessive feature selection [7, 41].

We compared our proposed model to author specific SVMs [18] and Burrows' delta method [7] on a balanced dataset (both methods use the trick with "false authors" mentioned above). Our approach is best among the three by a large margin, as shown in Table 2.4[1].

[1] Note, that the dataset used in this experiment is larger than the one from the first thesis.

Method	**Accuracy**	**FAR**	**CR**
Burrows' delta method	0.677	0.357	0.806
Author specific SVMs	0.720	0.266	0.689
Our approach	0.875	0.141	0.896

Table 2.4: Results obtained for detecting ghostwriting in a balanced dataset.

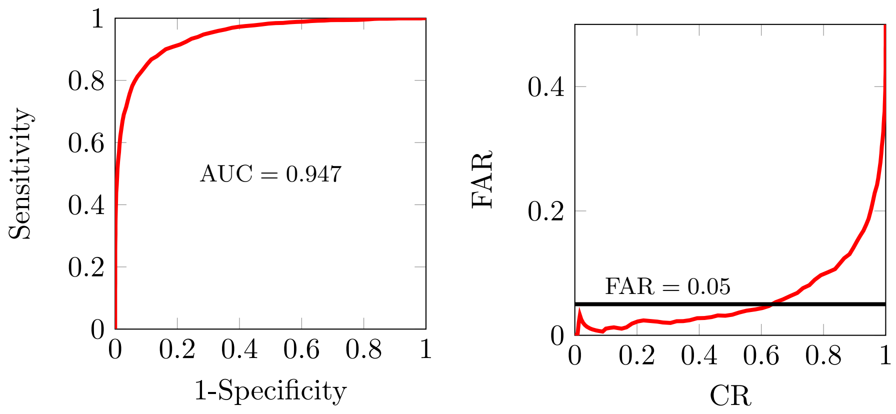


Figure 2.5: Plot of the ROC (left) and the false accusations rate as a function of catch rate (right).

When evaluating the final model, we also had a significant focus on the number of false negatives (FN), i.e. the number of false accusations. We define the *false accusation rate (FAR)* and the *catch rate (CR)* as follows:

$$\mathrm{FAR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} \qquad \mathrm{CR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

i.e. the fraction of false accusations and the fractions of ghostwriters caught, where TN denotes true negatives and FP denotes false positives.

We are particularly interested in these metrics, since falsely accusing a student of cheating is very problematic, as the consequences for said student can be quite severe. From Table 2.4, we see that we get a false accusation rate of 14.1% in a balanced dataset. However, at the cost of some accuracy, this can be directly reduced by decreasing the threshold for accepting an author (another advantage over some of the traditional methods, where the number of false negatives may only be adjusted in a more indirect manner).

Figure 2.5 shows a plot of the ROC (left) and the false accusation rate (right). As can be seen, we can achieve a low false accusation rate (< 0.05) while still maintaining a catch rate above 0.6. Furthermore, as seen from the ROC, we obtain a high AUC $= 0.947$. The problems regarding the false accusation rate are further discussed in Section 4.2, which also discusses the possible ways of deploying the system.

Another take-away from our authorship verification experiments is the optimal strategy found for combining the similarities [42]. As mentioned, the optimal strategy computes a weighted average of $s(t, x), t \in T_\alpha$. The optimal weight assigned to each similarity was found to be exponential decreasing in the time of hand-in between t and x, indicating that the writing style of a Danish student changes during high school. While not surprising, we investigated this further as a means to provide teacher insight.

2.2.2 Tracking Changes in Writing Style

In the paper *Investigating Writing Style Development in High School* [25] (see Appendix A.5), we utilized the similarity function $s(t_1, t_2,)$ computed by the network described in the previous section, in order to track the *writing style changes* for individual students during high school.
While the focus was mainly on the change in writing style occurring among groups of students in Danish high school, evaluating the *quality* of the writing style was a secondary objective.

Whereas the focus of the Danish primary school in the Danish subject is for students to write correctly without grammatical errors, students in high school can focus more on improving the writing style instead. Good writing style is hard to define, but it has been linked to readability [33], for which many empirically supported measures have been defined [30]. However, using the network of Figure 2.4 and training it in a similar fashion, allowed us to track the changes in the writing style of a single student, by comparing their later essays to their earlier essays.

The problem of tracking writing style changes has only seen limited interest in the educational data mining community, although the benefits are large. As done more directly in our previous work [42], one may detect fraudulent behavior, but one may also be able to detect deviating development, which could indicate suboptimal development of a student at risk.

In [25], we train the network in Figure 2.4 by supplying the text pair examples as before. The network is then used to analyze a separate set of students. For each student α in this set with texts $T = \{t_1, t_2, ...\}$ (chronologically ordered), a writing style development profile P_α is generated as a sequence of pairs (τ_i, p_i) where $\tau_i = \tau_\alpha(t_i)$, $\tau_\alpha : T_\alpha \to \mathbb{R}_+$ giving the number of months between the hand-in of $t_i \in T_\alpha$ and $t_1 \in T_\alpha$, and $p_i = s(t_1, t_i)$. Here, t_1 is used to determine the initial writing style of student α; however, this was found to be fragile, as t_1 may be an outlier, and instead, we settled on using the $k = 2$ first texts of α as the initial writing style and removing the first $k - 1$ data points from P_α (meaning p_i is an average of k similarities and τ_i is the months between the first data point based on t_k and t_{i-k+1}) [25]. Note, that P_α describes a curve.

We then clustered these profiles by applying a modified version of k-means

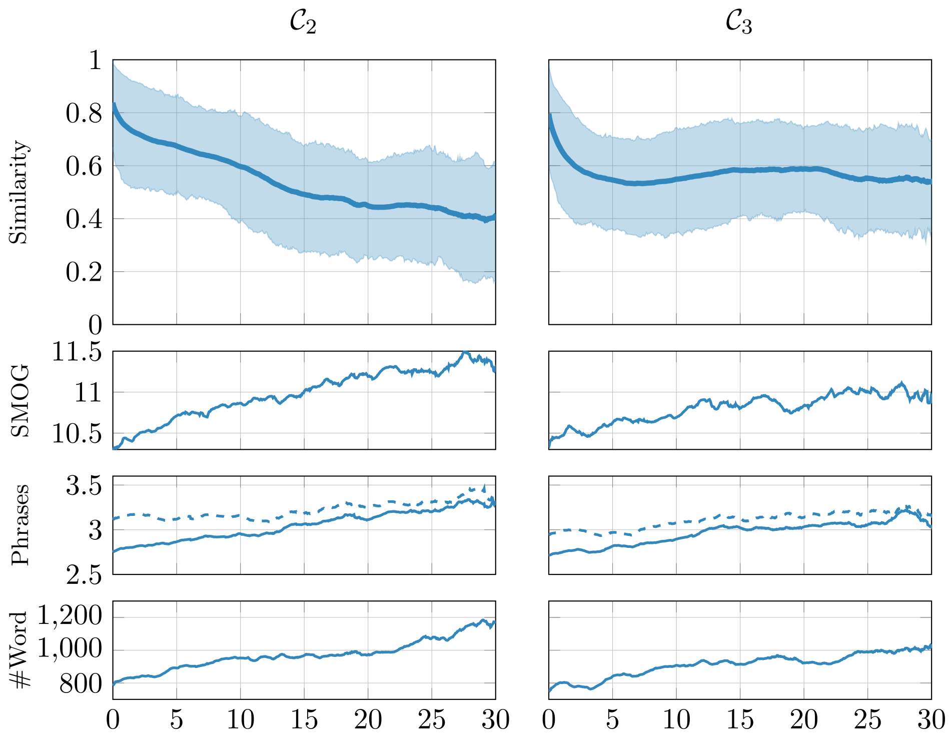


Figure 2.6: Clusters C_2 and C_3 found in the analysis set [25]. The plots run over a 30 month period, with the top plot showing the change in similarity, the second and third plots showing writing style quality indicators, and the fourth plot showing average text length.

clustering, which could handle the profiles having different lengths (since not all students have handed-in during the same time frame). By use of the “elbow” method [45], we found five clusters to be optimal. For each of these five clusters, we computed average length and some indicators of writing style quality: the *simple measure of Gobbledygook* (SMOG) grade, and noun and verb phrases [30].

From our analysis, we found two clusters, where the cluster center indicated near-optimal development, one cluster with fair development, and two clusters with clearly sub-optimal development profiles. Figure 2.6 shows example clusters C_2 and C_3 found in the analysis, which exhibit near-optimal and sub-optimal development respectively. Below follows a brief discussion of these cluster, while the full discussion is given in the paper.

For C_2, the initial value of approximately 0.84 indicates an initial low variance in writing style, but the following drop to about 0.4 is quite significant, indicating that the writing style of students in this cluster change a lot

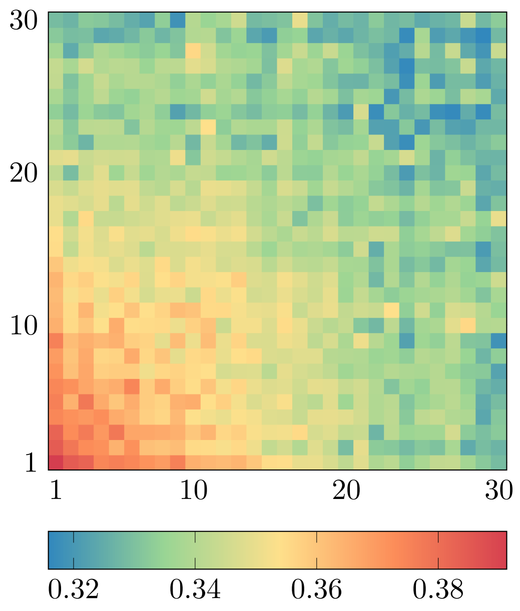


Figure 2.7: Average similarity between random students as a function of the time each student has spent in high school.

during high school. While the drop is significant across the 30 months, the rate is fairly constant, indicating stable learning during the period. Considering the other metrics, we see a large increase in SMOG grade, with modest increases to noun and verb phrases, indicating a good development of writing style among students in C_2.

Comparing this to C_3, we also see an initial drop in similarity, but here the drop is followed by an increase after the first year, showing that the students in this cluster actually revert to a writing style more similar to their original work. This may indicate "forgetting" learned skills or possibly reverting from a bad habit; in either case, the result is a smaller improvement in SMOG grade compared to C_2, indicating sub-optimal development.

Aside from inspecting the clusters, one may also inspect the development profiles of individual students, especially students who do not fit well in any cluster, in order to detect other, deviating development profiles.

We also investigated how the similarity develops between different students in different stages of high school. This was done by sampling random pairs of essays from different students, computing their similarity, and plotting the averages in a heat map as a function of the time spent in high school by the students. The result can be seen in Figure 2.7.

As can be seen from the figure, surprisingly, the writing style of students become less similar, as they progress through high school, although the overall similarities are still fairly small. An explanation could be that the writing styles of students are aligned when leaving primary school. The observed pattern, where students develop towards more unique writing styles as they become more educated, is probably preferred compared to the alternative, where education makes student more alike each other.

Whether this pattern also occurs in primary school and higher education could be an interesting topic to investigate further.

Chapter 3. Improved Data Mining and Machine Learning Techniques

This section describes some challenges identified during the work with the companies, which we investigated further, in order to improve the general methods. Specifically, this section will describe my work with *budgeted maximum inner product search* [27] and with *generalization bounds for random forests* [26]. The section is similar to the content of these papers and may contain passages from the papers without quotation.

3.1. Better Recommender Systems by Budgeted Maximum Inner Product Search

Some of the most competitive algorithms used in collaborative filtering based recommender systems are based on finding the maximum inner product of a set of vectors. In *maximum inner product search (MIPS)*, we are given a set of vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subset \mathbb{R}^d$ and a query vector $\mathbf{q} \in \mathbb{R}^d$, and we wish to determine the element $\mathbf{x}_i \in X$ such that $\mathbf{x}_i = \operatorname{argmax}_{\mathbf{x}_i \in X} \mathbf{x}_i \cdot \mathbf{q}$. The related variant k-MIPS denotes the problem, in which we wish to determine the k elements of X having the top-k inner products.

Aside from being an important algorithmic ingredient in a variety of machine learning tasks, such as multi-class prediction [12, 37] and neural networks [10], MIPS is the backbone of many collaborative filtering based recommender systems, in which a large inner product between user and item vectors indicates that the item is relevant to the user and should be in the recommendation list.

Several modern real world online recommender systems, e.g. Xbox or Netflix, relies on MIPS, and often deal with very large-scale datasets and limited amount of response time [20,21], showing the need for fast algorithms for finding these large inner products. This was also the case in our work with Clio, in which the best performing algorithms were based on matrix multiplication and finding items of the highest similarity, motivating further research into this topic.

In many use cases, Clio included, the recommendation is often performed in the *online* manner since the user vector is updated online with ad-hoc contextual information only available during the interaction [20, 22]; relevant recommendations may need to be presented to the user, at the press of a button or as they input their preferences. Furthermore, as user preferences or implicit history changes over time, recommender systems need to frequently update the model to address this *drift* in user preferences.

Often, "non-perfect" results are acceptable, when retrieving the recommended items, if the retrieval is "fast". However, the results should improve in terms of accuracy/relevance, as more time is used for retrieval. In many use cases, it is advantageous to be able to explicitly control this time/quality trade-off. Motivated by this need, we have investigated the *budgeted* MIPS problem, a natural extension of MIPS with a computational limit for the search efficiency and quality trade-off, while still keeping the preprocessing time near-linear (in order to still allow for frequent updates of the model as user preferences changes).

Assuming $d = O(\log n)$, we require an algorithm for budgeted MIPS to build a data structure in $\tilde{O}(n)$ time in the *preprocessing (query independent) phase*. In the *query (dependent) phase*, a budget $\mathcal{B}$ of computational operations is given, and the algorithm must answer the MIPS query in $O(\mathcal{B})$ [27].

Budgeted MIPS has been studied recently in [50], using a budget of $\mathcal{B} = \eta n$ inner product computations, where η is a small constant. Furthermore, the budgeted versions of several other search problems have also been studied [34].

In the paper *Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search* [27] (see Appendix A.7), we investigated different sampling approaches for budgeted MIPS. We proposed a novel *deterministic* technique, called *GreedySam* for improving sampling schemes for budgeted MIPS. These techniques were applied to *wedge* [9] and *diamond sampling* [2]. Furthermore, we proposed a *shifting technique* used to modify the schemes to also handle negative inputs. A novel analysis was provided, and our approach was compared to other sampling approaches: basic sampling, and traditional wedge and diamond sampling [2,9], as well as to the state-of-the-art methods for solving exact MIPS, FEXIPRO [23].

In general, sampling approaches for MIPS fits well for the budgeted version, as the number of samples allowed (and thus the quality of the result) will be tied to the budget, allowing for explicit control of the quality through the budget.

Given point set X, let $\mathbf{X} \in \mathbb{R}^{n\times d}$ denote the matrix with elements from X as rows, and let $\mathbf{y}_j$ denote column j. On query $\mathbf{q}$ the overall idea behind wedge sampling is to sample indexes of points in X, where index i is sampled with probability $\mathbf{x}_i \cdot \mathbf{q}/z$, where $z = \sum_{i=1}^{n} \mathbf{x}_i \cdot \mathbf{q}$. In practice, the algorithm first samples a column $\mathbf{y}_j$ from $\mathbf{X}$ with probability $q_j c_j / z$ and then samples a row i with probability x_{ij}/c_j. As can be seen:

$$\begin{aligned}\mathbb{P}\left[\text{Sampling } i\right] &= \sum_{j=1}^{d} \mathbb{P}\left[\text{Sampling } i|\text{Sampling } j\right] \cdot \mathbb{P}\left[\text{Sampling } j\right] \\ &= \sum_{j=1}^{d} \frac{x_{ij}}{c_j} \cdot \frac{q_j c_j}{z} = \frac{\sum_{j=1}^{d} x_{ij} q_j}{z} = \frac{\mathbf{x_i} \cdot \mathbf{q}}{z} \quad .\end{aligned}$$

Note that, in the query phase, z can be computed in $O(d)$ time by computing

$z = \sum_{i=1}^{n} \mathbf{x}_i \cdot \mathbf{q} = \sum_{j=1}^{d} c_j q_j$, where $c_j = \sum_{i=1}^{n} x_{ij}$ (i.e. the sum of $\mathbf{y}_j$) for all $j \in [d]$ can be computed in the preprocessing phase.

In this way, s rows are sampled. A histogram is used to keep track of the number of times any row i has been sampled. From this histogram, the top-k candidates can be extracted in $O(n \log k)$. Results might be improved by performing a post-processing step, where the exact value of the inner products between $\mathbf{q}$ and the top $m > k$ candidates extracted from the histogram are computed in time $O(dm + n \log m)$.

Diamond sampling was originally proposed for approximating matrix products [2], but works for MIPS as well. While the description of the algorithm is different from wedge sampling, we show in the paper [27] how it works in a similar way, but with an extra step. Basically, diamond sampling applies the techniques of wedge sampling to sample a row $\mathbf{x}_i$, before basic sampling (sampling with probability $q_{j'}/\mathbf{q}$) is applied in order to sample $x_{ij'}$ from $\mathbf{x}_i$. The final sample has expected value proportional to $(\mathbf{x}_i \cdot \mathbf{q})^2$, and is used to rank the top-k candidates. A detailed description is given in the the paper [27].

The implementation specific details of wedge and diamond sampling are given in the paper [27]. We proposed three techniques in order to improve upon the schemes:

1. A simple *shifting technique*, allowing for transforming any MIPS problem instance with negative values into an instance with values in $\mathbb{R}_+$, while preserving the order of the inner products. Wedge and diamond sampling employ methods for handling negative values, but their analyses break down in this case.
2. A new data structure for handling the sample counting in wedge sampling. In addition to the n-sized histogram, we store an s-sized array of hashmaps, which tracks indexes in the histogram according to their count. Traversing this tracking array, allows for candidate extracting in $O(s)$, meaning that the running time of the query dependent phase is independent of n.
3. A *deterministic sampling scheme*, called GreedySam. The scheme replaces the row sampling in the wedge sampling scheme, i.e. the sampling of the index i from the distribution given by column j. Wedge sampling pre-samples these rows in the preprocessing phase, and our deterministic scheme replaces this randomized sampling [9, 27].
   As diamond sampling also relies on wedge sampling [27], GreedySam is also applicable for this algorithm [2].

In our experiment, we compared and evaluated the sampling schemes and the proposed modifications:

- `Wedge` and `Diamond`: Standard wedge and diamond sampling.
- `dWedge` and `dDiamond`: Wedge and diamond sampling using GreedySam.
- `dsDiamond`: Diamond sampling using GreedySam and shifting.

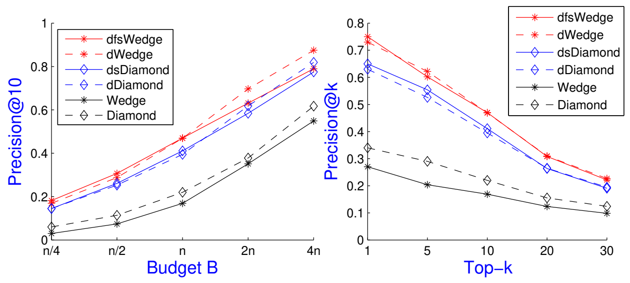


Figure 3.1: Comparison of accuracy with $k = 10$ and varying $\mathcal{B}$ (left) and speedup with $\mathcal{B} = n$ and varying k (right) between the wedge and diamond variants on MovieLens1M.

– `dfsWedge`: Wedge sampling using GreedySam, shifting, and the improved sample counting.
– `Greedy`: The greedy approach of Yu et al. [50].

For accuracy comparisons, the reported accuracy is the average of the top-k accuracy for each query in the dataset (also denoted *Precision@k*). For speed comparisons, the speedup is measured compared to a bruteforce implementation. The algorithms were evaluated on several large-scale benchmark datasets, using 100 query points for each set. A selection of the results is presented below.

Figure 3.1 shows a comparison of the accuracy of the wedge and diamond variants on the MovieLens1M dataset ($n = 3,952$, $d = 150$), with fixed $k = 10$ and varying $\mathcal{B}$ (left) and fixed budget $\mathcal{B} = n$ and varying k (right).

We note, how the modified schemes clearly outperform the non-modified, with the modified versions of wedge sampling obtaining the best accuracy in both cases. Interestingly, while `Diamond` performs better than `Wedge`, the order is reversed for the modified versions.

Between `dWedge` and `dfsWedge`, `dWedge` seems to have the edge, especially when $\mathcal{B}$ is large, although `dfsWedge` still has the advantage of working in cases where the input is negative.

Figure 3.2 shows a comparison of the accuracy (left) and speedup (right) obtained by the modified wedge sampling schemes and the greedy approach [50] on the Yahoo dataset ($n = 624,961$, $d = 50$), with $k = 5$ and varying budget $\mathcal{B}$.

On this large-scale dataset, `Greedy` provides the highest accuracy for small $\mathcal{B}$, and consistently beats `dWedge` in both running time and accuracy. However, `dfsWedge` performs better in terms of running time, and for $\mathcal{B} > n/2$, `dfsWedge` also outperforms `Greedy` in terms of accuracy, with the gap for $\mathcal{B} = 3n$ being around 10%.

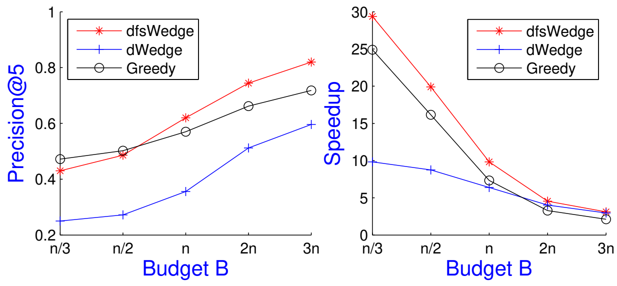


Figure 3.2: Comparison of accuracy (left) and speedup (right) between `dfsWedge`, `dWedge` and `Greedy` on YAHOO with varying $\mathcal{B}$ and $k = 5$.

Finally, we also compared `dfsWedge` to FEXIPRO, an exact solution for MIPS [23]. Compared on NETFLIX ($n = 17,770$, $d = 200$), and MOVIELENS10M ($n = 65,133$, $d = 150$) with $k = 5$ and budgets $\mathcal{B} = 10n$ and $\mathcal{B} = n/4$ respectively, `dfsWedge` maintained an accuracy above 80%, while achieving a speedup factor over `FEXIPRO` of 9 for NETFLIX and 1.17 for MOVIELENS10M. Furthermore, the speedup over the bruteforce solution for the latter dataset was an incredible 118.

As can be seen from the above experiments, our modified version of wedge sampling, `dfsWedge`, performs better on several real world large-scale recommender system datasets. The budget allows for explicit control of the quality/running time trade-off for every query, making our approach relevant in many real life cases, where a fast response is important.

3.2. (Free) Theoretical Guarantees for Random Forests

As discussed in Section 2.2, knowing when to trust a machine learning algorithm is important in many cases. In most cases, an algorithm is trained on one dataset (a *training set* T) and evaluated on an external second hold-out set (a *test set* T_{ext}, from the same distribution as T), which is kept separate from T in order to get an unbiased estimate of the quality of the algorithm.

However, aside from being unbiased, this estimate comes with no other guarantees. One may obtain bounds by analyzing the complexity of a given model using the VC dimension, through analysis of the training procedure, or by probabilistic analysis of the performance on a validation set T_{val}.

In the paper *X-PAC* (see Appendix A.6), we investigated theoretical guarantees for the performance of *majority vote classifiers*, a special kind of aggregate classifier, where the final classification is based on a (possibly weighted) majority vote among a set (an *ensemble*) of weaker classifiers (voters). Specifically, we considered the standard *random forest* classifier [5], a majority vote classifier where individual voters are decision trees [2].

The random forest is one of the most successful machine learning algorithms [5], being easy to apply and to parallelize, and it often achieves high accuracies in practice [16]. While Breiman provided some analysis and theoretical bounds in his initial paper [5], the algorithm is still not well understood theoretically [13].Several simpler and easier analyzable variants have been proposed, such as *purely random forests* [6] and *Bernoulli random forests* [49], although they are less successful empirically, than the standard random forest [49].

For the study, we limited ourselves to binary classification problems. We considered different bounds based on PAC-Bayesian analysis, where PAC stands for the *Probably Approximately Correct* frequentist learning model [48]. Usually, applying such bounds require a hold-out validation dataset T_{val}. However, random forests are usually trained using *bagging*: given training set T, for each tree, a random subset $T_i \subset T$ is sampled with replacement, such that $|T_i| = |T|$ [4,5], thus generating validation sets $\bar{T}_i = T \setminus T_i$ for individual trees as part of the training procedure. These sets can be utilized by some of the bounds, and thus we obtain bounds "for free" in the sense that no hold-out data needs to be reserved.

The bounds considered in our study is presented below. We first introduce some notation [26]. Let $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be an independent identically distributed sample from $\mathcal{X} \times \{-1, 1\}$, drawn according to an unknown distribution D. A hypothesis (for instance a decision tree) is a function $h : \mathcal{X} \to \{-1, 1\}$, and $\mathcal{H}$ denotes a space of hypotheses. The bounded loss function $\ell : \mathcal{Y}^2 \to [0, 1]$ is used to evaluate $h \in \mathcal{H}$. Furthermore, the expected loss of h is denoted by $L(h) = \mathbb{E}_{(X,Y)\sim D}\left[\ell(h(X), Y)\right]$ while the empirical loss on S is given by $\hat{L}(h, S) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(X_i), Y_i)$.

Given a set of hypotheses $\mathcal{H}$, the ρ-weighted *majority vote classifier* h_M is an aggregate classifier, which predicts

$$h_M(X) = \text{argmax}_{Y \in \mathcal{Y}} \sum_{h \in \mathcal{H} \wedge h(X) = Y} \rho(h).$$

h_M has expected loss $L^{\text{MV}}(h_M) = \mathbb{P}_{(X,Y)\sim D}\left[\mathcal{M}_\rho(X, Y) \leq 0\right]$ and the empirical loss $\hat{L}^{\text{MV}}(h_M, S) = \mathbb{P}_{(X,Y)\sim S}\left[\mathcal{M}_\rho(X, Y) \leq 0\right]$.

Closely related to h_M is the *Gibbs classifier*. A Gibbs classifier h_G is a stochastic classifier. On input X, h_G samples $h \in \mathcal{H}$ according to ρ, and $h(X)$ is returned. h_G has expected loss $L^{\text{Gibbs}}(h_G) = \mathbb{E}_{h\sim\rho}\left[L(h)\right]$ and empirical loss $\hat{L}^{\text{Gibbs}}(h_G, S) = \mathbb{E}_{h\sim\rho}\left[\hat{L}(h, S)\right]$.

We investigate three different kinds of bounds:

[2] While these bounds are not directly applicable to our final model in the ghostwriter study, some of our early experiments used random forests, although results were suboptimal.

- The *C-bounds* (specifically the $C1$-bound and the $C2$-bound) [17], which can bound $L^{\mathrm{MV}}(h_M)$ directly.
- The *PAC-Bayesian kl-bound (PBkl-bound)* [38], for bounding $L^{\mathrm{Gibbs}}(h_G)$. It can be seen, that the loss $L^{\mathrm{MV}}(h_M)$ of a majority vote classifier h_M is at most twice the risk $L^{\mathrm{Gibbs}}(h_G)$ of the corresponding Gibbs classifier h_G [17]:

$$L^{\mathrm{MV}}(h_M) \leq 2L^{\mathrm{Gibbs}}(h_G),$$

  thus allowing us to bound $L^{\mathrm{MV}}(h_M)$ using the PBkl-bound and paying a factor of two.
- The *single hypothesis bound (SH-bound)*, for bounding the loss of a single hypothesis, $L(h)$. The SH-bound can be used to bound $L^{\mathrm{MV}}(h_M)$ directly by considering h_M a single hypothesis.

The discussion here will be limited to a high level discussion of the bounds; the full definitions can be found in the paper [26]. However, we note that all bounds relies on an evaluation of the classifier on some hold-out data S.

In the case of the majority vote classifier, the SH-bound considers h_M a single hypothesis, and thus uses only $\hat{L}^{\mathrm{MV}}(h_M, S)$ and no other information. Being derived for Gibbs classifiers, the PBkl-bound utilizes information about ρ and $\hat{L}^{\mathrm{Gibbs}}(h_G, S)$. And finally, the C-bounds also include information about the correlation between classifiers. Thus, the $C1$-bound uses ρ, $\hat{L}^{\mathrm{Gibbs}}(h_G, S)$ and the correlation between classifiers, in the form of the *empirical disagreement* $\hat{d}^S_\rho$, computed as the ρ-weighted average disagreement between any two voters [26].

Theoretically, this should give the C-bounds an edge. However, when individual voters are strong, the bound deteriorates [17, 26], as it becomes harder to estimate the correlation.

As mentioned, we can apply the PBkl-bound and the C-bounds without a hold-out validation set, when the majority vote classifier is trained using bagging. This can be done, since we may compute the empirical Gibbs loss using only the hold-out sets $\bar{T}_i$:

$$\hat{L}^{\mathrm{Gibbs}}_{\mathrm{OOB}}(h_G, T) = \mathbb{E}_\rho \left[\frac{1}{|\bar{T}_i|} \sum_{(X,Y)\in\bar{T}_i} \ell\left(h_i(X), Y\right) \right].$$

The empirical disagreement can be computed in a similar fashion, but requires using the intersection of the out-of-bag sets $\bar{T}_i \cap \bar{T}_j$ between any two voters h_i, h_j. As these sets are generally much smaller, the C-bounds deteriorates if T (and hence $\bar{T}_i, \bar{T}_i$) is small.

Note, that the SH-bound cannot be applied without a separate validation set, as $\hat{L}^{\mathrm{MV}}(h_M, S)$ cannot be computed from the out-of-bag sets.

We evaluated the bounds on a set of benchmark datasets, using the standard

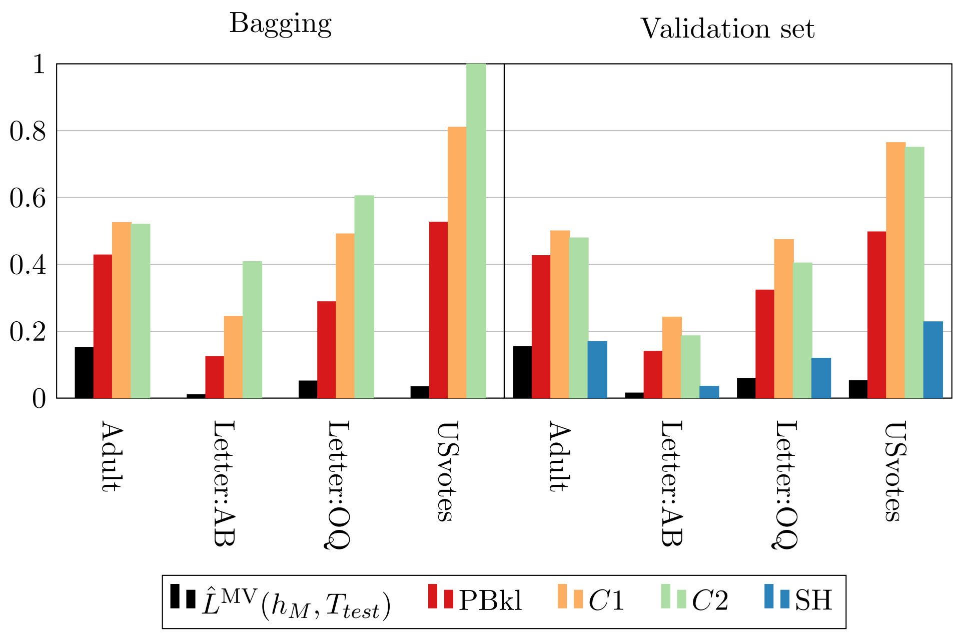


Figure 3.3: The bounds computed in the bagging setting (left) and the validation set setting (right) for a selection of the datasets tested in [26] (see Table 1 and Table 2 in Appendix A.6), as well as the error on the external test set.

random forest trained with bagging, and ρ being the uniform distribution. We considered both the setting without and with a hold-out dataset:

1. **Bagging setting:** The random forest is trained on all the data, and bounds (with exception of the SH-bound) are based solely on the out-of-bag sets, $\bar{T}_i$.
2. **Validation set setting:** Half of the data is used for training, while the other half, T_{val}, is used for the bounds.

In each setting, an external dataset T_{ext} was set aside for evaluating the strength of the final classifier.

Figure 3.3 presents the bounds evaluated on a subset of the datasets considered in [26] for both settings (numerical values given in Table 2 in Appendix A.6). In Figure 3.3, the test score $\hat{L}^{\mathrm{MV}}(h_M, T_{\mathrm{ext}})$ provides an estimate of the accuracy of the classifier. For the bagging setting (left), the PBkl-bound always gave the tightest bounds[3], beating the C-bounds, even with the factor of two. Thus we clearly see the C-bounds degrading, when

[3] This was true for all datasets tested, although some bounds were trivial, i.e. greater than 0.5

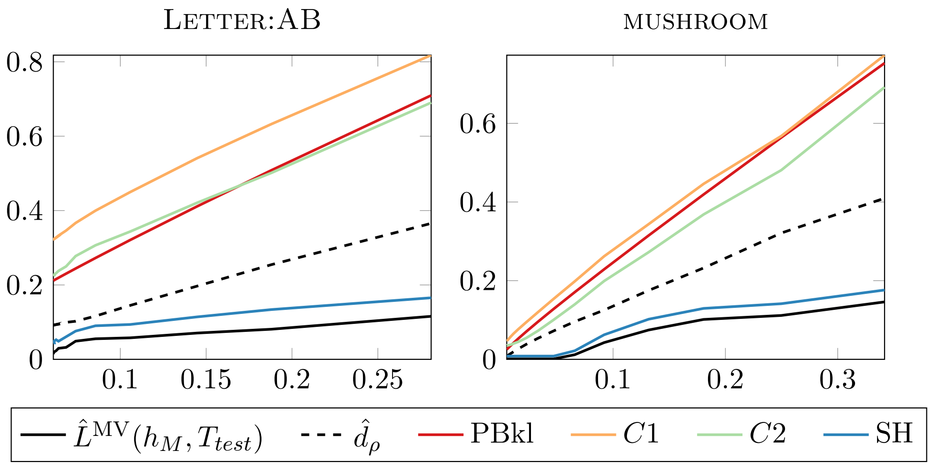


Figure 3.4: Development of bounds as individual voters become weaker. The plots show results on the Letter:AB and mushroom datasets in the validation set setting, using a random features for splits.

individual voters are strong [17] and when only limited amount of data is available for estimating correlations.

In the setting with a separate validation set (right), the results between the PBkl-bound and the C-bounds remain the same (again caused by the strong individual voters), although they are all looser than the SH-bound. This indicates that the PBkl-bound does indeed suffer from not taking correlations into account, even if it outperforms the C-bounds.

Comparing the bounds obtained between the two settings, we see that the bounds are similar (with the exception of the SH-bound). This is caused by the balance between using data for training and for evaluating bounds; in the second setting, more data is available for the bounds, resulting in tighter bounds at the cost of slightly weaker classifiers, as seen by the slightly worse $\hat{L}^{\mathrm{MV}}(h_M, T_{test})$ (this is especially clear for the USvotes dataset, which is very small, with $n = 232$).

The SH-bound provides the best guarantees for all datasets across both experiments, indicating that the other bounds are still too loose. However, the bound is only applicable when having a separate validation set, whereas the bounds in the bagging setting come “for free”, leaving more data available for selecting the hypothesis.

In order to further investigate the impact of the strength of the individual voters, we performed experiments with varying voter strength. This was done by limiting the depth of the trees in the random forest (from depth 1 to unlimited depth), as well as choosing different splitting strategies, i.e. choosing a random feature instead of the best feature.

This experiment was performed in both settings. Figure 3.4 shows the result for the validation set setting, when using a random feature in splits. The figure plots the resulting risk on the test set, the empirical disagreement, and the bounds as a function of the average voter strength. Only results for the MUSHROOM and LETTER:AB datasets in the validation set setting are shown, as these are representative; the full experiment can be found in the paper.

It can be seen, that between the PBkl-bound and the C-bounds, the $C2$-bound becomes the better alternative, when the voters are weak. However, the SH-bound is still clearly superior in this case. For the bagging setting, the PBkl-bound remained tighter than the C-bounds due to the smaller amount of data available, even when the voters are weak [26].

Finally, we also investigated and compared the λ-bound [44] and the $C3$-bound [17] for model selection. These bounds allows for optimizing the posterior ρ. In our experiment, we found that the λ-bound overfitted ρ to the strongest voters, while the $C3$-bound ended up with the uniform posterior. A detailed description will not be given here, but can be found in the paper [26].

In conclusion, we have seen how non-trivial bounds can often be obtained "for free" for classifiers trained using bagging. The PBkl-bound [38] is often the better alternative in this case, unless voters are weak and a lot of data is available.

When using a validation set, we typically obtained a worse classifier, as less data was available for training. However, much better bounds can be obtained by applying the SH-bound to the final classifier, indicating that both the PBkl-bound and the C-bounds are still too loose.

Chapter 4. Company Collaboration

As mentioned, the focus of DABAI is two-fold: providing solutions to companies and doing research. While the two previous sections discussed the data mining and research aspect of my work, this section will focus on the collaboration with the companies, including the implementation and integration of our solutions, as this has taken up a lot of time during my PhD, especially with Clio. Furthermore, the section will discuss the reception and the publicity surrounding these projects, as this has been a big part of the work with the companies as well, especially in the MaCom project, which saw a huge amount of interest in the media afterwards.

The collaboration part of DABAI has been structured around milestones every half year. Thus, a project with a company is expected to develop according to the following milestones:

M0 *(Month 0)* Project is initiated.
M1 *(Month 6)* Initial analysis is completed and problems are identified.
M2 *(Month 12)* Proof-Of-Concept is completed.
M3 *(Month 18)* Working prototype is completed.
M4 *(Month 24)* An industrial prototype is deployed.

As I have mainly been involved in the collaboration with Clio and MaCom, the two sections below will discuss the collaboration with these companies respectively. For either company, the collaboration follows the above structure, although the MaCom case has only reached **M3**. For both companies, the final prototypes are quite close to the methods described in Chapter 2; thus the discussion in this section will focus on other aspects, such as requirements, practical challenges and ethical considerations.

Common for all company projects in DABAI is the initial analytic phase, in which interesting problems are identified together with the companies. During late 2016 and early 2017, we conducted this analysis with the educational companies of DABAI: EduLab [15], Clio [8] and MaCom [29], which resulted in a set of formalized problems.

These problems were presented in the paper *DABAI: A data driven project for e-Learning in Denmark*[4], published at ECEL [1] (see Appendix A.2). The following discussion is similar to the content of the paper and may contain passages from the paper without quotation.

[4] In the paper, Clio is referred to by their former name: Clio Online

In the paper, each identified problem is described, together with the motivation and potential for the company. Furthermore, we identified three categories, which cover the types of problems encountered with the companies. Thus, each problem is categorized according to the following categories:

- Student Profiling
- Content Profiling
- Content Recommendation

The problems identified in the paper include student performance prediction (Clio), detecting ghostwriters (MaCom), optimizing e-learning personalization (EduLab), etc. The following sections will further elaborate on the problems, including motivation and potential, as identified in the paper.

4.1. Clio

This section describes the collaboration with Clio, with some focus on the time consuming task of integrating our solutions as part of the **M4** milestone.

4.1.1 Potential and Requirements

Working with Clio (former Clio Online), we identified two main problems, as described in [1]: *predicting student performance* and *finding similar quizzes.*

In our discussion, we quickly determined **predicting student performance** to be a central problem for Clio. The problem is also important to the educational data mining community in general, as good estimations of scores on unseen quizzes allow teachers to select the correct quiz and provide remedial support for weak students, or they may highlight problems with learning materials. Depending on the point of view, the problem may fit in all of the categories from [1], although most methods for prediction (at least the ones, we investigated) relies on *student profiling* and *content profiling.* The student performance prediction ended up being our main project with Clio. In practice, as quizzes in the Clio system is graded on the Danish grade scale, the system should predict the correct grade, out of seven possible. The initial hope from Clio was for the system to achieve as high as 80% accuracy.

Furthermore, we determined detecting the **similarity among quizzes** to be of interest. Here, similarity is considered in terms of answer history, i.e. two quizzes are considered similar, if similar students get similar grades. Determining this similarity is a *content profiling* problem. With thousands of quizzes in the Clio system, a tool for determining similar quizzes has great potential, such as helping content creators create unique quizzes, or allow teachers to select suitable training quizzes for their students.

While we never directly developed a tool for determining quiz similarity, some of the tested prediction algorithms were based on quiz similarity, by using similar, answered quizzes in order to give a prediction for a student on an unseen quiz (as also mentioned in Section 2.1). These methods store the quiz similarity as part of the model, allowing easy access to similar quizzes.

4.1.2 Data Cleaning and Prototype Development

Having identified the above problems, we started working on prototypes for solving the performance prediction problem.

Clio supplied us with raw (anonymized) data from their data base. This presented us with several initial challenges:

- Getting an initial overview of the database.
- The relevant data had to be extracted. For instance, quizzes were graded by an internal points system specific to each quiz, and required normalization.
- Sorting out invalid data, for instance quizzes with invalid or inconsistent quiz ids or grading scales.
- Reducing the "noise" in the data.

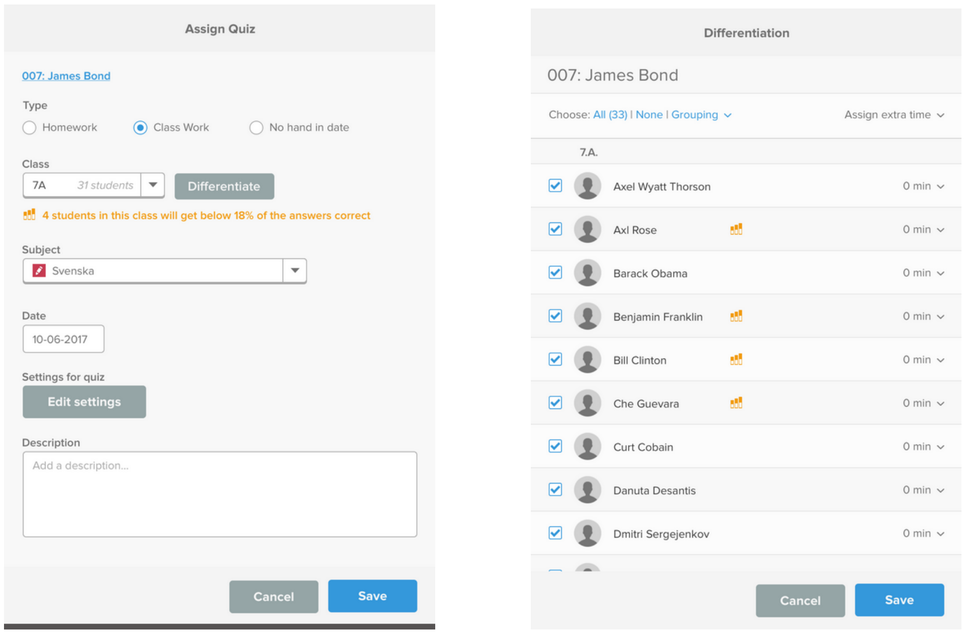
Especially the last point was time consuming, as the initial data set had a lot of noise. Most notably, around 60% of answers received a score of 0 (i.e. a grade of -03). We suspected, and confirmed through dialog with the technical lead at Clio, that this was due to students aborting quizzes. As we were only interested in students trying to answer the entire quiz, we implemented several cleaning procedures in order to extract the relevant data (for instance including only quizzes with a minimum amount of time spent). This cleaning alleviated the problem to some degree, although we still discovered anomalies from time to time.

With the data cleaned, we performed the experiments as described in Section 2.1 and [28]. As mentioned, the initial prototype was tested on a densified dataset, including only the most active students and quizzes. This reduction in data size was used in order to ease the development of the initial prototype.

During our development of the initial prototype using non-negative matrix factorization, Clio started developing a user interface. The mock-up, is shown in Figure 4.1, and displays the desired features. On assigning a quiz, the system initially warns the teacher, that some students are at risk (left). Pressing "Differentiate" shows the teacher exactly which students are at risk, and allows the teacher to assign more time.

As mentioned, Clio requested a solution able to predict the correct grade for 80% of students. Our initial prototype did not reach this goal, obtaining only an RMSE of around 0.1744, corresponding to less than 50% grades correctly predicted on the densified data set.

The results were discussed with Clio, and the goals were redefined as follows: in stead of predicting a single grade, the system could be allowed to


Assign Quiz

007: James Bond

Type
Homework [✓ Class Work] No hand in date

Class
7A 31 students
Differentiate
4 students in this class will get below 18% of the answers correct

Subject
Svenska

Date
10-06-2017

Settings for quiz
Edit settings

Description
Add a description...

Cancel Save

Differentiation

007: James Bond

Choose: All (33) | None | Grouping
Assign extra time

7.A.

✓	Axel Wyatt Thorson	0 min
✓	Axl Rose	0 min
✓	Barack Obama	0 min
✓	Benjamin Franklin	0 min
✓	Bill Clinton	0 min
✓	Che Guevara	0 min
✓	Curt Cobain	0 min
✓	Danuta Desantis	0 min
✓	Dmitri Sergejenkov	0 min

Cancel Save

Figure 4.1: Proof-of-Concept user interface designed by Clio. **Left:** screen for assigning a quiz to a class. **Right:** overview of class with students expected to obtain a low score highlighted.

predict two grades. Furthermore, we started to investigate how to include other data, such as log data, in the prediction [24], in order to give better predictions for students with limited history in the quiz system.

As can be seen from Section 2.1, the results worsened when including the full data set. It appears that the historical data by itself is not enough, and unfortunately we did not managed to incorporate the log data in the prediction successfully.

Even so, while working on improving the prototype, we also started the work to integrate the prediction system in the Clio system.

4.1.3 Full Integration

As we started the full integration with Clio, we faced a new set of challenges:

- **Fetching data from the live database**: Clio provided access to the live database through a set of APIs. These APIs allowed for accessing the relevant (anonymized) data from their database, i.e. quiz answers, log entries, and quiz/site meta data, although rather slow, as the data was stored at different instances.
- **Accessible to the Clio application**: As Clio is based on AWS,

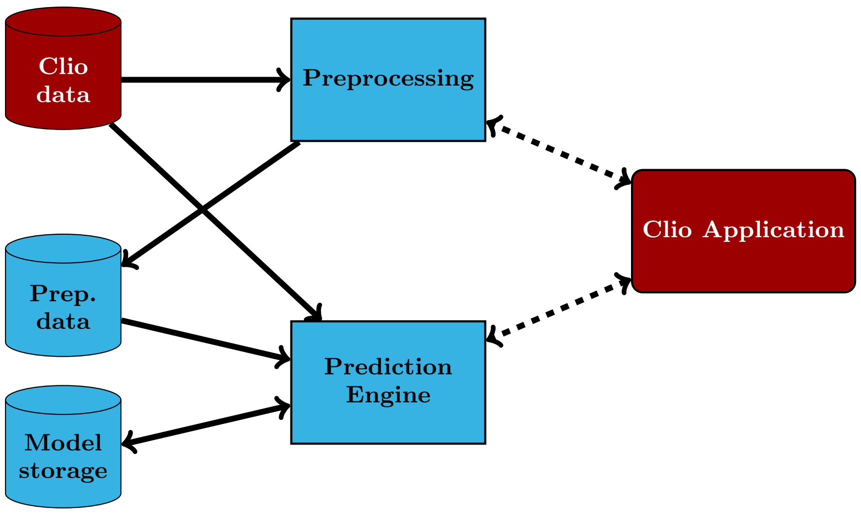


Figure 4.2: Overview of the integration of the prediction system with the Clio software. Cylinders indicate databases, while squares indicates applications, with the red color indicating existing Clio instances. Arrows indicate reads/writes from applications, with the dashed arrows indicating communication through the AWS lambda protocol.

they requested that our application should run on the AWS Lambda platform[5], and be accessible through the AWS Lambda protocol.

- **Frequent model updates:** The model should be updated regularly, at least once a day. While the training procedures were efficient enough, that training the models once every night was feasible, fetching and preprocessing the data presented a bottleneck; hence we settled on a model, in which new data is preprocessed every night, and stored in a local database on the AWS instance, before the model is updated.

Thus, we ended with the architecture shown in Figure 4.2: our solution is split in two applications: the **preprocessing** application and the **prediction engine**, both of which can be invoked through the AWS Lambda protocol.

The preprocessing application takes care of preprocessing the raw **Clio data** and storing it in the local database (**Prep. data**), while the prediction engine updates the model, which is stored in **Model storage**. Furthermore, the prediction engine answers prediction requests (which also requires reads from the **Clio data** in order to properly convert the predicted score to the corresponding grade.

The final system was handed over to Clio, with the models described in

[5] `https://docs.aws.amazon.com/lambda/latest/dg/welcome.html`, accessed August 2019.

Section 2.1. However, Clio is yet to deploy our solution.

4.2. MaCom

This section describes the collaboration with MaCom. While our solution with MaCom only made it to the prototype state (**M3**), this section will also discuss the huge media interest in our prototype.

4.2.1 Motivation and Potential

In our initial meetings with MaCom, we discussed the problem of **detecting ghostwriters** (or **authorship verification**), as described in [1].

The problem is motivated by the increasing use of ghostwriters in Danish high schools (especially for the Studieretningsprojekt, SRP) [47]. As also mentioned in Section 2.2, MaCom already has a system for detecting copy-paste plagiarism, but the authorship verification problem is harder, as the texts in question are original work. Thus, methods for solving the problem must *learn to recognize the writing style of a student*, and the problem is therefore clearly a case of *student profiling.*

Hiring ghostwriters in Denmark became easier with the launch of the first Danish *paper mill*, www.fixminopgave.dk [47]. While the government has since banned the site, the sale of SRPs and essays happened before on other web forums, and continues to happen; hence, MaCom is eager to have a solution incorporated in their system. Furthermore, the problem is widespread outside of Denmark; in the USA, paper mills are abundant. Websites need only include a disclaimer somewhere on their site, telling students to only use the papers as “study support”, in order to be legal [46].

As any solution for detecting ghostwriters basically makes accusations towards students, MaCom had some requirements for the software. The final solution should not make more than 5% false accusations on a test set, while still having a significant catch rate (although a catch rate high enough to have a deterring effect would be enough).

As mentioned in Section 2.2, we also investigated how to track changes in writing style as a means to provide feedback to the teacher. We did not consider this problem in our paper for ECEL [1]; rather the idea surfaced as a “positive” (in the sense of providing insight instead of making accusations) use of the writing style analysis tools developed for the ghostwriter detection task.

4.2.2 Data Access and Prototype Development

Working with MaCom, we took a different approach than with Clio. Most notably, the ground work was done in two master theses during 2017-2018 [19, 43]. As mentioned, the first thesis considered traditional methods from authorship verification. However, the methods applied were not easily configurable with respect to limiting the number of false accusations. Furthermore, the accuracy obtained was not satisfactory. Hence, the second thesis consider neural network based methods, which saw a lot more success, and allowed for easy configuration of sensibility.

During the master's projects, we would meet with the students on a weekly basis, discussing ideas and implementation. The students would then go to MaCom and test the ideas. Compared to Clio, MaCom had more restrictions regarding data access; aside from being anonymized, the raw text data could only be accessed by a computer at the MaCom office. Thus, working by proxy through the master students was beneficial, as we did not have to go ourselves for the initial analysis and prototyping.

After the second master's project had been concluded, we implemented the final framework for writing style analysis and authorship verification, based on ideas tested in the thesis. The results obtained using this prototype were published in the paper *Detecting Ghostwriters in High Schools* [42], and summarized in Section 2.2. The method used allowed for easy configuration of the similarity threshold required, meaning that the number of false accusations could easily be adjusted. As can be seen from Figure 2.5 (right), the software still performs well on a balanced data set, when the number of false accusations is below 5%.

While MaCom is yet to deploy our solution, we have discussed, how to deploy it. Two main ways of deploying the software has been discussed: a) as a screening procedure for all essays, or b) as a tool to be used, when there is already a suspicion of cheating.

While the screening option will be similar in function to the regular plagiarism checker, we will encounter the "problem" of only having very few ghostwriters. The live dataset will (most likely) contain much fewer than 50% ghostwriters, which will require a re-tuning of the threshold and lead to a lower catch rate. As such, the second option appears to be the better alternative; often a teacher will have a suspicion beforehand, for instance based on the quality/grade of the essay, and considering only suspected students, the input data will likely be more balanced.

However, even if used as a supporting mechanism, the final decision rests with the teacher or school administration. A first step may be to require the suspected student to do an oral presentation, rather than making an outright accusation. This is indeed the way suspicions of cheating are currently handled in many Danish high schools.

After the publication of the ghostwriter detection results, we started looking into the other problem: tracking changes in writing style during high school [25]. The prototype was extended to included the functionality for performing the analysis described in Section 2.2 and handed over to MaCom. Thus this final framework includes three modules:

– A writing style similarity module, used for computing the similarity of the writing style in any two texts.
– An authorship verification module, allowing the verification of authorship of a given text when given the corpus of texts for the claimed author [42].
– A writing style analysis module, used for detecting development patterns among groups of students [25].

While MaCom is yet to deploy the software for either task, the implementation is open source and available at GitHub[6], which also contains a more thorough explanation of the software.

4.2.3 Media Interest

After the conclusion of the initial master project dealing with ghostwriter detection [19], the University of Copenhagen issued a press release about the results. The story was picked up by a few publications, for instance the Danish newspaper Berlingske[7].

However, the story about the ghostwriter detection software did not really take off until after the conclusion of the second thesis [43] and the publication of the paper [42]. The university issued a new press release, which were picked up by more than 50 Danish publications and more than 100 international publications from USA, India, Russia, Argentina, Spain, Netherlands, Canada and more.

In general, most publications were positive about the project. The story was featured on the two major news sites in Denmark, Danmarks Radio (DR)[8] and TV2[9], as well as in national radio, and shared on Facebook and Twitter. Among the largest international publications to pick up the story were Sputnik[10] and Scientific American[11], the latter of which interviewed me for a podcast about the project.

The project did also receive some negative publicity, such as comments on Facebook/Twitter, and an article from Version2[12], questioning the legality of using the high school students' essays.

[6] Writing style analysis framework: `https://github.com/StephanLorenzen/AuthorshipVerification`
[7] https://www.berlingske.dk/samfund/ny-metode-kan-opdage-eksamenssnyd
[8] https://www.dr.dk/nyheder/viden/tech/kunstig-intelligens-afslorer-om-du-selv-har-skrevet-eksamensopgaven
[9] https://nyheder.tv2.dk/samfund/2019-05-29-nyt-vaerktoj-kan-afslore-om-opgaven-er-kobt-pa-nettet
[10] https://sputniknews.com/military/201906111075784789-machine-learning-vs-fake-learning-ai-can-now-predict-when-youll-cheat/
[11] https://www.scientificamerican.com/podcast/episode/high-school-cheaters-nabbed-by-neural-network/
[12] https://www.version2.dk/artikel/lectio-stiller-130000-opgaver-raadighed-forskere-uden-at-informere-gymnasier-eller-elever

Publication	**Country**	**Note**
UCPH	Denmark	Press release
DR	Denmark	National news
TV2	Denmark	National news
Jyllands-Posten	Denmark	Newspaper
Politiken	Denmark	Newspaper
Version2	Denmark	Web media
Scientific American	USA	Web media
Science Daily	USA	Web media
Sputnik	Russia	News agency
wallstreet:online	Germany	Financial portal
India Today	India	News magazine
Scientific India	India	Web media
La Nacion	Argentine	Newspaper
La Vanguardia	Spain	Newspaper

Table 4.1: Selection of publications with mentions of the ghostwriter software.

Table 4.1 presents a list of a few of the major publications from different countries, which published the story.

Chapter 5. Conluding Remarks

In this thesis, I have presented my work in educational data mining, some improved data mining techniques and theoretical bounds for majority vote classifiers, as well as an overview of the collaboration with the Danish companies Clio and MaCom. The structure of my PhD has followed that of the DABAI project, in which research has been motivated by problems encountered during the collaboration with the companies.

In collaboration with the companies Clio and MaCom, we have demonstrated, how using data from digital educational tools can provide valuable insight to teachers, as well as assist school administration in detecting fraud. With Clio, we evaluated methods based on matrix factorization for predicting quiz scores in primary school. While results were only slightly better than the baseline, our analysis of the resulting model provided insight about the students and the quizzes; for instance, the model might provide information about the skill level of a student or the difficulty of a quiz.

We further developed a method for analyzing student behavior, using the log data available from Clio. A set of activity measures were defined and correlated with performance (measured by scores obtained in quizzes), in order to detect optimal and suboptimal behavioral patterns. Furthermore, the method allowed for tracking changes in behavioral patterns among students, providing valuable insight for teachers. Finally, the method also provided valuable insight about the complexity of learning materials for the different classes covered by Clio.

With MaCom, we investigated methods for detecting the use of ghostwriters (known as authorship verification in the literature) in high school, a problem seeing a recent increase in Denmark. The proposed method relied on convolutional Siamese neural networks in order to compare writing styles of Danish essays, and obtained an accuracy of 0.875, catching almost 90% of cheaters, while making only 14.1% false accusations, although the latter can be improved at the cost of accuracy. The proposed method beat all of the traditional methods for authorship verification, while being more flexible.

As a spin-off from the ghostwriter detection project, we developed methods for tracking changes in writing style of high school students. We computed writing style development profiles of students, which were clustered and compared to several writing style quality measures. An analysis of the clusters showed both optimal and suboptimal development patterns; another valuable insight for high school teachers. Furthermore, a comparison of writing style between different students indicated, that the writing style becomes more individual, as students progress through high school.

Based on our work with Clio, we investigated sampling methods for *budgeted maximum inner product search*, an important algorithmic ingredient in many data mining methods. We evaluated the state-of-the-art methods empirically and theoretically, and proposed a series of algorithmic engineering techniques for improving wedge sampling. Our novel deterministic wedge-based algorithm runs significantly faster than the state-of-the-art methods on budgeted MIPS. The algorithm also compares well to exact MIPS, maintaining an accuracy of at least 80% on standard real-word recommender system data sets, while achieving a significant speedup.

Our work with MaCom highlighted the need for strong theoretical guarantees for machine learning algorithms. We investigated PAC-Bayesian generalization bounds for the standard random forest classifier. These bounds can be applied with no modification of the algorithm necessary. Furthermore, no additional data is required because the out-of-bag samples can be exploited, giving bounds "for free".

Our experiments showed how bounds inherited from the corresponding Gibbs classifiers clearly outperformed bounds tailored to majority vote classifiers, due to the individual decision trees being accurate classifiers by themselves, making estimation of correlations of errors difficult. Even with weaker individual classifiers, the majority vote bounds were outperformed when using only the out-of-bag samples for computing the bounds. Using a separate validation set, the single hypothesis bound was tightest by far, at the cost of a weaker classifier (as less data is available for training). This indicates, that the existing bounds for ensemble method (even those taking correlations into account) are not yet sufficiently tight.

A large part of my time during the PhD has been spent working with the companies. Thus, we started by identifying problems in collaboration with EduLab, Clio and MaCom, through several meetings and brain storming. In doing so, we found several interesting problems, as well as a categorization into three types of problems: student profiling, content profiling and content recommendation. The problems found were not only interesting to the companies, but also to the educational data mining community, and highlighted the great potential for the companies.

Working with Clio, the most time consuming part has been in accessing and cleaning their data, and later in integrating our final solution into their product. However, the project did reach the state of having a working, integrated prototype, although Clio is yet to deploy it.

In our collaboration with MaCom, we initially utilized master students with great success. In supervising these students, we managed to test several ideas. The main challenges in working with MaCom has been the strict data access policy, requiring physical presence at the MaCom office. However, we reached the goals required by MaCom, and while the final prototype has not yet been integrated in Lectio, the project received huge interest from national and international media.

5.1. Where do we go from here?

While we achieved interesting results in working with the companies, there are still several problems to be investigated further.

Most notably, while we spent a lot of time on trying to incorporate the data from the Clio log system into our prediction algorithms, we did not manage to improve significantly upon the algorithms relying only on the historical quiz data. Incorporating the log data into the performance prediction algorithms in a suitable way would hopefully improve predictions, especially for students with only limited activity in the quiz system.

While our later work with incorporating the log data did not improve the performance prediction, Clio has expressed interest in our writing style analysis with MaCom. The Clio system has several exercises centered around writing free text, and hence they are interested in tracking the improvement of writing style among the students. As it is a different domain (primary school vs. high school) and as the written texts are much shorter than the typical high school essay, adapting the method will most likely require significant work, for instance testing other architectures for the Siamese network. However, it is an interesting project for DABAI moving forward.

As for the ghostwriter detection project, the current method provides only a similarity score, but no justification. The convolutional neural network can be inspected in order to determine which stylistic markers in the text are investigated, but a more detailed feedback to the teachers is still desired. One approach could be to train and apply the network on sections of an essay instead, in order to be able to give more fine-grained feedback, such as highlighting sections with questionable authorship.

In regards to the theoretical bounds investigated for majority vote classifiers, our experiments showed how the existing bounds tailored to ensemble classifiers are not tight enough; the fact that the single hypothesis bound was much tighter than the C-bounds, even for weak classifiers, indicates that the C-bounds does not managed to properly estimate the correlation between individual voters. Therefore, more work is required for tightening the analysis of the effect of correlations in majority voting.

References

[1] S. Alstrup, C. Hansen, C. Hansen, S. Lorenzen, N. Hjuler, and N. Pham, DABAI: A data driven project for e-Learning in Denmark, *Proceedings 16'th European Conference on e-Learning (ECEL)* (2017), 18–24.

[2] G. Ballard, T. G. Kolda, A. Pinar, and C. Seshadhri, Diamond sampling for approximate maximum all-pairs dot-product (MAD) search, *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)* (2015), 11–20.

[3] C. Boutsidis and E. Gallopoulos, Svd based initialization: A head start for nonnegative matrix factorization, *Pattern Recogn.* **41**, 4 (2008), 1350–1362.

[4] L. Breiman, Bagging Predictors, *Machine Learning* **24**, 2 (1996), 123–140.

[5] L. Breiman, Random Forests, *Machine Learning* **45**, 1 (2001), 5–32.

[6] L. Breiman, Some infinity theory for predictor ensembles, *Journal of Combinatorial Theory A* **98** (2002), 175–191.

[7] J. Burrows, 'Delta': a Measure of Stylistic Difference and a Guide to Likely Authorship, *Literary and Linguistic Computing* **17**, 3 (2002), 267–287.

[8] Clio, *Clio*, `www.clio.me/dk`. Accessed August 2019

[9] E. Cohen and D. D. Lewis, Approximating matrix multiplication for pattern recognition tasks, *Journal of Algorithms* **30**, 2 (1999), 211–252.

[10] P. Covington, J. Adams, and E. Sargin, Deep neural networks for youtube recommendations, *RecSys* (2016), 191–198.

[11] DABAI, *Danish Center for Big Data Analysis driven Innovation*, `www.dabai.dk`. Accessed August 2019

[12] T. L. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, Fast, accurate detection of 100, 000 object classes on a single machine, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2013), 1814–1821.

[13] M. Denil, D. Matheson, and N. D. Freitas, Narrowing the gap: Random forests in theory and in practice, *Proceedings of the 31st International Conference on Machine Learning (ICML)*, PMLR (2014), 665–673.

[14] M. Desmarais, Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization, *Proceedings of the 4th International Conference on Educational Data Mining (EDM)* (2011), 41–50.

[15] EduLab, *EduLab*, `edulab.dk`. Accessed August 2019

[16] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, *Journal of Machine Learning Research* **15** (2014), 3133–3181.

[17] P. Germain, A. Lacasse, F. Laviolette, M. Marchand, and J.-F. Roy, Risk bounds for the majority vote: From a PAC-bayesian analysis to a learning algorithm, *Journal of Machine Learning Research* **16** (2015), 787–860.

[18] N. Hansen, C. Lioma, B. Larsen, and S. Alstrup, Temporal context for authorship attribution: a study of Danish secondary schools, *Multidisciplinary information retrieval*, Springer (2014), 22–40.

[19] K. Jürgensen, *Applied Algorithms for Detecting Text Authorship* (2017). Master's thesis

[20] N. Koenigstein, G. Dror, and Y. Koren, Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy, *Proceedings of the 10'th ACM Conference on Recommender Systems (RecSys)* (2011), 165–172.

[21] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, *Computer* **42** (2009), 30–37.

[22] Y. Koren, R. M. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, *IEEE Computer* **42**, 8 (2009), 30–37.

[23] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis, FEXIPRO: fast and exact inner product retrieval in recommender systems, *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD* (2017), 835–850.

[24] S. Lorenzen, N. Hjuler, and S. Alstrup, Tracking Behavioral Patterns among Students in an Online Educational System, *Proceedings 11'th International Conference*

on Educational Data Mining (EDM) (2018), 280–285.

[25] S. Lorenzen, N. Hjuler, and S. Alstrup, Investigating Writing Style Development in High School, *Proceedings 12'th International Conference on Educational Data Mining (EDM)* (2019).

[26] S. Lorenzen, C. Igel, and Y. Seldin, On PAC-Bayesian Bounds for Random Forests, *Machine Learning* **108**, 8-9 (2019), 1–20.

[27] S. Lorenzen and N. Pham, *Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search*, In submission for International Conference on Data Mining (ICDM) (2019).

[28] S. Lorenzen, N. Pham, and S. Alstrup, On Predicting Student Performance Using Low-rank Matrix Factorization Techniques, *Proceedings 16'th European Conference on e-Learning (ECEL)* (2017), 326–334.

[29] MaCom, *Lectio*, `www.lectio.dk/ommacom.htm`. Accessed August 2019

[30] H. G. McLaughlin, SMOG grading - a new readability formula, *Journal of Reading* (1969), 639–646.

[31] Y. Meier, J. Xu, O. Atan, and M. van der Schaar, Predicting grades, *IEEE Trans. Signal Processing* **64**, 4 (2016), 959–972.

[32] M. Pacheco, K. Fernandes, and A. Porco, Random Forest with Increased Generalization: A Universal Background Approach for Authorship Verification—Notebook for PAN at CLEF 2015, *CLEF 2015 Evaluation Labs and Workshop - Working Notes Papers* (2015).

[33] E. Pitler and A. Nenkova, Revisiting readability: A unified framework for predicting text quality, *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2008), 186–195.

[34] P. Ram, D. Lee, and A. G. Gray, Nearest-neighbor search on a time budget via max-margin trees, *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)* (2012), 1011–1022.

[35] D. Reinsel, J. Gantz, and J. Rydning, The Digitization of the World - From Edge to Core, Technical report, International Data Corporation (2018).

[36] S. Ruder, P. Ghaffari, and J. G. Breslin, *Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution* (2016).

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, ImageNet Large Scale Visual Recognition Challenge, *IJCV* **115**, 3 (2015), 211–252.

[38] M. Seeger, PAC-Bayesian generalization error bounds for Gaussian process classification, *Journal of Machine Learning Research* **3** (2002), 233–269.

[39] P. Shrestha, S. Sierra, F. A. Gonzalez, P. Rosso, and T. S. Manuel Montes-y Gomez, Convolutional neural networks for authorship attribution of short texts, *Proceedings of the 15'th Conference of the European Chapter of the Association for Computational Linguistics* (2017), 669–674.

[40] N. Srebro and T. S. Jaakkola, Weighted low-rank approximations, *Proceedings of the 20'th International Conference on Machine Learning (ICML)* (2003), 720–727.

[41] E. Stamatatos, A survey of modern authorship attribution methods, *Journal of the American Society for Information Science and Technology* **60**, 3 (2009), 538–556.

[42] M. Stavngaard, A. S. rensen, S. Lorenzen, N. Hjuler, and S. Alstrup, Detecting Ghostwriters in High Schools, *Proceedings 27'th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)* (2019), 197–202.

[43] M. Stavnsgaard and A. Sørensen, *Authorship Verification - Deep Learning Based Methods for Authorship Verification* (2018). Master's thesis

[44] N. Thiemann, C. Igel, O. Wintenberger, and Y. Seldin, A strongly quasiconvex PAC-Bayesian bound, *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, PMLR (2017), 466–492.

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

[46] D. A. Tomar, *The Ghostwriting Business: Trade Standards, Practices, and Secrets*, https://thebestschools.org/resources/ghostwriting-business-trade-standards-

practices-secrets/. Accessed August 2019

[47] TV2, *Politisk flertal vil gøre salg af eksamensopgaver ulovligt*, nyheder.tv2.dk/politik/2017-06-21-politisk-flertal-vil-gore-salg-af-eksamensopgaver-ulovligt. Accessed August 2019

[48] L. G. Valiant, A Theory of the Learnable, *Communications of the ACM* **27**, 11 (1984), 1134–1142.

[49] Y. Wang, Q. Tang, S.-T. Xia, J. Wu, and X. Zhu, Bernoulli Random Forests: Closing the Gap between Theoretical Consistency and Empirical Soundness, *Proceedings of the 17th International Conference on Machine Learning (ICML)*, Morgan Kaufmann (2016), 2167–2173.

[50] H. Yu, C. Hsieh, Q. Lei, and I. S. Dhillon, A greedy approach for budgeted maximum inner product search, *Proceedings of the 31'st Conference on Neural Information Processing System (NIPS)* (2017), 5459–5468.

Appendix A. Papers

This appendix contains all papers included with this thesis. Each paper contains a title page with information about the paper, followed by the paper itself using original typesetting. All authors listed are affiliated with the Univeristy of Copenhagen, unless otherwise stated.

The papers included are:

A.1. On Predicting Student Performance Using Low-rank Matrix Factorization Techniques

Authors:	Stephan Sloth Lorenzen
	Ninh Pham
	Stephen Alstrup
Presented at:	European Conference on e-Learning (ECEL), Porto, Portugal
Journal:	Proceedings of the 16'th European Conference on e-Learning
Year:	2017
Pages:	326-334

On Predicting Student Performance Using Low-rank Matrix Factorization Techniques

Stephan Lorenzen, Ninh Pham, and Stephen Alstrup
Department of Computer Science, University of Copenhagen, Denmark
lorenzen@di.ku.dk
pham@di.ku.dk
s.alstrup@di.ku.dk

Abstract. Predicting the score of a student is one of the important problems in educational data mining. The scores given by an individual student reflect how a student understands and applies the knowledge conveyed in class. A reliable performance prediction enables teachers to identify weak students that require remedial support, generate adaptive hints, and improve the learning of students. This work focuses on predicting the score of students in the quiz system of the Clio Online learning platform, the largest Danish supplier of online learning materials, covering 90% of Danish elementary schools and hundred of thousands of students. In particular, we formalize our prediction task as the *weighted* low-rank matrix factorization (LRMF) problem, a very attractive problem in machine learning community due to its extensive applications in collaborative filtering. We investigate the two variants of weighted LRMF including standard weighted LRMF and weighted non-negative LRMF, and apply the Expectation-Maximization (*EM*) procedure to solve them. We also study different Singular Value Decomposition (SVD)-based initialization methods for these variants since the *EM* method is sensitive to the initial values. Experimental results in the Clio Online data set confirm that the proposed initialization methods lead to very fast convergence. Regarding the prediction accuracy, surprisingly, the advanced *EM* method is just slightly better than the baseline approach based on the global mean score and student/quiz bias. In order to understand the behaviour of the algorithm, we extract a dense subset of the data set and visualize its eigenvalue spectrum. The highly skewed eigenvalue spectrum of such subset explains our interesting findings. We conclude that since the active students in the platform perform very similar and the current version of the data set is very sparse, the very low-rank approximation can capture enough information. This means that the simple baseline approach achieves similar performance compared to other advanced methods. In future work, we will restrict the quiz data set, e.g. only including quizzes with a time limit, considering several quiz types. We expect that students will behave differently and the advanced *EM* methods might improve the prediction accuracy.

Keywords. Predicting student performance, collaborative filtering, matrix factorization.

1. Introduction

Prediction of student performance, estimating the unknown score of a given task, is an important problem in educational data mining (Elbadrawy et al. 2016; Meier et al. 2016; Ayala 2014; Thai-Nghe et al. 2010). The scores given by an individual student reflect how a student understands and applies the knowledge conveyed in class. Accurate prediction of such scores enables teachers to provide remedial support to weak students and to recommend appropriate tasks to excellent students.

In this work, we consider data from the Danish online learning platform Clio Online (Clio Online n.d.) is currently the largest Danish supplier of online learning materials, covering 90% of Danish elementary schools and hundreds of thousands of students. Their platform includes texts, videos, quizzes, exercises, and more, spanning several different elementary school subjects. We study the performance of students on the quizzes; specifically we will focus on predicting how a student perform on an unseen quiz. More formally, given n students $S = \{s_1, ..., s_n\}$, and a set of m quizzes $Q = \{q_1, ..., q_m\}$, any student s_i will answer some quizzes in Q and we need to predict the score of the other quizzes that he has not finished yet. In other words, given an *incomplete* matrix X of size n x m reflecting the scores of n students over m quizzes, our task is to efficiently complete such matrix X given its partial known values.

Matrix completion (Candes and Recht 2012) has recently become a very attractive problem in the machine learning community due to its extensive applications in collaborative filtering (Goldberg 1992). The objective of this task is filling in missing entries in a partially observed matrix. One example is the movie-rating matrix in the Netflix problem (Netflix Price n.d.) where rows and columns correspond to users and movies, respectively, and entries are the ratings. Filling in a missing entry corresponds to estimating the unknown rating of the corresponding {user, movie} pair.

If we do not restrict the number of degrees of freedom in the completed matrix, this problem is under-determined since we can assign any arbitrary values to the missing entries. Therefore, one often tries to estimate a low-rank matrix from the sparse, observed matrix, i.e. a matrix of rank r that matches the known entries best. In the Netflix problem, the ratings matrix is expected to be low-rank since one can use a few latent factors to reflect the tastes and movie preferences of users (Koren, Bell and Volinsky 2009). Therefore, we can cast the matrix completion problem as the *weighted* low-rank matrix factorization (LRMF) (Srebro and Jaakkola 2003) where the weights are *1s* for observed entries and *0s* for missing entries. More formally, given an incomplete matrix X of size $n \times m$, a *binary* weight matrix W of size $n \times m$ and an integer $r > 0$. The weighted LRMF, with respect to the Frobenius norm, would like to find the low-rank matrices U of size $n \times r$, and V of size $r \times m$, such that $\| W \otimes (X - UV) \|_F$ is minimum where $\otimes$ is the entrywise product (Hadamard product). The missing entry X_{ij} can simply be estimated by $(UV)_{ij}$. It is worth noting that by adding the non-negative constraint to the matrices U and V, i.e. all values in U, and V are non-negative, we have the weighted non-negative LRMF problem. The weighted non-negative LRMF has been successfully used in many collaborative prediction tasks (Koren, Bell and Volinsky 2009; Lee and Seung 2000).

We view our problem as the weighted LRMF since we represent the observed scores of students as an incomplete matrix X. Furthermore, this is also due to the fact that we can assume that there are a small number of latent features revealing the students and tasks preferences. Such assumption is natural and has been widely used in research and application work in educational data (Barnes 2005; Desmarais 2011; van der Linden and Hambleton 2010). It is worth noting that there are prior work using matrix factorization for predicting student performance (Desmarais 2011; Elbadrawy et al. 2016; Thai-Nghe et al. 2010). Since matrix factorization in general requires iterative methods (Koren, Bell and Volinsky 2009; Lee and Seung 2000; Srebro and Jaakkola 2003) to solve the non-convex optimization, the initialization stage is essential in the design of successful methods. Unsuitable initial values can result in either bad local minima or slow convergence.

Contribution

Our contribution is as follows.

- We formalize the problem of predicting the scores of students from their partial observed scores as the weighted LRMF problem. We study the well-known Expectation-Maximization procedure (*EM*) (EM algorithm n.d.) for solving it.
- We investigate the *non-negative* constrained problem, i.e. all entries of the estimated low-rank matrices *U*, *V* are non-negative, and make use of the *EM* method for solving the non-negative constrained problem.
- Since the behaviour of the *EM* method is sensitive to the initial values, we propose using the singular value decomposition (SVD) (Golub and Loan 1999) and the non-negative double SVD (Boutsidis and Gallopoulos 2008) as the initialization stages for the standard weighted and non-negative weighted LRMF, respectively. Experimental results show that the proposed initialization methods lead to fast convergence ratio for both constrained and non-constrained problems.
- We implement and measure the performance of the proposed methods on new real-life data from a Clio Online learning platform. We compare the mean squared error of the *EM* method with the simple baseline approach based on the global mean score and student/quiz bias. Surprisingly, the advanced *EM* method is only slightly better or comparable to the baseline approach. We visualize the eigenvalue spectrum of a dense subset of the data set to explain our interesting findings.
- We conclude that since the *active* students in the platform perform very similarly and the current version of the data set is very sparse, the very low-rank approximation can capture enough information. This means that the simple baseline approach (rank at most 2) achieves similar performance compared to other advanced methods. We believe that by restricting the quiz data set, e.g. only including quizzes with a time limit, students will behave differently and the advanced *EM* methods might improve the prediction accuracy.

2. Related Work

Depending on the available information of students, different data mining techniques can be used to tackle this issue. Koprinska, Stretton, and Yacef 2015; Strecht et al. 2015 made use of classification and regression techniques over characteristics, activities and historic assessments of students in order to predict their scores. Such methods demand significant meta-data about students and tasks.

Other work take advantages of collaborative filtering techniques since these solutions require no knowledge of students and tasks, and therefore avoid the need for extensive data collection. Bydzovska 2016 exploited the neighbourhood models and other work (Desmarais 2011; Elbadrawy et al. 2016; Thai-Nghe et al. 2010) leveraged LRMF approaches, i.e. matrix factorization on latent factor models. In spite of the simplicity, the results of such methods indicate that there is sufficient information in the historical student-task score data to make the prediction feasible.

3. Our Approach

We consider the problem of predicting the scores of students based on some of their observed scores as the weighted LRMF since the set of known scores can be represented as an incomplete matrix. We study both the standard weighted LRMF and non-negative weighted LRMF problems. In general, weighted LRMF problems require non-convex optimization solvers and do not admit a closed-form solution. We study the popular Expectation-Maximization (*EM*) algorithm (Srebro and Jaakkola 2003; Zhang et al. 2006) for these two variants since it can handle the non-negative constraint. We also study the SVD-based initialization methods (Boutsidis and Gallopoulos 2008; Srebro and Jaakkola 2003) for the *EM* methods.

The baseline approach (mean/bias)

Similar to the Netflix solution (Koren, Bell and Volinsky 2009), we investigate the role of student/quiz bias or intercept in prediction. Given the incomplete matrix *X* of size *n x m* where the row/column corresponds to the student/quiz and the corresponding weighted matrix *W* of size *n x m*, $W_{ij} \in \{0, 1\}$, we define the global mean score and student/quiz bias as follows.

- Denote by μ the average over all known values of *X*, i.e. $\mu = \sum_{ij} X_{ij} / \sum_{ij} W_{ij}$. It is clear that μ is the average performance of all students.
- Define the *student (row) bias* α_i and *quiz (column) bias* β_j as follows:

$$\alpha_i = \frac{\sum_{j=1}^{m} \mathbf{W}_{ij}\mathbf{X}_{ij}}{\sum_{j=1}^{m} \mathbf{W}_{ij}} - \mu \ ,$$

$$\beta_j = \frac{\sum_{i=1}^{n} \mathbf{W}_{ij}\mathbf{X}_{ij}}{\sum_{j=1}^{m} \mathbf{W}_{ij}} - \mu \ .$$

α_i describes how much better/worse the student s_i is compared to the average student, while β_j describes how much easier/harder the quiz q_j is compared to the average quiz.

The natural assumption of the baseline approach is that the student performance is consistent over all kinds of quizzes. In other words, if any student is good/bad, he will be good/bad at *all* quizzes. Therefore, we can simply predict the score of a given student s_i on the quiz q_j as $\mu + \alpha_i + \beta_j$. Define the *bias matrix B* with entries $B_{ij} = \mu + \alpha_i + \beta_j$, the complete matrix will be

$$\mathbf{X} \leftarrow \mathbf{W} \otimes \mathbf{X} + (\mathbf{1}_{n \times m} - \mathbf{W}) \otimes \mathbf{B}$$

where $\mathbf{1}_{n \times m}$ is the matrix of size *n x m* filled with 1s.

The EM approach

The *EM* method alternates between the *E*xpectation step, in which values from the current estimate are filled in for the missing values in *X*, and the *M*aximization step, in which *X* is re-estimated as a low-rank approximation. These two steps are repeated until the rate of convergence decreases. For simplicity, we denote by $X^{(0)}$ the original incomplete matrix *X*.

For the expectation step, we make use of the bias matrix *B* in the initialization stage. In particular, we impute missing entries of $X^{(0)}$ by entries in *B* as shown in the baseline approach. Depending on the non-negative constraint, we use different SVD-based approaches to initialize $U^{(0)}$ and $V^{(0)}$.

Standard weighted LRMF: Init-step and E-step

We compute the truncated SVD of rank r of the imputed matrix X as initial values and the E-step is straightforward.

$$\begin{aligned}
&\texttt{Init-step:}\\
&[\mathbf{U_r}, \mathbf{\Sigma_r}, \mathbf{V_r}] \leftarrow SVD_r\left(\mathbf{W} \otimes \mathbf{X}^{(0)} + (\mathbf{1}_{n\times m} - \mathbf{W}) \otimes \mathbf{B}\right)\\
&\left[\mathbf{U}^{(0)}, \mathbf{V}^{(0)}\right] \leftarrow [\mathbf{U_r}\mathbf{\Sigma_r}, \mathbf{V_r}]\\
&\texttt{E-step:}\\
&\mathbf{X}^{(1)} \leftarrow \mathbf{W} \otimes \mathbf{X}^{(0)} + (\mathbf{1}_{n\times m} - \mathbf{W}) \otimes \left(\mathbf{U}^{(0)}\mathbf{V}^{(0)}\right)
\end{aligned}$$

Weighted non-negative LRMF: Init-step and E-step

Since SVD can introduce negative values, we use of a modified SVD called the non-negative double SVD (NNDSVD) (Boutsidis and Gallopoulos 2008) with the rank *r* in the initialization stage. NNDSVD utilizes an algebraic property of unit rank matrices to approximate positive sections of the partial SVD factors of the data matrix. It provides fast convergence and small approximation error for many non-negative matrix factorization algorithms. Here, we apply NNDSVD on the imputed matrix *X* as initial values. The E-step is identical to the standard case.

$$\begin{aligned}
&\texttt{Init-step:}\\
&\left[\mathbf{U}^{(0)}, \mathbf{V}^{(0)}\right] \leftarrow NNDSVD_r\left(\mathbf{W} \otimes \mathbf{X}^{(0)} + (\mathbf{1}_{n\times m} - \mathbf{W}) \otimes \mathbf{B}\right)\\
&\texttt{E-step:}\\
&\mathbf{X}^{(1)} \leftarrow \mathbf{W} \otimes \mathbf{X}^{(0)} + (\mathbf{1}_{n\times m} - \mathbf{W}) \otimes \left(\mathbf{U}^{(0)}\mathbf{V}^{(0)}\right)
\end{aligned}$$

For the maximization step, we make use of the truncated SVD with rank *r* for the standard weighted LRMF since it is the optimal solution (Golub and Loan 1999; Srebro and Jaakkola 2003). For the non-negative case, we use the popular multiplicative update rules (Lee and Seung 2000) in place of the truncated SVD procedure.

Standard weighted LRMF: M-step

$$\begin{aligned}
\texttt{M-step:}\quad &\left[\mathbf{U_r}^{(t+1)}, \mathbf{\Sigma_r}^{(t+1)}, \mathbf{V_r}^{(t+1)}\right] \leftarrow SVD_r\left(\mathbf{X}^{(t)}\right),\\
&\left[\mathbf{U}^{(t+1)}, \mathbf{V}^{(t+1)}\right] \leftarrow \left[\mathbf{U_r}^{(t+1)}\mathbf{\Sigma_r}^{(t+1)}, \mathbf{V_r}\right].
\end{aligned}$$

Non-negative weighted LRMF: M-step

$$\begin{aligned}
\texttt{M-step:}\quad &\mathbf{V}^{(t+1)} \leftarrow \mathbf{V}^{(t)} \otimes \frac{\left(\mathbf{U}^{(t)}\right)^\top \mathbf{X}^{(t)}}{\left(\mathbf{U}^{(t)}\right)^\top \mathbf{U}^{(t)}\mathbf{V}^{(t)}},\\
&\mathbf{U}^{(t+1)} \leftarrow \mathbf{U}^{(t)} \otimes \frac{\mathbf{X}^{(t)}\left(\mathbf{V}^{(t+1)}\right)^\top}{\mathbf{U}^{(t)}\mathbf{V}^{(t+1)}\left(\mathbf{V}^{(t+1)}\right)^\top}.
\end{aligned}$$

We note that the multiplicative update rules might not converge to a stationary point if the initial values $U^{(0)}$ and $V^{(0)}$ have some zero row/column (Lin 2007). We observe that the factorization given by NNDSVD might provide zero row/column values. Hence we modify the update rules for the M-step as suggested by Lin 2007.

4. Experiments

Following the approach described above, we have tested the *EM* algorithm with and without the non-negativity constraint.
For short notation, we use *EM-NMF* and *EM-MF* for the non-negative case and the standard case, respectively. Furthermore, we have experimented with using (NND-)SVD for initializing *U* and *V*, with the goal of reducing the number of required iterations. All tests have been run on the data from Clio Online. The bias matrix *B* in itself turns out to be a good predictor for the missing students' scores of X; hence we have used it as a baseline comparison for the *EM* algorithms.

Data

A quiz in the Clio Online system consists of several pages, each of which may contain one of several types of tasks, e.g. multiple choice questions or gap-fill tasks. When the student has finished all tasks, they are awarded a score for the quiz. We pre-process the data such that all scores are between 0 and 1, 0 being all questions wrong and 1 being all questions right. Furthermore, all repeated attempts at the same quiz are removed, so that only the first attempt remains. Thus, the data consists of a list of tuples *(i,j,v)*, where $v \in [0,1]$ is the score of student *i* on quiz *j*, and combinations of *i* and *j* are unique.
While very large, the full data set is also very sparse; thus we extract a subset of the data such that for each student s_i, the set contains at least 15 tuples for that student, and for each quiz q_j, the set contains at least 15 tuples for that quiz. This means that each student/quiz has at least 15 answers. Correspondingly, the matrix *X* with entries $X_{ij} = v$ has at least 15 known values in each row and column. The resulting data set contains data for $n = 1141$ students and $m = 245$ quizzes. The data set is split into a training and a test set by sampling data points for the test set with probability 0.25. Table 1 shows the resulting size and density of each data set, *n* being the number of students and *m* the number of quizzes.

Experimental setup

For each of the two methods, *EM-MF* and *EM-NMF*, we run the method on the training data in order to obtain *U* and *V*, which admits the approximation *UV* of *X*. Having obtained the model, we then measure the training/test error *e*, given by the *root mean square error* (RMSE) on the training/test set *S*:

$$e = \sqrt{\frac{1}{|S|} \sum_{(i,j,v)\in S} (v - (\mathbf{UV})_{ij})^2} \ .$$

We test each method for *r = 1,2,...,6*. For each *r*, we perform the following experiments:

- The method is run with random initialization of *U* and *V*. The method is run 10 times, and the average training/test errors are reported, together with the average number of iterations.
- The method is run with *U* and *V* initialized using the mean/bias method. Since the initialization/method is deterministic, we only report the training/test error and iterations for a single run.

The results for *EM-MF* can be found in Table 3, while the results for *EM-NMF* can be found in Table 4. As mentioned, we use the bias matrix *B* as the baseline. The results from using *B* as the approximation is given in Table 2.

e_{train}	e_{test}
0.15760	0.17681

Table 1: Data overview. *n = 1141*, *m = 245.*

Data set	Size	Density
Training set	16206	(5.80%)
Test set	5453	(1.95%)

Table 2: Training/test error when using the bias matrix.

	Random init.			Bias init.		
r	e_{train}	e_{test}	#it.	e_{train}	e_{test}	#it.
1	0.15918	**0.17891**	68.2	0.15606	**0.17512**	8
2	0.14845	0.18410	63.7	0.14171	0.17626	25
3	0.13658	0.19770	78.2	0.12613	0.18331	34
4	0.13218	0.21860	92.5	0.11437	0.18943	44
5	0.12979	0.24597	106.7	0.10323	0.19438	50
6	0.12654	0.28006	124.8	0.09153	0.19967	58

Table 3: Training/test error and iterations for *EM-MF*.

	Random init.			Bias init.		
r	e_{train}	e_{test}	#it.	e_{train}	e_{test}	#it.
1	0.18040	0.31153	69.3	0.15606	0.17512	8
2	0.16354	**0.19101**	88.7	0.15506	**0.17442**	9
3	0.16332	0.19457	98.5	0.15506	0.17442	9
4	0.16201	0.19584	112.2	0.15506	0.17442	9
5	0.16017	0.19796	131.4	0.15506	0.17442	9
6	0.15855	0.20137	152.0	0.15506	0.17442	9

Table 4: Training/test error and iterations for *EM-NMF*.

Analysis

From the results, we clearly see that the methods converge faster, when *U* and *V* are initialized using the mean/bias based methods, described in the previous section. Furthermore, both methods achieve lower training and test errors, when using the mean/bias method for initialization, compared to using random *U* and *V*. This observation is also confirmed in Figure 1, which plots the training error after each iteration for each method/initialization.

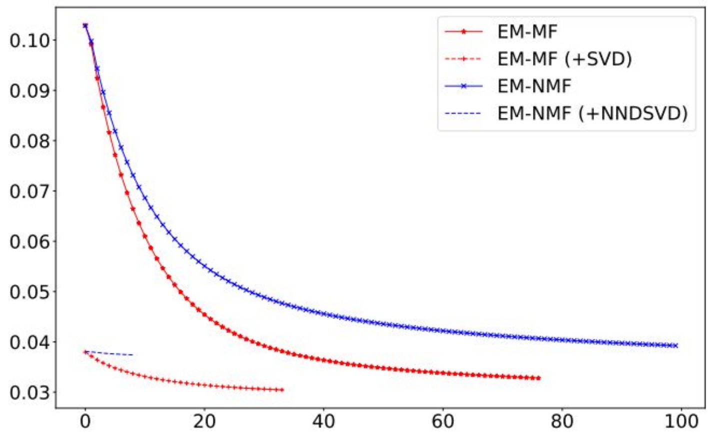


Figure 1: Convergence of the *EM-MF* and *EM-NMF* algorithms on the large data set with *r = 3* and using random initialization and SVD based initialization of *U* and *V*.

Regarding the training/test errors, we see that both methods perform better when initialized with the mean/bias method. With random initialization, *EM-MF* performs better than *EM-NMF*, but it does still not beat the baseline method, and appears to overfit for *r > 1*. When using the bias matrix initialization, both methods achieve lower training/test errors, and actually outperform the bias matrix approximation, although not to any significant degree.

This indicates that the bias matrix is quite close to a local minima for this data set. It also explains the fast convergence, since the algorithm starts quite close to the local minima. Considering the training/test errors for the random initializations, it seems unlikely that we can find a better local minima. The best results (*r = 1* for *EM-MF* and *r = 2* for *EM-NMF*) also indicates that the data has low rank. This fits with the bias matrix, which can easily be seen to have rank at most 2. The good low rank results also explain why the results for *EM-NMF* with NNDSVD initialization are identical (within the given number of decimals) for $r \geq 2$, since the extra rows/columns of *U* and *V* are initially all 0 (from the NNDSVD method), and very little extra improvement is gained by changing that.

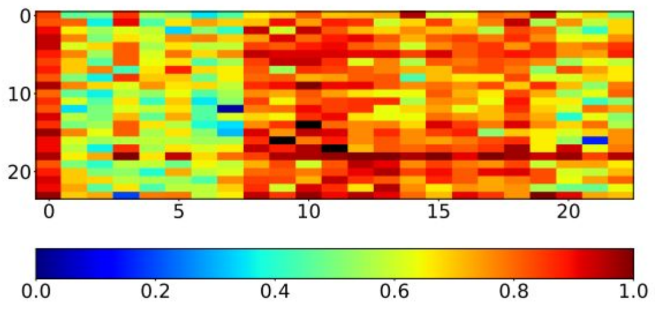


Figure 2: A small, dense subset of the data. Black tiles indicate unknown values.

Figure 2 visualizes a smaller, almost complete subset of the data, with only three unknown entries. Looking at the visualization, some row/column patterns definitely emerge, e.g. quizzes 1-7 and 19-22 appear to be more difficult than the rest. Such patterns are easily captured by the bias matrix. However, Figure 3 plots the eigenvalues for the small dense matrix of Figure 2 (with the three unknown values imputed using the bias matrix). Here we see that only the first eigenvalue is very significant (17.3 compared to 2nd, 3rd and 4th eigen values, which are close to 1), which supports the findings from the experiment, e.g. that very low rank approximations seem to be optimal for this data.

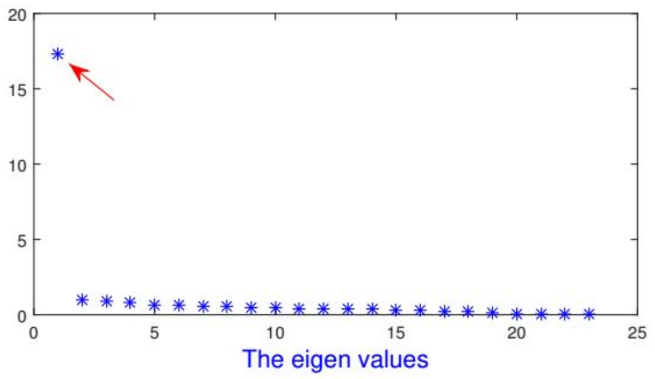


Figure 3: The eigenvalues spectrum of the small, dense subset of the data.

Discussion

Since the quizzes span several topics, we expected the approximation to have higher rank, to fully capture the different skills needed in order to obtain a good score within the different topics. However, it seems to not be the case. As the analysis shows, only few latent features are needed to obtain a good model of the data. It seems that these few latent features rather relate to the 'base' skill of a student/difficulty of a quiz, i.e. students are either good or bad at everything.

While the experiment and analysis suggests that only a single skill determines the score of a student, this may come down to the sparsity of the data. It may be that good, high rank approximations actually exists, but does not show up in the experiment due to the low number of answers for many quizzes.

5. Conclusions

We have successfully applied Expectation-Maximization matrix factorization methods, both the unconstrained version and the non-negative constrained version, on the quiz data from Clio Online. Experiments indicate that matrix factorization does not improve significantly on the results obtained by the simpler method based on the global score mean and student/quiz biases, and that only a few latent factors are needed in the optimal approximation. As a result, the number of iterations can be significantly reduced by initializing the methods by using the bias matrix, when compared to using a random initialization.
The fact that very low rank approximations gave better accuracy, together with further examination of a smaller, almost complete subset, indicates that the Clio Online data set can be described by a very low rank matrix. Thus it may be difficult to improve upon the mean/bias method, by use of advanced matrix factorization, at least when the data is sparse.
With regards to interpreting the results, our hypothesis is that the few latent features relate to the skill of the students/difficulty of the quiz. If this hypothesis holds, it provides useful information for e.g. recommending the appropriately difficult quizzes to students.

Future work

In regards to the Clio Online data, it may be interesting to apply other matrix factorization techniques. Adding regularization may also improve the prediction. It could also be interesting to include more meta data about the quizzes from the Clio Online system; e.g. we expect that only considering quizzes with a time limit may admit a higher rank approximation. Furthermore, it could also be interesting to apply the techniques tested in this paper to quiz/test scores from other sources.

6. Acknowledgments

The work is supported by the Innovation Fund Denmark through the Danish Center for Big Data Analytics Driven Innovation (DABAI) project. The authors would like to thank members of the DABAI project for helpful discussions.

7. References

Barnes, T. (2005) The Q-matrix method - mining student response data for knowledge. In American Association for Artificial Intelligence 2005 Educational Data Mining Workshop, pp 1—8.
Boutsidis, C., and Gallopoulos, E. (2008) SVD based initialization: A head start for nonnegative matrix factorization. Pattern Recognition, Vol. 41, No. 4, pp 1350—1362.
Strecht, P., Cruz, L., Soares, C., Mendes-Moreira, J., and Abreu, R. (2005) A comparative study of regression and classification algorithms for modelling students' academic performance. In Proceedings of the 8th International Conference on Educational Data Mining (EDM), pp 392—395.
Candes, E. J., and Recht, B. (2012) Exact matrix completion via convex optimization. Communication ACM, Vol. 55, No. 6, pp 111 – 119.
Desmarais, M. (2011) Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization. In Proceedings of the 4th International Conference on Educational Data Mining, pp 41—50.
Elbadrawy, A., Polyzou, A., Ren, Z., Sweeney, M., Karypis, G., and Rangwala, H. (2016) Predicing student performance using personalized analytics, Computer, Vol. 49, No. 4, pp 61—69.
Goldberg, D., Nichols, D. A. ,Oki, B. M. , and Terry, D. B. (1992) Using collaborative filtering to weave an information tapestry. Communication ACM, Vol. 35, No. 12, pp 61—70.
Golub, G. H., and Loan, C. F. V. (1996) Matrix computations (3. ed.), Johns Hopkins University Press.
Meier, Y., Xu, J., Atan, O., and van der Schaar, M. (2016) Predicting grades, IEEE Trans. Signal Processing, Vol. 64, No. 4, pp 959—972.
Ayala, A.P. (2014). Review educational data mining: A survey and a data mining-based analysis of recent works, Expert system application, Vol. 41, No. 4, pp 1432—1462.
Srebro, N., and Jaakkola, T. S. (2003) Weighted low-rank approximations. In Proceedings of the Twentieth International Conference (ICML 2003), pp 720—727.
Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., and Schmidt-Thieme, L. (2010) Recommender system for predicting student performance. In Proceedings of the 1st workshop on recommender systems for technology enhanced learning, pp 2811—2819.
van der Linden, W. J., and Hambleton, R. K. (2010) Handbook of Modern Item Response Theory, Springer-Verlag, New York.

Zhang, S., Wang, W., Ford, J., and Makedon, F. (2006) Learning from incomplete ratings using non-negative matrix factorization. In Proceedings of the Sixth SIAM International Conference on Data Mining (SDM), pp 549—553.
Clio Online 2017, Clio Online Home Page. Available from: http://www.clioonline.dk/. [24 May 2017].
EM algorithm, EM algorithm description. Available from: https://en.wikipedia.org/wiki/Expectation-maximization_algorithm. [24 May 2017].
Netflix Prize 2007, Netflix Prize. Available from: https://en.wikipedia.org/wiki/Netflix_Prize. [24 May 2017].
Koren, Y., Bell, R.-M., AND Volinsky, C. (2009) Matric factorization techniques for recommender system, IEEE COMPUTER, Vol. 42, No. 8, pp 30 – 37.
Lee, D.-D., and Seung, H.-S. (2000) Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing System (NIPS), pp 556—562.
Koprinska, I., Stretton, J., and Yacef, K. (2015) Students at risk: Detection and remediation. In Proceedings of the 8th International Conference on Educational Data Mining (EDM), pp 512—515.
Bydzovska, H. (2016) A comparative analysis of techniques for predicting student performance. In Proceedings of the 9th International Conference on Educational Data Mining (EDM), pp 306—311.
Lin, C.-J. (2007) On the convergence of multiplicative update algorithms for nonnegative matrix factorization. Trans. Neur. Netw., Vol.18, No. 6, pp 1589—1596.

A.2. DABAI: A data driven project for e-Learning in Denmark

Authors:	Stephen Alstrup
	Casper Hansen
	Christian Hansen
	Niklas Hjuler
	Stephan Sloth Lorenzen
	Ninh Pham
Presented at:	European Conference on e-Learning (ECEL), Porto, Portugal
Journal:	Proceedings of the 16'th European Conference on e-Learning
Year:	2017
Pages:	18-24

DABAI: A data driven project for e-Learning in Denmark

Stephen Alstrup, Casper Hansen, Christian Hansen, Niklas Hjuler, Stephan Lorenzen, and Ninh Pham

Department of Computer Science, University of Copenhagen, Denmark
s.alstrup@di.ku.dk
bnq@di.ku.dk
chrh@di.ku.dk
hjuler@di.ku.dk
lorenzen@di.ku.dk
pham@di.ku.dk

Abstract: A new Big Data research team called DABAI have been launched in Denmark, which aims at integrating cutting edge computer science research from machine learning, algorithms and visualization into the education sector. The educational part of the DABAI project is a cooperation between Danish universities and multiple enterprises providing e-Learning solutions for the Danish market. The companies' services cover over 90% of the Danish schools, with more than one million students, who on a daily basis do millions of exercises and interactions using the involved companies' solutions. The study presented in this paper is an initial investigation of the needs of the three largest companies in e-Education in Denmark directly involved in DABAI, as well as other companies, with the goal being to continue providing novel and high-demand features for their customers. The three companies are MaCom, Clio Online, and Edulab. Clio Online together with Edulab provide an online platform for teaching material and exercises for the primary school level covering all subjects. MaCom provides a lecture management system used by most Danish high schools. Overall the study shows that the problems encountered at the different companies are varied, but can be categorized into three general sub categories: Student Profiling, Content Profiling, and Content Recommendation. Some problem types fall into multiple sub categories, and in general to accomplish the goal of providing e-Learning of the highest quality, research into all of them is necessary. This paper presents the fundamental problems in e-Learning. For each encountered problem, we describe its objectives and challenges in detail, followed by the current state of the art for solving it.

Keywords: e-Learning, e-Learning challenge categorization, Big Data

1. Introduction

A new Big Data research team called Danish Center for Big Data Analytics driven Innovation (DABAI) has been launched in Denmark, which aims at integrating cutting edge computer science research from machine learning, algorithms, and visualization into the domains encompassing Food Supply Chain Data, Societal Data, and Education Data. DABAI is funded by Innovation Fund Denmark, with a total funding of 17M euro. In this paper we will present our initial finding for problems in the educational sector. The educational part of the DABAI project is a cooperation between Danish universities and multiple enterprises providing e-Learning solutions for the Danish market. The companies' services cover over 90% of the Danish schools, with more than one million students, who daily do millions of exercises using the involved companies' solutions. In Denmark there is a strong political focus and funding to ensure the education system being more data driven, and e-Learning tools developed being integrated quickly. Such a hard political push on the software level is now possible, since it has been ensured that the schools have the necessary hardware level over the past years.

This work is an initial investigation of the needs of the three largest companies in e-Education in Denmark directly involved in DABAI, with the goal being to continue providing novel and high-demand features for their customers. The three companies are MaCom, Clio Online, and Edulab. Clio Online together with Edulab provide an online platform for teaching materials and exercises for the primary school level covering all topics. MaCom provides a lecture and assignment management system used by most Danish high schools.

Overall our study shows that the problems encountered at the different companies are very diverse, but can be categorized into three general sub categories: Student Profiling, Content Profiling, and Content Recommendation. Some problem types fall into multiple sub categories, and in general, to accomplish the goal of providing e-Learning of the highest quality, research into all of them is necessary.

In this paper we present the fundamental problems in e-Learning, and for each encountered problem we describe its objectives and challenges in detail, followed by the current state of the art for solving it. We strongly believe that, even though this work is done in Danish context, the encountered problems and solutions generalize worldwide.

1.1 Introduction of companies

In this section we will briefly introduce the three companies we are working with, such that the challenges, they are facing, can be better understood in the setting they are going to be used in.

Clio Online is a complete digital learning platform for the primary school level, and aims at offering everything a teacher and student need. They offer all subjects except mathematics, which Edulab provides. The material is tailored to the national targets for education, such that the material always follows the national curriculum for each subject. They want the teachers to be able to focus more on teaching, and less on preparing and finding the right material.

Edulab is a complete digital learning platform for mathematics, with the goal of making every child in the world better at mathematics. Edulab's products offer many different ways of learning mathematics, ranging from question heavy workloads to video and text lessons, as well as other activities depending on if the student is in class or at home.

MaCom provides *Lectio*, which is a lecture management system used in high schools. They provide a tool to manage the lessons carried out, and in a way that is transparent to the students and the parents. They also provide a tool to manage homework and for handing in assignments, giving them a big repository of student work.

Together these companies service over 90% of all schools in Denmark.

1.2 Categorization of e-Learning challenges

One of the end goals for educational software is obtaining differentiable teaching materials fitted to each student’s needs, such that the overall progression of the students is as high as possible. The current generation of educational software offers personalization options, but not as in depth compared to what a real human personal tutor could provide, and bridging this gap is one of the major challenges for current educational software providers. However, this task is complex and can be split in many subtasks, which also independently can provide great value. Our cooperation with the three largest e-Learning companies in Denmark have revealed that their needs can be put into three categories:

1. Student Profiling
2. Content Profiling
3. Content Recommendation

Student profiling is the broadest of the categories and cover modelling of students' skills, knowledge, and/or behaviour. These can be modelled by how the students interact with the educational systems, and how they perform in these systems. We present two cases on student profiling, one from modelling how students interact with the website and material, and another on modelling a student's writing style.

Content profiling is the task of understanding the material provided by the educational system, the understanding can be in the case of finding material that students generally perform much worse on than expected, or finding relations between material necessary for building solid material recommender systems. We present a case on content profiling with the task of finding similarities among a large number of quizzes.

Content Recommendation is in its essence the combination of student and content profiling, where the task is to recommend the right learning material to the student. A detailed profiling of both students and content can be used for creating optimal learning paths for the individual students. We present three cases on this category from improving personalized learning, to predicting student performance on quizzes based on other students' performance, and a system to train specific parts of a subject curriculum.

2. Company faced e-Learning challenges

In this section we will present our initial work and related literature on six e-Learning challenges faced by the companies.

2.1 Optimize e-Learning personalization

E-Learning personalization refers to an online educational approach that provides differentiated instruction to support the individual student's need. Currently, EduLab is deploying and developing an online learning platform called **"*SuperTrainer*"** to optimize the e-Learning personalization for individual student on learning Mathematics. In a nutshell, SuperTrainer can be seen as a specific recommender system that recommends *"the best question at the right time"* to students to optimize their mathematical learning process in primary school. Such adaptive recommender system provides a sequence of math questions in order to maximize student's learning gain while taking into account the limited amount of time and number of questions. One traditional mechanism is that students will be assigned easier/harder questions if they answer incorrectly/correctly on previous questions. This is due to the argument in psychology (Berlyne, 1960) and neuroscience (Gottlieb et al., 2013) that the human brain shows intrinsic pleasure in activities of optimal difficulty, i.e. balance between interest (not too difficult) and challenge (not too easy).

While traditional recommender systems are focusing on "one-shot" recommendations, e-Learning personalization investigates a *"personal path"* through the questions and adjusts this path depending on how students progress in learning. Therefore, it is essential to have a systematic measurement of individual students' performance when using personalized products. Currently, to the best of our knowledge, there is no measurement of students' performance when using e-Learning personalization. This means that we do not know how well we could do with e-Learning personalized platforms. Furthermore, there is no efficient mechanism with guarantees to recommend "*the best question at the right time"* to students.

To handle recent problems of e-Learning personalization, we aim to introduce and formalize important progress factors to measure students' performance, and then exploit such measurements to evaluate and improve an e-Learning personalization mechanism. In particular, we decompose students' progress into three factors as follows:

- **Ability of understanding:** Given a fixed number of questions, for each student we want to *maximize* the correctness/difficulty level of the given questions.
- **Learning speed:** Given a fixed amount of questions, we want to *minimize* the total amount of time that each student has to spend on solving them.
- **Balance of challenge and interest:** Given a fixed amount of questions, we want to maintain a well-defined correct ratio to balance between the *challenge* (i.e., student feels difficult to answer correctly) and the *interest* (i.e., student feels happy when answering correctly).

After formalizing students' performance measurements, our goal is to model the e-Learning personalization problem, i.e., recommending the best question at the right time to students, as an optimization problem. Following very recent machine learning approaches (Lopes et al., 2015; Tekin, Braun, and van der Schaar, 2015), we are investigating the *"Multi-Arm bandit models"* for our optimization. Assume that an e-Learning personalization has to recommend a sequence of questions to each individual student. Each question will give the student a reward value that highly depends on the student's level at that time. Note that such reward value must express the three progress factors above so that we are able to measure the student's progress. Then, we develop the optimization algorithms to maximize the sum of rewards earned by each student through a sequence of recommended questions. Our collaboration with EduLab will deploy a scalable and efficient algorithm for optimizing their e-Learning personalization platform – SuperTrainer – since EduLab is now dealing with very large-scale data, i.e., millions of questions answered every day.

2.2 Student behavior modelling

Most learning platforms allow a student to access the material through many different paths. An example could be a video lesson which could be given as homework, it could come up as recommended material to an exercise if the student needs help solving it, or the student could simply find the video lesson themselves because they want to review the material without being asked to do it. Edulab is interested in the paths users take for two primary reasons: 1) as for every other company providing a web solution, knowledge of user behavior can be used directly when planning further extensions of the system. 2) Educational systems are built based on knowledge of didactics which guide the usage of the system, and unintended usage of the system can therefore in some cases correspond to sessions with suboptimal learning for the user.

The number of daily user sessions is too large to make any meaningful qualitative study of the individual user sessions, where session in this context is defined as the interactions a user does from logging into the system to the time they log out. It is therefore necessary to do some clustering of sessions and analyze the resulting clusters, to extract general trends. A common approach (Köck, M. and Paramythis, A. 2011) for modelling the sessions, is to consider them as a sequence over an action space. A simple action space is that the user either answers a quiz (Q) or watches a video (V). A user session would then consist of entering the system, a sequence of actions, e.g. QQQVVQ, and then logging off. A clustering over such a simple space, would still allow us to ask questions like, does students in general start with videos or quizzes, and are the user interactions very binary such that they either answer questions or watch lessons? This is a very trivial example of an action space, but it can be made arbitrarily complex according to the system we wish to model, with the limitation being that larger action spaces require more data to find meaningful insights. The method of considering user sessions as sequences over an action space have been used with success in deriving insights from educational systems (Klingler, S. et al. 2016; Faucon, L., Kidzinski, L., and Dillenbourg, P. 2016).

We are currently in the process of clustering all sessions done in Edulab's system on multiple different state spaces with the focus on finding "unproductive" sessions, where the students use the system in an unintended way. An initial study on a subset of the available data have been done (Hansen, C. et al. 2017), where the clustering lead to insight into a significant number of sessions where students had very binary behavior, meaning they either always watched lessons, answered questions correctly or answered wrongly. This study used an action space of considering video lesson, if a question was correctly and incorrectly answered, and if the student switched to material of different topics. The overall goal of this work is to provide Edulab with a new tool for future development of their system.

2.3 Predicting student performance

Prediction of student performance, estimating the unknown score of a given task, is one of the important problems in e-Learning. The scores given by an individual student reflect how a student understands and applies the knowledge conveyed in class. A reliable performance prediction of such scores enables teachers to provide remedial support for weak students, recommend appropriate tasks to excellent students, generate adaptive hints, and improve the learning of students. This section focuses on predicting the score of students in the quiz system of the Clio Online learning platform, where a student has answered only a subset of the quizzes.

Currently, Clio Online learning platform is maintaining a quiz system that contains thousands of quizzes with several types, including multiple choice quizzes, gap-filling quizzes, etc., spanning several different elementary school subjects. We study the performance of students on the quiz data; specifically we will focus on predicting how a student performs on an unseen quiz. More formally, given n students $S = \{s_1, ..., s_n\}$, and a set of m quizzes $Q = \{q_1, ..., q_m\}$, any student s_i will answer some quizzes in Q and we need to predict the score of the other quizzes that he has not finished yet. In other words, given an incomplete matrix X of size $n \times m$ reflecting the scores of n students over m quizzes, our task is to efficiently complete such matrix X given its partial known values. Since we can view a student/quiz as a user/item, we view our prediction task as the *collaborative filtering problem* (Collaborative Filtering, 2017) and investigate state of the art techniques for solving it.

In particular, we have studied the *matrix factorization techniques* (Koren, Bell, and Volinsky, 2009; Lee and Seung, 2000) for improving our prediction. This is due to the fact that we can assume that there are a small number of latent features, e.g., skill sets, revealing the students and tasks preferences. Such assumption is natural and has been used widely in research and application work in educational data (Barnes, 2005; Desmarais, 2011). It is also worth noting that these solutions require no knowledge of students and tasks, and therefore avoid the need for extensive data collection. Prior work using matrix factorization for predicting student performance (Elbadrawy et al., 2016; Thai-Nghe et al., 2010) indicates that there is sufficient information in the historical student-task score data to make the prediction feasible.

2.4 Similarity among quizzes

Given a pair of quizzes, Quiz 1 and Quiz 2, how similar are they? There exist many qualitative ways of addressing this question and some quantitative like Pearson correlation measures and cosine similarity. Deary et al (2007) research correlation among intelligence tests and how well kids perform in different subjects. Even with the high number of answers in the Clio Online setting (millions of quiz answers), the thousands of quizzes results in many pairs of quizzes which have only a few or even no students who have taken both quizzes. This results in very high variance for the usual techniques. One could combat this by densifying the data with prediction values for the quizzes which would reduce the similarity problem to predicting student performance. The other option is to infer similarity in the sparse quizzes going through a third quiz taking advantage of the "transitive" property of objects being similar.

Formally we solve the problem of making the best prediction on normalized quiz scores (0 mean and unit variance for each quiz). The prediction we make is a weighted average of the other quizzes taken and the weights are the similarities. We require the similarities to be symmetric and non-negative, but other than that there is no restriction. This allow us to both train and check different overfitting measures to capture the "transitive property" of objects being similar.

The knowledge of similarities would give insight to the teachers. If for example a student suddenly does worse on a writing test than he usually does, one can find the similarities of more specific quizzes (like grammar quizzes) and thus pinpoint why this student has trouble with the specific writing quiz. Furthermore, taking advantage of similarities is likely to improve prediction (Khajah, Lindsey, and Mozer 2016). Chen, Gonxáles-Brenes and Tian (2016) suggest that spurious similarities could be due to the quiz requiring multiple skills.

2.5 Authorship verification

The competition for high grades in Danish secondary education (high school) is tougher than ever. This has led to an increase in fraud in written assignments. While efficient techniques for detecting simple copy-paste plagiarism exist and are deployed in Denmark (Frølich and Hansen, 2012), there have recently been an increase in students resorting to ghost-writing, i.e. handing in assignments written by someone else (e.g. a teacher, college student, or other professional). Normal plagiarism control does not detect this kind of fraud, since the assignment is original work. The problem has been highlighted recently by the emergence of so-called *paper-mills* in Denmark; online services providing academic ghost-writing (e.g. www.fixminopgave.dk), a phenomenon already seen in other pars of the world, e.g. the U.S., where paper mills have existed for several years (Tomar, 2014). MaCom wishes to combat this emerging trend by deploying a system for detecting cases of ghost-writing in Lectio. With data for more than 150,000 Danish students, including more than 15 million handed-in assignments, there is large potential.

Formally, we can define the problem of detecting ghost-writing as the *Authorship Verification Problem*: Given an author *a*, a set of texts $S = \{s_1, s_2, \ldots, s_n\}$ written by *a* and a text *x*, determine whether *a* is the author of *x*. This problem (and the related problem of Authorship Attribution) has been considered in the literature (Koppel, Schler and Bonchek-Dokow, 2007; van Halteren, 2004; Stamatatos 2009), and is in general considered challenging, since only limited data about the author is available. Most approaches take inspiration from the study of *stylometrics* and employ techniques from natural language processing and machine learning. Approaches usually follow either the *profile-based* paradigm (*x* is compared to a constructed profile of *a*) or the *instance-based* paradigm (*x* is compared to all instances s_1, s_2,...); in either case, the problem can be seen as a case of *Student Profiling*.

For the case of authorship verification in Danish high schools, there are several important considerations to be made. Most notably the fact that:

1. Students are still learning, and their writing style may change over time.
2. We have access to a large corpus of text consisting of the 15 million assignments, which may be utilized in order to improve verification.

These considerations are already investigated in two studies using the data from MaCom: (Hansen et al., 2014) considers authorship attribution with the temporal aspect, and concludes that, using SVMs, good accuracy can be obtained even when *S* only contains the most recent assignments for a student, while (Aalykke, 2016) utilizes the corpus by performing verification through a profile-based authorship attribution approach.

We hope to improve upon the previous studies by utilizing both (1) and (2), coupled with comprehensive feature selection and state-of-the-art techniques for authorship verification and outlier detection. Furthermore, we hope that our student profiling may also aid in measuring progress of the student in terms of writing style.

2.6 Curriculum trainer

Every week teachers all over the world plan lessons to teach their kids. Much of this work is overlapping between teachers since what the students need to learn is in many cases the same. Now, in the era of "sharing economy", it is possible to utilize the community of teachers via large lecture management systems, such as Lectio, which is made by MaCom. Lectio is used by almost all high schools in Denmark (90%).

Curriculum trainer is a small step in the utilization of taking advantage of "sharing economy". It is a system made for high school students. They can use it to prepare them for their exams. The system is to be implemented in Lectio.

The system is made such that all high school teachers in Denmark (who use Lectio) can pose questions directly into the system. The teacher only has to state the question, the correct answer, some wrong answers and in which subject the question is related to. When the students use the system it adaptively estimates the difficulty of the question and the skill of the student. This is done by the ELO rating from chess (Elo, 1978), where the game is between a question and the student, similar to (Pelanek, 2016) and (Antal, 2013). The student wins if the question is answered correctly and is thus rewarded with an increase in his skill score. The difficulty index of the question is then decreased. How much depend on both the skill level of the student and the difficulty index of the question. The opposite happens if the student answers incorrectly. However unlike in chess the update rules can be different for students and questions, since the skill level of a student is expected to change whereas the difficulty level of a question is expected to be static at least over shorter periods of time, like a school year. The ELO ratings are also used to make sure that students get question at the level which is optimal for their learning. The initialisation of a student's skill level is based on his/her grades in the subject. In the future, log files could be studied to personalize how difficult question should be for a student to keep motivation high. The ELO rating is preferred for its simplicity since we need to minimize the amount of information the teacher needs to give.

The possibility of using "sharing economy" in the teaching scene has huge potential for generating teaching material of high quality, since all teachers can contribute. There are of course some barriers, the most often mentioned is the fear of being judged for their work. Hence starting at a small level and share questions should help overcome this barrier and in this case anonymity could easily be granted.

3. Conclusion

The overall goal of the study was to find a wide array of interesting problems that companies providing e-Education in Denmark have, and present early work in the direction we plan to go to solve them. The uncovered problems are interesting from an educational point of view, but they also raise interesting problems from a computer science perspective, showing the importance of the cooperation between the fields for e-Learning.

None of the presented problems were novel for the e-Learning community, and there are varying levels of existing work that potentially can be used directly by the companies without further improvement. This demonstrate the potential value between industry and university cooperation in the e-Learning community, where a cooperation will help reduce the gap between current state of the art research in academia, and what solutions are currently available for students.

The goal of the educational part of DABAI is to go one step beyond bridging the gap between industry and academia. With access to data generated from more than a million students, the goal is to push the state of the art in cooperation with the companies, with the advantage of having access to extremely large amounts of real-world data to verify the methods against.

4. References

Aalykke, A. H. (2016), Computational Authorship Attribution in Danish High Schools. Master Thesis. DTU.
Antal, M., (2013) On the use of elo rating for adaptive assessment. Studia Universitatis Babes-Bolyai, Informatica 58
Barnes, T. (2005) The Q-matrix Method – Mining Student Response Data for Knowledge. In American Association for Artificial Intelligence 2005 Educational Data Mining Workshop, pp 1—8.
Berlyne, D. (1960) Conflict, arousal, and curiosity, McGraw-Hill Book Company.
Chen, Y. ,González-Brenes, J. , Tian, J. (2016) Joint Discovery of Skill Prerequisite Graphs and Student Models , Proceedings of Educational Data Mining (EDM), pp 46-53.
Collaborative Filtering 2017, Wikiversity, wiki, 22 May 2017. Available from https://en.wikipedia.org/wiki/Collaborative_filtering. [22 May 2017]
Deary, I., Strand , S., Smith, P., Fernandes, C. (2007) Intelligence and educational, Intelligence 35 (2007) pp 13–21.
Desmarais, M. (2011) Conditions for Effectively Deriving a Q-matrix from Data with Non-negative Matrix Factorization, In Proceedings of the International Conference on Educational Data Mining, pp 41—50.
Elbadrawy, A., Polyzou, A., Ren, Z., Sweeney, M., Karypus, G., and Rangwala, H. (2016) Predicting Student Performance Using Personalized Analytics, Computer Vol. 49, No. 4, pp 61—69.
Elo, A. E. (1978), The rating of chess players, past and present. B.T. Batsford, Ltd., London
Faucon, L.; Kidzinski, L. & Dillenbourg, P. (2016), Semi-Markov model for simulating MOOC students., in Proceedings of Educational Data Mining (EDM), pp. 358-363 .
Frølich, M. and Hansen, K. (2012), Efficient Plagiarism Detection. Master Thesis. DTU.
Gottlieb, J., Oudeyer, P.-Y., Lopes, M., and Baranes, A. (2013) Information-seeking, curiosity, and attention: computational and neural mechanisms, Trends in Cognitive Sciences 17, 11, pp 585–593.
Hansen, C., Hansen, C., Hjuler, N., Alstrup, S., and Lioma, C. (2017), Sequence Modelling For Analysing Student Interaction with Educational Systems. To appear in Proceedings of Educational Data Mining (EDM).
Hansen, N., Lioma, C., Larsen, B. and Alstrup, S. (2014), Temporal context for authorship attribution: a study of Danish secondary schools, in Proceedings of the 7th Information Retrieval Facility Conference (IRFC), pp 22-40.
Khajah, M., Lindsey, R., Mozer, M. (2016) How Deep is Knowledge Tracing? Proceedings of the 9th International Conference on Educational Data Mining pp 94-101
Klingler, S., Käser, T., Solenthaler, B. and Gross. M. (2016), Temporally Coherent Clustering of Student Data., in Proceedings of Educational Data Mining (EDM), pp. 102-109 .
Köck, M. & Paramythis, A. (2011), Activity sequence modelling and dynamic clustering for personalized e-learning., in User Model. User-Adapt. Interact. 21 (1-2), pp 51-97.
Koppel, M., Schler, J. and Bonchek-Dokow, E. (2007), Measuring Differentiability: Unmasking Pseudonymous Authors, Journal of Machine Learning Research Vol. 8, pp 1261-1276.
Koren, Y., Bell, R.-M., and Volinsky, C. (2009) Matrix Factorization Techniques for Recommender System, IEEE Computer, Vol. 42, No. 8, pp 30 – 37.
Lee, D.-D., and Seung H.-S. (2000) Algorithms for Non-negative Matrix Factorization. In Advances in Neural Information Processing System (NIPS), pp 556—562.
Lopes, M., Clement, B., Roy, D., and Oudeyer, P.-Y. (2015) Multi-Armed Bandits for Intelligent Tutoring Systems, Journal of Educational Data Mining (JEDM), Vol. 7, No. 2, pp 20–48.
Pelanek, R. (2016). Applications of the Elo Rating System in Adaptive Educational Systems. In Computers and Education , pages 169-179
Stamatatos, E. (2009), A survey of modern authorship attribution methods, Journal of the American Society for Information Science and Technology Vol. 60, No. 3, pp 538-556.
Tekin, C., Braun, J, and van der Schaar, M. (2015) eTutor: Online Learning for Personalized Education, In: Proceeding of International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 5545—5549.
Thai-Nghe, N., Drumond, L., Krohn-Gimberghe, A., and Schimidt-Thieme, L. (2010) Recommender System for Predicting Student Performance, In Proceedings of the Workshop on Recommender System of Technology Enhanced Elearning, pp 2811—2819.
Tomar, D. A. (2014), The Ghostwriting Business: Trade Standards, Practices, and Secrets. [online] The Best Schools. Available at: http://www.thebestschools.org/resources/ghostwriting-business-trade-standards-practices-secrets/ [Accessed 24th May 2017]

van Halteren, H. (2004), Linguistic profiling for author recognition and verification, in Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL), Article No. 199.

A.3. Tracking Behavioral Patterns among Students in an Online Educational System

Authors:	Stephan Sloth Lorenzen Niklas Hjuler Stephen Alstrup
Presented at:	International Conference on Educational Data Mining (EDM), Buffalo, NY
Journal:	Proceedings of the 11'th International Conference on Educational Data Mining
Year:	2018
Pages:	280-285
arXiv:	`https://arxiv.org/abs/1908.08937`

Tracking Behavioral Patterns among Students in an Online Educational System

Stephan Lorenzen
University of Copenhagen
lorenzen@di.ku.dk

Niklas Hjuler
University of Copenhagen
hjuler@di.ku.dk

Stephen Alstrup
University of Copenhagen
s.alstrup@di.ku.dk

ABSTRACT

Analysis of log data generated by online educational systems is an essential task to better the educational systems and increase our understanding of how students learn. In this study we investigate previously unseen data from Clio Online, the largest provider of digital learning content for primary schools in Denmark. We consider data for 14,810 students with 3 million sessions in the period 2015-2017. We analyze student activity in periods of one week. By using non-negative matrix factorization techniques, we obtain soft clusterings, revealing dependencies among time of day, subject, activity type, activity complexity (measured by Bloom's taxonomy), and performance. Furthermore, our method allows for tracking behavioral changes of individual students over time, as well as general behavioral changes in the educational system. Based on the results, we give suggestions for behavioral changes, in order to optimize the learning experience and improve performance.

Keywords

Student clustering, Non-negative matrix factorization, Educational Systems

1. INTRODUCTION + RELATED WORK

How students behave in educational systems is an important topic in educational data mining. Knowledge of this behavior in an educational system can help us understand how students learn, and help guide the development for optimal learning based on actual use. This behaviour can be understood both through an explicit study [5], or as in this paper through the automatically generated log data of the system.

The analysis of log data is usually done as an unsupervised clustering of students [2, 3, 4, 7]. A popular approach is to extract action sequences and transform them into an aggregated representation using Markov models [4, 7]. The Markov chains can then be clustered by different methods. Klingler et al. did student modeling with the use of explicit Markov chains and the clustering with different distance measures defined on the Markov chains [7]. Hansen et al. assumed the actions sequences to be generated by a mixture of Markov chains and used an heuristic algorithm to find the generating Markov chains [4]. Gelman et al. used non-negative matrix factorization to find clusters for different measures of activity aggregated in weekly periods during a MOOC course. These clusters are then matched from week to week by cosine similarity.

Our work is similar to Gelman et al. [3] in that we also use *Non-negative Matrix Factorization* (NMF) to make a soft clustering at the student level in a given time period, however our clustering is only made once, and we are looking at primary school data over a vastly longer period of time, (2 years compared to 14 weeks).

Our soft clustering by non-negative matrix factorization is based on log data from Clio Online.[1] Clio Online is the largest provider of digital learning for all subjects in the Danish primary school (except mathematics), having 90% of all primary schools in Denmark as customers.

Using NMF, we assume that the set of features chosen can be represented by a set of fewer underlying behaviors. These underlying behaviours would each be represented by a cluster in the non-negative matrix factorization. Each student will then get a number for each cluster in each time period representing how much of that underlying behavior he has shown in the given time period. Non-negativity gives the behaviors an additive structure, which is more natural than showing a negative amount of a given behavior. We reason that the soft clustering will show both the behaviors of individual students, as well as how the behaviors change over time, both individually and on a system-wide level.

In this paper, we will consider two main questions: a) how does student activity in the system affect performance, and b) how does student activity distribute between different levels of Bloom's taxonomy in different subjects. Both questions are important in regards to optimizing learning; the first in relation to performance, the latter in relation to utilization of all taxonomy levels.

[1]This data is proprietary and not publicly available.

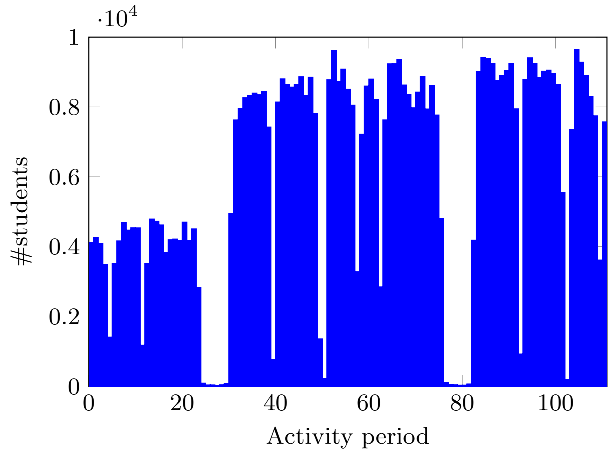


Figure 1: Number of students active in each period. Note that period 0 starts on 2015-01-08, while period 111 ends on 2017-03-01. The drops in activity occur due to vacation in Danish primary school, with the two large drops around periods 25 and 79 being due to the summer vacation.

2. EXPERIMENTAL SETUP

This section describes our experimental setup and methods. We start by describing our data and how it is preprocessed, and then move on to describing our clustering method.

2.1 Data Preprocessing

As mentioned, we consider log data generated in the Danish online educational system Clio Online. The system is used in Danish primary schools and contains learning objects across all Danish subjects (except mathematics), for instance texts, videos, sound clips and exercises. Furthermore, the system includes a large number of quizzes, used for evaluating students. Students may use the system for self study, but they may also be assigned homework by their teacher. Our data covers 14,810 students.

The raw data consists of logs detailing page accesses for individual students in the system. For quizzes, the final score (between 0 and 1) and total time spent for the quiz is also available. In our preprocessing, we combine these log entries to *sessions*. Two consecutive entries are considered in the same session, if they have the same subject, and their timestamps differ by less than some threshold. For our study, we choose this threshold to be 600 seconds, based on recommendations from Clio Online, who have a deeper knowledge of the content and flow of the system (e.g. expected time per page). Furthermore, quizzes are considered separate sessions. A total of 3 million sessions is obtained in this way.

With the sessions defined, we consider student activity in *activity periods*, with a length of one week. The data spans a total of 112 activity periods, starting January 2015 and ending in March 2017. For each activity period, we add an entry for a student, if the student is active (accesses the system) within that period. The entry for the given student contains all sessions for that student, which starts within the activity period. We end up with approximately 677,000 student entries across the 112 periods. Figure 1 shows the active number of students in each period. Note the drop in active students around periods 25 and 79; these drops in activity occur due to summer vacation.

The final step of data preprocessing is the feature extraction. For each activity period, a set of activity/performance related features are extracted. The features are chosen so as to answer the questions posed in the previous section. A complete overview of all features considered in our experiments is given in Table 1, including the maximum, mean and variance across all active students in all periods. Not all features are used for each experiment, see section 3.

All features are aggregates over the activity period. Below follows a detailed description:

- f_1 describes the activity during the period of day, where Danish students are normally in school, while f_2 describes the activity during non-school hours.
- f_3, f_4 and f_5 describe time spent doing exercises, reading texts and taking quizzes respectively.
- f_6, f_7 and f_8 describe time spent working with different topics: languages (Danish, English, German), societal (social studies, history, etc.) and science (physics, biology, etc.), respectively.
- f_9 is the average session length during the activity period.
- f_{10} is the average quiz score; this feature may be missing, if a student takes no quizzes during an activity period, but our analysis methods can handle this, see section 2.2.
- f_{11}, f_{12}, f_{13} and f_{14} describe the time spent doing exercises of different complexity, measured by their level in Bloom's taxonomy. We regroup the levels of Bloom's taxonomy into 4 levels:

  f_{11} **Remember/Understand**: Exercises involving reading and describing, e.g. "Read a map".

  f_{12} **Apply**: Exercises involving application of previously learned concepts, e.g. "Practice adjectives".

  f_{13} **Analyze/Evaluate**: Exercises involving discussion, analysis and experimenting, e.g. "Work with the poem", "Analyze the game".

  f_{14} **Create**: Exercises involving creation of a product, e.g. "Create a cartoon", "Write a story".

Having extracted m features for each student in each period, we construct the matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, where each of the n rows consists of the feature vector for an active student in a given activity period. Thus each student occurs several times in $\mathbf{X}$; once for each period, where they are active.

2.2 Soft Clustering using Non-negative Matrix Factorization

We will utilize non-negative matrix factorization for our soft clustering. The use of NMF as a soft clustering technique has become popular in recent times [10], with applications within several fields, such as clustering of images and documents [8, 13]. NMF has also seen success in the educational data mining community, for clustering tasks, as well as other tasks such as performance prediction [3, 12].

i	f_i	Max	Mean	Variance
1	Hours between 8AM and 4PM	31.85	0.940	0.862
2	Hours before 8AM and after 4PM	71.84	0.174	0.283
3	Hours doing exercises	3.61	0.048	0.019
4	Hours reading texts	7.73	0.344	0.148
5	Hours taking quizzes	23.76	0.231	0.297
6	Hours working with language subjects	58.28	0.531	0.693
7	Hours working with societal subjects	45.96	0.294	0.285
8	Hours working with science subjects	103.69	0.277	0.326
9	Average session length in hours	7.91	0.268	0.027
10	Average quiz score (in $[0,1]$)	1.00	0.733	0.034
11	Hours working with Bloom level 1	2.83	0.016	0.006
12	Hours working with Bloom level 2	1.64	0.008	0.002
13	Hours working with Bloom level 3	1.51	0.014	0.003
14	Hours working with Bloom level 4	2.04	0.009	0.003

Table 1: Overview of features.

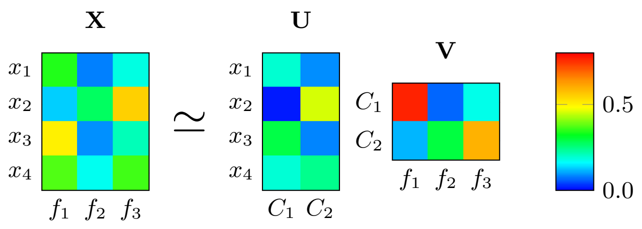


Figure 2: The soft clustering given by NMF.

NMF is a dimensionality reduction method, in which we are given a non-negative matrix $\mathbf{X} \in \mathbb{R}_+^{n\times m}$ and $k \in \mathbb{N}$, and wish to determine $\mathbf{U} \in \mathbb{R}_+^{n\times k}, \mathbf{V} \in \mathbb{R}_+^{k\times m}$, such that $\mathbf{X} \simeq \mathbf{UV}$. More specifically, we search for $\mathbf{U}$ and $\mathbf{V}$, such that the error $||\mathbf{X} - \mathbf{UV}||_F$ is minimized, where $||\cdot||_F$ is the Frobenious norm. For our analysis, we need to be able to handle missing values in $\mathbf{X}$. In this case the NMF problem is reformulated as the *weighted non-negative matrix factorization*, in which we are also given a binary weight matrix $\mathbf{W} \in \{0,1\}^{n\times m}$, where a 0 indicates missing data. Now, we wish to find $\mathbf{U}, \mathbf{V}$ such that $||\mathbf{W} \odot (\mathbf{X} - \mathbf{UV})||_F$ is minimized[2].

$\mathbf{U}$ and $\mathbf{V}$ admits a soft k-clustering as shown in Figure 2; $\mathbf{V}$ describes the importance of each feature for each cluster (for instance, f_1 has high importance in C_1), while $\mathbf{U}$ describes the membership of each data point to the different clusters (for instance, x_3 is mostly in C_1, while x_4 is in both clusters).

Note, that for NMF, we have $\mathbf{X} \simeq \mathbf{UV} = \mathbf{UIV} = \mathbf{UA}^{-1}\mathbf{AV}$, where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{A}$ is a $k \times k$ invertible matrix. This means that we may rescale $\mathbf{U}$ and $\mathbf{V}$ by this matrix, $\mathbf{A}$, and its inverse. In our analysis, we use this to rescale $\mathbf{V}$, such that all rows of $\mathbf{V}$ (the clusters) sum to one, thus making the clusters comparable, and membership of the different clusters easier interpretable.

There exist several algorithms for obtaining the non-negative matrix factorization of $\mathbf{X}$, for instance basic gradient descent, multiplicative update rules and alternating least squares; [1] gives a good overview in the non-weighted setting. Several of these algorithms have been adapted for the WNMF case, while approaches based on *expectation maximization* have also been proposed, see [6]. For our analysis, we will use the weighted version of the multiplicative update method, proposed by Lee and Seung [9].

The NMF algorithm given in [9], adopted to WNMF [6], is as follows:

1. Initialize $\mathbf{U}$ and $\mathbf{V}$.
2. Repeatedly update $\mathbf{U}$ and $\mathbf{V}$ by the following rules:

$$\mathbf{U} \leftarrow \mathbf{U} \odot \frac{(\mathbf{W} \odot \mathbf{X})\,\mathbf{V}^T}{(\mathbf{W} \odot (\mathbf{UV}))\,\mathbf{V}^T}$$

$$\mathbf{V} \leftarrow \mathbf{V} \odot \frac{\mathbf{U}^T\,(\mathbf{W} \odot \mathbf{X})}{\mathbf{U}^T\,(\mathbf{W} \odot (\mathbf{UV}))}$$

where division is done element-wise.

The literature explores several ways of initializing $\mathbf{U}$ and $\mathbf{V}$; in our case, we will simply use random initialization. The alternating optimization steps are applied until the decrease in error reaches below a set threshold. Finally, Lin has noted that the procedure described above may not converge to a stationary point, hence we modify the update rules as proposed by them [11]. Furthermore, since we in our case know all missing values of $\mathbf{X}$ to be bounded by a constant c, we modify the above procedure such that 0-weight values of $\mathbf{UV}$ that deviate above c are penalized, i.e. whenever a value $(\mathbf{UV})_{ij}$ with $\mathbf{W}_{ij} = 0$ gets larger than c, we set $\mathbf{X}_{ij} = c$ and $\mathbf{W}_{ij} = 1$, before the next update step. If $(\mathbf{UV})_{ij}$ decreases below c again, the weight is reset to 0.

It remains to be seen, how we select the number of clusters, k. For each experiment, we construct clusterings with $k = 1, 2, \ldots$, and stop when the decrease in error going from k clusters to $k+1$ clusters is below some threshold, which depends on the initial error. As a consequence clusters will be uncorrelated on a student level, since otherwise we would pick a lower k.

[2]$\odot$ denotes the Hadamard product (element-wise multiplication).

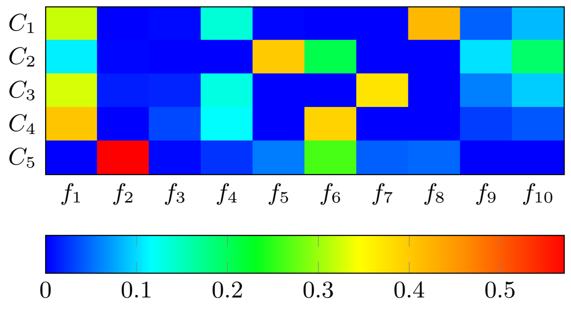


Figure 3: The cluster matrix for the first experiment.

3. EXPERIMENTS AND RESULTS

In this section, we present two different experiments using the setup described above. In the first experiment, we investigate the relation between activity, activity type, subject, time of day, average session length and performance. In the second experiment, we investigate the relation between complexities of exercises and subjects.

3.1 Performance and Optimal Behavior

In the first experiment, we investigate the relation between activity, activity type, subject, time of day, average session length and performance, i.e. we consider features $f_1, ..., f_{10}$. The features are extracted and $k = 5$ is selected, as described in section 2. We run the WNMF algorithm, and obtain the cluster matrix V as shown in Figure 3. From the figure, we can make several observations about the clusters:

C_1 In this cluster, we find students mostly working with the science subjects (f_8). These students seem to work mostly during school hours (f_1). The students also seem to spent a lot of time reading (f_4).

C_2 Students in this cluster spend a lot of time taking quizzes (f_5). They will spend some time during school hours (f_1) and some time working with language subjects (f_6). Furthermore, students in this cluster seem to both have fairly long average session length and high performance (f_9 and f_{10}).

C_3 In cluster C_3, we see students working with societal subjects (f_7). They work during school hours (f_1) and spend time reading texts in the system (f_4).

C_4 This cluster shows a relationship between being active in school (f_1) and spending time in the language subjects (f_6). Students in this cluster also spend time reading texts (f_4) and doing some exercises (f_3).

C_5 The most important feature for C_5 is f_2, i.e. the students in this cluster spend most time using the system during non-school hours. These students spent time in all subjects, but mostly languages (f_6), and they spent time taking quizzes (f_5).

From the clusters, we can see that the impact on performance from different behaviors depends on the subject. From cluster C_2, we see that students working mostly with language subjects gain most performance from spending time taking quizzes and working during school hours, whereas students working mostly with societal (cluster C_3) and science (cluster C_1) subjects gain most from reading texts,

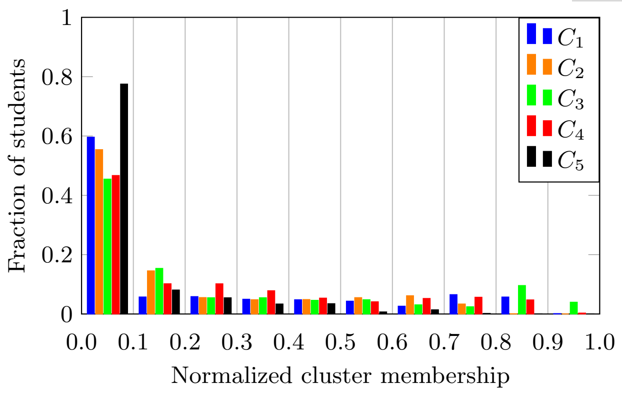


Figure 4: The distribution of cluster membership for the first experiment.

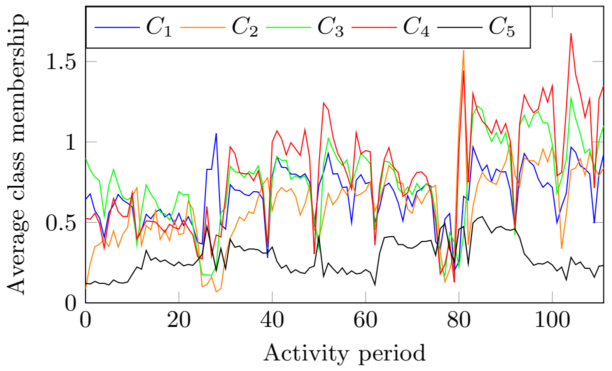


Figure 5: The average cluster membership in each activity period for the first experiment.

while working mostly during school hours. Note that cluster C_4 indicates that students working with languages may also improve performance by reading texts, but to a lesser degree than students working in other subjects. Finally, C_5 indicates that working mostly from home and primarily taking quizzes, does not improve performance. While C_5 indicates this for all subjects, the high importance of f_4 indicates that this most often occur for students working with languages, confirming the observations from C_2. Finally, it is also worth noticing, that there is a strong relation between performance and average session length (clusters C_1, C_2 and C_3), indicating that students, who perform well, also have longer sessions on average.

From the above discussion, it appears that the behavior in clusters C_4 and C_5 are sub-optimal, when considering performance, while students gain more from being in C_1, C_2 or C_3, i.e. by working during school hours, having longer sessions and taking quizzes (in the case of languages) or reading texts (in the case of societal or science subjects).

Figure 4 describes the distribution of cluster membership across all students and all activity periods , i.e. the columns of the first interval $[0, 0.1)$ gives for each cluster the fraction of students with 0%-10% membership. We see, that we do indeed get a soft clustering, with students often belonging to more than one cluster. Only C_3 seems to be the sin-

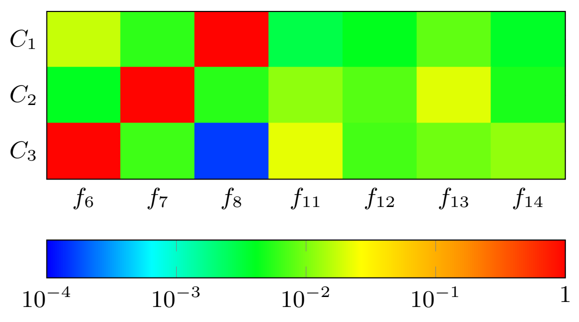


Figure 6: The cluster matrix for the second experiment. Note, that a logarithmic scale is used for this plot.

gle dominant cluster of some students. From the figure, we also see that students are typically never exclusively in C_5, which is positive, as the behavior observed in that cluster was not very productive in terms of performance. Other than that, we generally observe that students seem to distribute fairly well between the top four clusters, indicating most time spent during school hours and a varied use of both quizzes and texts across all subjects.

Next, we analyze how the membership of different clusters change over time. Figure 5 plots the average membership for each period, i.e. the average of rows from **U** belonging to the given period. The first observation we make from Figure 5, is that clusters C_1, C_2, C_3 and C_4 appear correlated at the system-wide level. This is due to these clusters being dependent on the general activity in the online system; most of the sudden drops occur at the same time as Danish school vacations, most notably the two larger drops around activity periods 25 and 79 (see Figure 1). C_5 seems to be relatively unaffected by the general activity, but this makes sense, as C_5 contains mostly students, who work outside school hours, and thus a lower membership is expected in that cluster in general, which is also the pattern we see in periods with no vacation.

Looking at the general distribution between the different clusters, C_3 and C_4 seem to be the most dominant, indicating that most students are working with language and societal subjects and reading texts. Cluster C_1 (science subjects) is fairly constant in the non-vacation periods, and C_2 seems to increase starting period 80, indicating that more students spend time taking quizzes. Finally, as mentioned, C_5 is the least active cluster across most periods. One general trend for the top four clusters seem to be an increase in activity during the 112 periods, indicating that students are spending more time in the system on average.

3.2 Subject and Exercise Complexity

In the second experiment we look at the relation between subjects and exercises grouped by Bloom's taxonomy level, i.e. we consider features $f_6, f_7, f_8, f_{11}, f_{12}, f_{13}, f_{14}$

We expect three clusters, one for each of the subject classes, which will tell us how much each Bloom level is used within each subject class. Figure 6 shows the cluster matrix found. From Figure 6, we make the following observations:

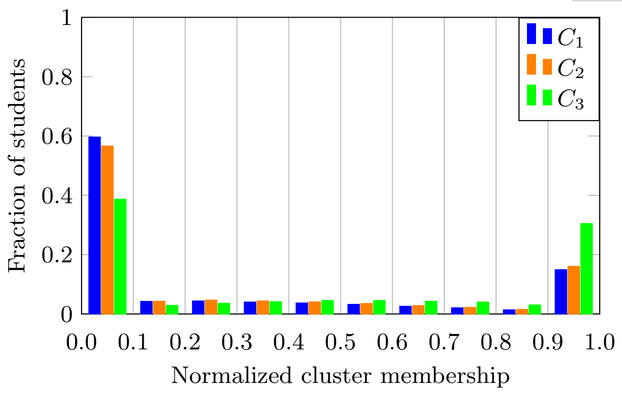


Figure 7: The distribution of cluster membership for the second experiment.

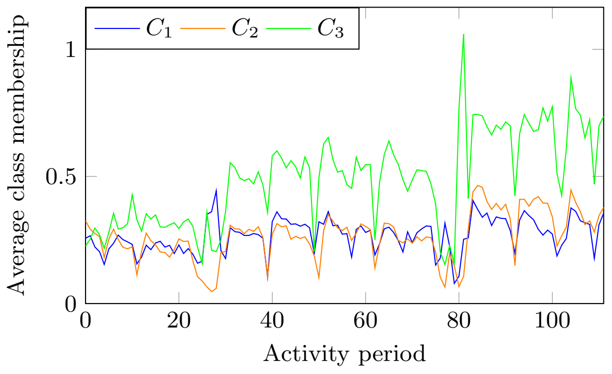


Figure 8: The average cluster membership in each activity period for the second experiment.

- C_1 In the science subjects, only very little of the 3 higher levels are used, and almost none of reading and understanding.
- C_2 For societal subjects, students have only little activity in the first 2 levels, a lot in analyzing and evaluating, and very little activity in creation.
- C_3 In languages, students have a tendency to read and understand a lot, and then distribute almost evenly on the 3 higher levels.

This implies that if we want to attract students to use an online educational system for languages, focus should be on exercises with Bloom's taxonomy level read and understand. For societal subjects the focus should be on exercises with analyzing and evaluating. For science we see no preference.

From Figure 7, we see that the clustering has many high values which is most likely explained by having a teacher who uses the system exclusively in only one of the subjects, which we can see happens most often for languages.

As we can see in Figure 8 all three clusters share similar curvature, which is partly explained by holidays. Especially the science and societal clusters behave seem highly correlated on a general level. We also see that in all three subjects, the average time spent during a week has gone from 15 minutes, to 45 minutes for languages and 25 minutes for both societal subjects and sciences. A clear indication that teachers and students in Denmark are using online educational systems more, especially for languages.

4. CONCLUSIONS AND FUTURE WORK

Several points can be taken from our analysis. We have identified three optimal and two sub-optimal behaviors in relation to subject and performance. One notably conclusion is that students using the Clio Online system during non-school hours (at home) do not seem to gain any significant boost to performance. We also saw how taking quizzes seems to increase the performance of students in languages, more so than in other subjects, where reading texts are of more importance. This fits the intuition that skills such as grammar need to be trained, in order to be learned. We inform how exercises are used depending both on their subject and their level in Bloom's taxonomy. And lastly we see that the average amount of time spent in the system is increasing both generally and for the individual students in all subjects, but especially for students working with languages. Furthermore, both experiments show how behaviors can have high correlation on a system-wide level, despite being uncorrelated on the individual student level. While the change of behavior for individual students was not directly analyzed in this paper (due to privacy concerns), our method allows for tracking such individual changes, hopefully helping teachers encourage optimal student behavior, e.g. by recommending training quizzes for students working with languages, or making sure that students are allowed more time to use the system in school.

In our setting, the number of clusters is fixed. It may be interesting to use an adaptive clustering strategy instead, as done in [7], as one might expect clusters to change over time. In the future, it might also be interesting to include other features, that were not available to us at this time, for instance whether a text (or quiz) have been assigned by a teacher, or whether the student reads it by themselves. For this study, we also only had access to a limited amount of data; better and more reliable results might be obtained by including more data.

5. ACKNOWLEDGMENTS

The work is supported by the Innovation Fund Denmark through the Danish Center for Big Data Analytics Driven Innovation (DABAI) project. The authors would like to thank Clio Online, and the reviewers for their thorough and insightful feedback.

6. REFERENCES

[1] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and Applications for Approximate Nonnegative Matrix Factorization. *Computational Statistics & Data Analysis*, 52(1):155 – 173, 2007.

[2] Louis Faucon, Lukasz Kidzinski, and Pierre Dillenbourg. Semi-Markov model for simulating MOOC students. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM)*, pages 358–363. International Educational Data Mining Society (IEDMS), 2016.

[3] Ben U. Gelman, Matt Revelle, Carlotta Domeniconi, Kalyan Veeramachaneni, and Aditya Johri. Acting the Same Differently: A Cross-Course Comparison of User Behavior in MOOCs. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM)*, pages 376–381. International Educational Data Mining Society (IEDMS), 2016.

[4] Christian Hansen, Casper Hansen, Niklas Hjuler, Stephen Alstrup, and Christina Lioma. Sequence modelling for analysing student interaction with educational systems. In *Proceedings of the 10th International Conference on Educational Data Mining (EDM)*, pages 232–237. International Educational Data Mining Society (IEDMS), 2017.

[5] Stephen Hutt, Caitlin Mills, Shelby White, Patrick J. Donnelly, and Sidney K. D'Mello. The Eyes Have It: Gaze-based Detection of Mind Wandering during Learning with an Intelligent Tutoring System. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM)*, pages 86–93. International Educational Data Mining Society (IEDMS), 2016.

[6] Yong-Deok Kim and Seungjin Choi. Weighted Nonnegative Matrix Factorization. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1541–1544, 2009.

[7] Severin Klingler, Tanja Käser, Barbara Solenthaler, and Markus Gross. Temporally Coherent Clustering of Student Data. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM)*, pages 102–109. International Educational Data Mining Society (IEDMS), 2016.

[8] Cosmin Lazar and Andrei Doncescu. Non Negative Matrix Factorization Clustering Capabilities; Application on Multivariate Image Segmentation. In *Proceedings of the 3rd International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 924–929, 2009.

[9] Daniel D. Lee and H. Sebastian Seung. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 556–562, 2000.

[10] Tao Li and Chris Ding. Non-negative matrix factorization for clustering: A survey. In *Data Clustering: Algorithms and Applications*, pages 149–176. Chapman & Hall/CRC, January 2013.

[11] Chih-Jen Lin. On the Convergence of Multiplicative Update Algorithms for Non-negative Matrix Factorization. *Trans. Neur. Netw.*, 18(6):1589–1596, 2007.

[12] Stephan Lorenzen, Ninh Pham, and Stephen Alstrup. On Predicting Student Performance Using Low-rank Matrix Factorization Techniques. In *Proceedings of the 16th European Conference on e-Learning (ECEL)*, pages 326–334. Academic Conferences and Publishing International, 2017.

[13] Farial Shahnaz, Michael W. Berry, Victor P. Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373 – 386, 2006.

A.4. Detecting Ghostwriters in High Schools

Authors:	Magnus Stavngaard August Sørensen Stephan Sloth Lorenzen Niklas Hjuler Stephen Alstrup
Presented at:	European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)
Journal:	Proceedings of the 27'th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning
Year:	2019
Pages:	197-202
arXiv:	`https://arxiv.org/abs/1906.01635`
Source code:	`https://github.com/StephanLorenzen/AuthorshipVerification`
Comment:	The initial study was conducted as a master thesis by Magnus Stavngaard and August Sørensen, supervised by the PhD-student. Their thesis formed the basis for the paper. Subsequent work by the PhD-student confirmed and slightly improved their findings.

Detecting Ghostwriters in High Schools

Magnus Stavngaard August Sørensen Stephan Lorenzen[†]
Niklas Hjuler Stephen Alstrup [*]

University of Copenhagen - Department of Computer Science
Universitetsparken 3, Copenhagen, Denmark
† Corresponding author, e-mail: lorenzen@di.ku.dk

Abstract. Students hiring *ghostwriters* to write their assignments is an increasing problem in educational institutions all over the world, with companies selling these services as a product. In this work, we develop automatic techniques with special focus on detecting such ghostwriting in high school assignments. This is done by training deep neural networks on an unprecedented large amount of data supplied by the Danish company MaCom, which covers 90% of Danish high schools. We achieve an accuracy of 0.875 and a AUC score of 0.947 on an evenly split data set.

1 Introduction

The number of Danish high school students using ghostwriters for their assignments has been rising at an alarming rate due to the emergence of several new online services, allowing students to hire others to write their assignments[1].

We consider in this paper the problem of detecting such ghostwriting, or as it is more commonly known: *authorship verification*. Authorship verification is a common task in natural language processing [2, 3, 4]: Given author α with known texts $t \in T_\alpha$ and unknown text x, determine whether α is the author of x. Often, a set of texts $\overline{T_\alpha} = T \setminus T_\alpha$ (T denoting the complete set of available texts) not written by α is also available, which can be utilized as examples of different writing styles, when training a model. Note however, that $\overline{T_\alpha}$ is unlikely to contain examples written by the true author of x, unlike in the related *authorship identification* problem, in which the task is to determine the exact author of x, given a set of candidate authors and their texts [5, 6].

In this paper, we focus on the problem in high schools. We have access to a large data set consisting of 130K Danish essays, written by more than 10K high school students[1]. Thus we have access to a lot of different authors, each with a large amount of text. We suggest a *generalizing* technique for authorship verification (as opposed to *author specific* models); using a Siamese network working at character level (an approach inspired by [5]), writing style representations are learned and compared, in order to compute the style similarity between two texts. Using the similarity measure provided by this network, x are compared to previous works $t \in T_\alpha$, and a final answer is given by a weighted combination of the individual similarities. The data used is supplied by MaCom, the company behind Lectio, the largest learning management system in Denmark.

[*]Supported by the Innovation Fund Denmark through the Danish Center for Big Data Analytics Driven Innovation (DABAI). The authors would like to thank MaCom.

[1]The data set is proprietary and not publicly available.

Many previous approaches for authorship verification/identification are based on excessive feature selection [7, 2], but neural network approaches have also been considered, for instance [3] who utilize recurrent neural networks for identification. Previous work on Danish high school essays have used author specific models for verification/identification [6], but this work is the first neural network based approach used on this data (and, to our knowledge, in this setting).

2 Method

As mentioned, we solve the authorship verification problem in two steps. First, we solve the problem of computing the writing style similarity between two texts by learning the similarity function $s : T \times T \to [0, 1]$ using a Siamese network (Section 2.1). Second, we solve the authorship verification problem for author α by combining similarities computed between the unknown text x and the known texts $t \in T_\alpha$. We consider several different ways to combine these similarities, based on their value and relevant meta data. (Section 2.2).

2.1 Network

Several different architectures are considered, using different input channels (e.g. char, word, POS-tags), and evaluated on a validation set. The architecture of our best performing network is shown in Figure 1.

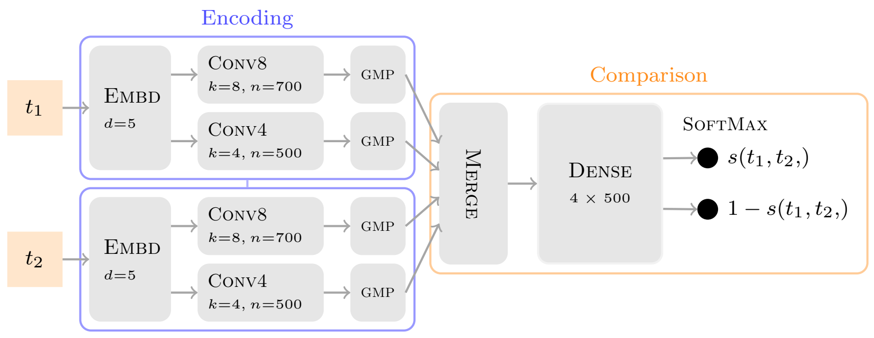


Fig. 1: Network architecture.

The Siamese network can be considered in two parts: *encoding* and *comparison*, the main idea being to learn an encoding of writing style, that the network is then able to distinguish. Our network uses only character level inputs.

The **encoding part** consists of a character embedding (EMBD), followed by two different convolutional layers: CONV8 using kernel size $k = 8$ and $n = 700$ filters, and CONV4 using $k = 4$ and $n = 500$. Each convolutional layer is followed by a global max pooling layer (GMP). The weights of EMBD and CONV8/CONV4 are shared between encoding t_1 and t_2.

In the **comparison part**, we first compute the absolute difference between the encodings in the MERGE layer. Afterwards, 4 dense layers with 500 neurons each are applied (Dense), and finally, the output is normalized by use of a softmax layer with two outputs.

2.2 Combining similarities

Having a good estimate of $s(t_1, t_2)$ for any two texts, we consider different ways to combine these similarities, in order to give the final answer to an authorship verification query. More specifically, we consider functions $C_s : \mathcal{P}(T) \times T \to [0, 1]$, such that, given x and T_α, we will answer the query positively (i.e. α is the author of x) if:

$$C_s\left(T_\alpha, x\right) \geq \delta$$

where δ is a configurable threshold, which describes how likely we are to answer positively. In the experiments, we consider several different ways to combine similarities, for instance using weighted sums, the min/max similarity or majority vote, while utilizing meta data such as time stamps and text length. From the experiments, we found that the optimal strategy was a weighted sum with weights decaying exponentially with time:

$$C_s\left(T_\alpha, x\right) = \sum_{t \in T_\alpha} e^{-\lambda \tau(t)} s(t, x) \tag{1}$$

where $\tau\left(t\right)$ denotes the time in months since t was written, and λ is a configurable parameter, which is determined experimentally.

3 Experiment

This section describes our experiments performed on the MaCom data. Section 3.1 will describe the preprocessing and partitioning of data. Baselines will be described in Section 3.2. Finally, Section 3.3 lists and discusses the final results. We use accuracy, *false accusation rate*, $\text{FAR} = \text{FN}/(\text{TN} + \text{FN})$, and *catch rate*, $\text{CR} = \text{TN}/(\text{TN} + \text{FP})$ as performance metrics.

3.1 Data

The data is partitioned into three sets: T_{train} used for training, T_{val} used for early stopping and selecting C_s, and T_{test} used only for estimating the metrics of the final models. The three sets are author disjoint, meaning no author will appear in more than one of the sets. In an effort to remove invalid data (blank hand-ins, etc.), we clean the data by filtering according to length (keeping texts with lengths between 400 and 30,000 characters). Furthermore, some texts were found to include author revealing information (such as name, address); hence we removed all proper pronouns from the texts, as well as the first 200 characters. Finally, authors with less than 5 texts were removed.

After cleaning, the data set contains a total of 131,095 Danish essays, written by 10095 authors, with an average 13.0 texts per author, and an average text length of 5894.8 characters.

For each data set, we construct two types of problem instances: Sim and AV, used for training the network and selecting the combination strategy respectively. The data set has no labelled ghostwriters, so we assume all authors to be correct[2], and construct balanced (50/50) data sets as follows:

A Sim instance simply consists of two texts t_1, t_2 and a label indicating whether the texts are by the same author. Positive samples are generated by using $t_1, t_2 \in T_\alpha$, while negative samples are generated by using $t_1 \in T_\alpha$ and $t_2 \in \overline{T_\alpha}$. An AV instance consists of a set of known texts T'_α, an unknown text x, and a label indicating whether α is (positive) or is not (negative) the author of x. Letting t_{last} denote the most recent text of T_α, samples are generated using $T'_\alpha = T_\alpha \setminus \{t_{last}\}$ with $x = t_{last}$ for the positive sample, and $x \in \overline{T_\alpha}$ chosen at random for the negative sample.

Table 1 provides an overview of the data after partitioning and preprocessing.

Data set	#authors	#texts	#Sim	#AV
T_{train}	5418	70432	934720	10836
T_{val}	989	12997	173536	1978
T_{test}	3688	47666	627744	7376

Table 1: Data set overview.

3.2 Baselines

We will compare our method to Burrows's Delta method and author specific SVMs:

Burrows's Delta method (Burrows) [7] is a method for authorship identification based on the l_1-distance between the z-scores of word frequencies in x and in the corpus for each of the candidate authors $\beta_1, ..., \beta_k$. We adapt it for verification by sampling a set of 'wrong' authors, $\beta_2, ...\beta_k$, and querying with x and $\beta_1 = \alpha, \beta_2, ..., \beta_k$. answering positively, if x is attributed to α. The top 150 word frequencies are considered. The optimal k is determined using T_{train}.

An author specific SVM [6, 2] is trained for each author in order to recognize T_α from $\overline{T_\alpha}$. Hyper parameters and features are selected using cross validation. Forward feature selection is used, considering char, word and POS-tag n-grams for varying n. The SVM will be trained on a balanced set, meaning that only a limited amount of data is available for each SVM. However, they have previously been shown to work well in this data set [6].

3.3 Results

Methods were trained and validated on T_{train} and T_{val}. For Burrows, we found $k = 4$ to give the best results, while the parameters $C = 10, \gamma = 10^3$ were found optimal for the RBF kernel SVM. The optimal combination strategy C_s was

[2]An undoubtedly false assumption, which will be discussed in Section 3.3

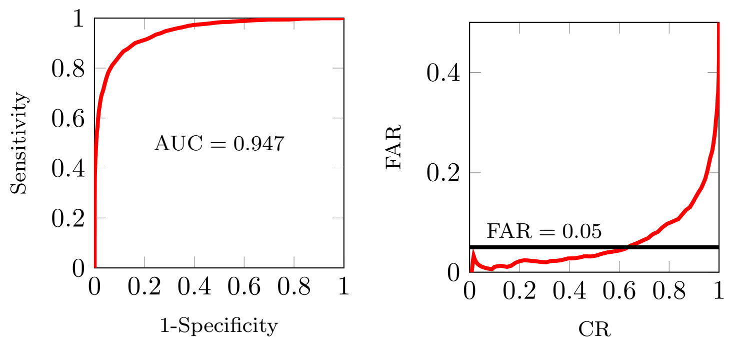


Fig. 2: ROC (left) and plot of false accusation rate/catch rate (right) on T_{test}.

found to be exponentially decaying weights (see (1)) with $\lambda = 0.1$. Furthermore, $\delta = 0.57$ was found to be optimal. Using these parameters, the baselines and our method were evaluated on T_{test}; Table 2 presents the results, while Figure 2 shows the ROC/AUC and a plot of false accusation/catch rate for our method. As it can be seen, our method clearly outperforms the baselines, on all metrics.

Method	Accuracy	FAR	CR
Burrows	0.677	0.357	0.806
SVM	0.720	0.266	0.689
Our method	0.875	0.141	0.896

Table 2: Results obtained on T_{test}

The false accusation rate is especially important considering the use case: when trying to detect ghostwriting in high schools, making false accusation can be especially devastating, as students found guilty of cheating could risk severe punishment and maybe even be expelled. Using this metric, our method performs very well, as illustrated in Figure 2 (right), a fairly low FAR can be obtained, while still catching a lot of ghostwriters. Optimizing the method on T_{val} while restricting FAR < 0.1, we achieved an accuracy of 0.864, FAR $= 0.106$ and CR $= 0.825$ on T_{test} (with exponential weighting and parameter $\lambda = 0.16$). However, even if these results are promising, the system should only be used as a warning system for the teacher, who should always have the final say.

An interesting aspect to note about the combination strategy C_s, is that it takes time into account with $\lambda = 0.1$, weighing recent assignments more than older ones. Since $\tau(t)$ measures in months, this means that a recent assignment gets $e^{12 \cdot 0.1} \approx 3.3$ times the weight of a one year old assignment. This corresponds well with the idea that high school students writing style changes over time, as also observed in [6].

When looking at the low false accusation rates of Figure 2 (right), one have to consider two things before translating them into practice: a) T_{test} is balanced, while in reality much less than half of assignments are written by a ghostwriter, and b) ghostwriting does happen, also in our data set, and thus most likely some of our labels are wrong. A possible remedy for the second point could be to adjust FN to $\mathrm{FN} - \frac{\mathrm{TN}}{\mathrm{TN+FP}}\gamma\mathrm{T}$ (where γ is the estimated fraction of ghostwriters and $\mathrm{T} = \mathrm{TP}+\mathrm{FN}$), and similar for TP, under the assumption that a negative sample and a corrupted positive sample are indistinguishable. Adjusting for this would obviously lead to improved accuracy and false accusation rate, but requires a good estimate of γ.

4 Conclusion

We achieved an accuracy of 0.875, with a false accusation rate of 0.141 and a catch rate of 0.896. We show how false accusation rate can be improved at the cost of catch rate and accuracy. Results are good enough for practical use, and even with a slightly lower catch rate, the system is still expected to have a preventive effect. However, one has to keep in mind that, in practice, the data set is not 50/50 balanced, which obviously will affect the results. Making a split imitating the real world is hard for two reasons: one needs a good approximation of the actual fraction of ghostwriters, and even if this fraction is known, the number of corrupt labels would be approximately the same as the number of negatives, making it impossible to beat a false accusation rate of 0.5, even for a perfect classifier. Finding a clean data set or establishing ground truth would alleviate these problems, and could be interesting prospects for future work.

Another interesting direction is to analyze writing style changes over time more in depth, motivated by the chosen combination strategy and preliminary experiments, which show how two texts written within a shorter time span have higher similarity on average.

References

[1] Politisk flertal vil gøre salg af eksamensopgaver ulovligt. `http://nyheder.tv2.dk/politik/2017-06-21-politisk-flertal-vil-gore-salg-af-eksamensopgaver-ulovligt`. Accessed: 2018-11-25.

[2] Efstathios Stamatatos. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556, March 2009.

[3] Douglas Bagnall. Author Identification using multi-headed Recurrent Neural Networks. In *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers*. CEUR-WS.org, September 2015.

[4] Alberto Bartoli, Alex Dagri, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. An author verification approach based on differential features. In *CEUR WORKSHOP PROCEEDINGS*, volume 1391. CEUR, 2015.

[5] Chen Qian, Tianchang He, and Rao Zhang. Deep learning based authorship identification. 2018. report, Stanford University.

[6] Niels Dalum Hansen, Christina Lioma, Birger Larsen, and Stephen Alstrup. Temporal context for authorship attribution: a study of Danish secondary schools. In *Multidisciplinary information retrieval*, pages 22–40. Springer, 2014.

[7] John Burrows. 'Delta': a Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing*, 17(3):267–287, 2002.

A.5. Investigating Writing Style Development in High School

Authors:	Stephan Sloth Lorenzen Niklas Hjuler Stephen Alstrup
Presented at:	International Conference on Educational Data Mining (EDM), Montreal, Canada
Journal:	Proceedings of the 12'th International Conference on Educational Data Mining
Year:	2019
Pages:	572-575
arXiv:	`https://arxiv.org/abs/1906.03072`
Source code:	`https://github.com/StephanLorenzen/AuthorshipVerification`
Comment:	The attached paper is the long version of the paper presented at EDM'19.

Investigating Writing Style Development in High School

Stephan Lorenzen
University of Copenhagen
lorenzen@di.ku.dk

Niklas Hjuler
University of Copenhagen
Hjuler@di.ku.dk

Stephen Alstrup
University of Copenhagen
alstrup@di.ku.dk

ABSTRACT

In this paper we do the first large scale analysis of writing style development among Danish high school students. More than 10K students with more than 100K essays are analyzed. Writing style itself is often studied in the natural language processing community, but usually with the goal of verifying authorship, assessing quality or popularity, or other kinds of predictions.

In this work, we analyze writing style changes over time, with the goal of detecting global development trends among students, and identifying at-risk students. We train a Siamese neural network to compute the similarity between two texts. Using this similarity measure, a student's newer essays are compared to their first essays, and a writing style development profile is constructed for the student. We cluster these student profiles and analyze the resulting clusters in order to detect general development patterns. We evaluate clusters with respect to writing style quality indicators, and identify optimal clusters, showing significant improvement in writing style, while also observing suboptimal clusters, exhibiting periods of limited development and even setbacks.

Furthermore, we identify general development trends between high school students, showing that as students progress through high school, their writing style deviates, leaving students less similar when they finish high school, than when they start.

Keywords

Student clustering, Writing style analysis, Siamese Neural Network, Educational Systems

1. INTRODUCTION

One of the most essential skills, learned during the course of primary, secondary and high school, is writing. While the main focus of primary school are on basic writing skills (such as grammar), secondary or high school will be more focused on improving *the linguistic writing style* of a student, that is, the quality of the written text as perceived by the reader. With many jobs being highly dependent on producing relatively large amounts of well-written text, no justification is needed for why *good writing* is an essential skill.

The definition of quality in linguistic writing style is widely discussed [3, 23]. While correct grammar being a prerequisite, several other measures are also correlated to writing style being perceived as good, for instance use of vocabulary, sentence structure and readability [18]. Our focus in this work will mainly be writing style *development* through the course of high school, while writing style quality will have a secondary role. We consider data from Danish high schools, consisting of Danish essays, and investigate the general development patterns among the students during the three years of study. The end goal is to be able to provide feedback to teachers about the development of their students' writing styles. We identify patterns among thousands of students across different classes and institutions, allowing us to provide teachers with new insights, which the data available to the teacher might not show. For instance insights about students, whose writing style development patterns may be unique within their own classes.

By itself, our method potentially allows for identifying students with deviating writing styles development (which might be good or bad), or students with sudden significant changes in writing style, which could be an indicator of cheating. However, we also consider several measures for the *quality* of writing. We investigate how these measures correlate with the different patterns of writing style development found, as a mean to detect optimal and suboptimal development profiles with respect to text quality. Information of this kind could be used to help teachers tailor their teaching style to specific groups of students, who may need training in specific areas challenging to their development profile.

1.1 Our Contribution

As mentioned, we concern ourselves with the development of linguistic writing style (as opposed to e.g. handwriting) during the course of high school. Specifically, we investigate the development of writing style in Danish essays handed-in by students in Danish high schools [1].

[1]Note, that high school in Denmark usually consists of three years of study with students normally starting at age 15-17 and finishing at age 18-20.

We are interested in determining general patterns of development, and to discuss which of the patterns are optimal, in the sense of improving writing style quality. In particular, we consider the following questions:

- How does the writing style of a student develop, and what are the typical kinds of development in writing style?
- How does writing style changes correlate with measures of quality?
- How does writing style similarity between students behave, with respect to how far the students are in their education?

Our study is based on data from the company MaCom[2], who is behind the learning management system Lectio, a system used by 90% of Danish high schools. Students submit their written essays through Lectio, giving MaCom access to a huge corpus of Danish texts by high school students, marked with author and date of submission.

Our approach is based on methods from authorship verification; in order to learn a similarity measure for writing style, we consider examples of writing styles in texts from the same or different authors, similar to how it is done in verification tasks. We use a Siamese neural network for learning this similarity measure. While training, time is not taken into account. Assuming that writing style actually changes over time, this will lead to a suboptimal network. However, testing the network, we see clear patterns in how the "errors" distribute for a single author, indicating that the network simulates the best similarity measure possible, and the "errors" are actual changes in writing style. Using this method, writing style development profiles are generated and clustered for a large set of students. Analyzing the clusters, we see optimal and suboptimal types of development. In general, the average similarity is found to decay with time to a great extend, which corresponds well with the general perception, that writing style changes during high school, and also matches conclusions made in the literature [4, 9, 25].

While this paper presents a case study of the data from MaCom, the methods used for analysis are of independent interest, and not specific to the Danish language or high school, except for the neural network, which would at least require retraining in the given language. Considering other network architectures than the one used in this work, might also improve upon the analysis, see for instance [19] for a network used with English.

1.2 Related Work

Writing style analysis, in one way or another, has been studied in the natural language community for many years. Typically, the analysis of writing style is used as a middle link for tasks such as *authorship verification* [19,24,25], in which a text of unknown authorship is given, together with a set of texts by some known author, and we wish to verify, whether the given author is the author of the unknown text. Similarly, in *authorship attribution* the unknown text must be attributed to one of several known authors. Traditional methods for verification and attribution utilize both unsupervised methods from the field of outlier detection [24], as well as standard supervised learning techniques, such as SVMs [9] and techniques based on neural networks [19, 25].

Other uses of writing style analysis include distinguishing features of the writer (e.g. sex and age [1, 17, 21, 22], demographics [2], or nationality [12]), using supervised learning algorithms such as SVMs, random forest, and neural networks. Other studies have investigated written conversations on online forums, trying to infer whether one person is trying to convince another [8].

Some studies investigate the quality of writing, for instance prediction of popularity of news articles [27], or the quality of scientific articles [14]. The former uses the popularity of an article on social media as a measure of quality, while the quality measure of scientific papers considered in the latter is based on acceptance of a paper to "The Best American Science Writing", an anthology of popular science articles published in the United States on a yearly basis.

Few studies consider development of writing style as the main objective. [5] uses neural network models to track style of *handwriting* (i.e. not linguistic writing style) and investigate the development of handwriting among young students, and how similar it is when compared to different students, in the same/different grade level. [3] shows how students in higher grades get higher scores for their essays from teachers, in a blind experiment, where all student information is hidden from the grading teacher. [4] considers two famous Turkish writers, investigating their change in writing style over time, the most significant finding being average word length increasing with the age of the author.

Finally, several studies related to writing style have been conducted using the data available from MaCom. [9] investigates temporal aspects of authorship attribution, and concludes that considering more recent essays improves authorship attribution algorithms, indicating that the writing style among high school students does indeed change with time. [25] also uses the MaCom data for testing their neural network based authorship verification methods; their results also support these findings.

2. METHODS AND SETUP

This section describes our experimental setup and methods. We start by giving some basic notation.

We consider a set of students $\mathcal{A}$, and let $\alpha \in \mathcal{A}$ denote a single student with texts $t \in T_\alpha$. Furthermore, let $T = \cup_{\alpha \in \mathcal{A}} T_\alpha$ denote the entire corpus of texts.

Since our main focus is how the writing style of a student develops during the time they spend in high school, we are interested in computing a similarity function $s : T \times T \rightarrow [0, 1]$, allowing us to compare the writing style between two texts. As mentioned, we utilize a Siamese neural network to compute s; this approach is widely used for computing writing style similarity [7, 19, 25], specifically, our network

[2]The data set is proprietary and not publicly available

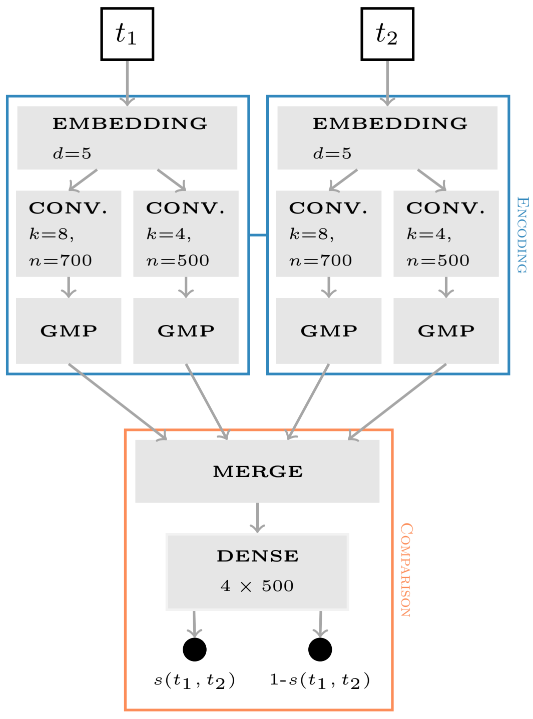


Figure 1: Network architecture.

will be similar to that of [25]. Section 2.1 will describe our network in detail.

The similarity measure s found will then be utilized for writing style analysis. Primarily, we will focus on determining development patterns by generating a *writing style development profile* P_α for each student α. These profiles are then clustered and analyzed with respect to different measures for text quality. The profile generation and clustering are described in more detail in Section 2.2.

Finally, we also explore how the similarity between random students change depending on their current progress through high school. This is done by sampling random pairs of texts $t_1 \in T_\alpha, t_2 \in T_\beta$ and computing their similarity. We then consider how the similarity changes depending on if α and β are in the same grade or not.

2.1 Text Similarity using a Siamese Neural Network

As mentioned, we use a Siamese neural network for computing the similarity $s(t_1, t_2)$ between two texts t_1 and t_2. We considered several different architectures, using different input channels (e.g. char, word, part of speech tags). These architectures were evaluated using a validation set (see Section 3.1), and the best architecture was selected, as shown in Figure 1. The network relies only on character level inputs.

The basic philosophy behind the network is to a) *encode* the two texts in some space using a replicated encoder network with shared weights, and b) *compare* the two texts in this space.

- The encoder network in the **encoding** part of our network consists of a character embedding (using ReLu activation functions), followed by two different convolutional layers (**CONV**): one using kernel size $k = 8$ and $n = 700$ filters, and one using $k = 4$ and $n = 500$, each followed by global max pooling layers (**GMP**).
- In the **comparison** part of the network, the **MERGE** layer first computes the absolute difference between the outputs of the two encoder networks. Afterwards, four dense layers (**DENSE**) with 500 neurons each are applied, using ReLu for activation function and with a dropout of 0.3. Finally a two neuron softmax layer is used to normalize the output.

Using the convolutional layers, the network extracts character n-grams. Specifically, it compares 8- and 4-grams. Character n-grams have been shown to be an important feature in writing style analysis tasks such as authorship attribution [24]. We did also consider architectures using recurrent networks, however none of them performed as well as convolutional networks.

2.2 Student Profiling and Clustering

As mentioned, we construct writing style development profiles for the students, in order to analyze the general development patterns. The profile P_α for student α is constructed by first determining their initial writing style. The natural way to do so, and indeed our approach, is to consider their early work. One or more texts may be used to represent the initial writing style, as a trade off between the amount of data available for the profile and the robustness of the initial writing style estimation. P_α then consists of a chronologically ordered sequence of similarities, between any $t \in T_\alpha$ and this initial writing style. More specifically, if $t_1, t_2, ..., t_{|T_\alpha|}, t_i \in T_\alpha$ is a chronologically ordering of T_α, we compute the similarity p_i between t_i and the initial writing style by:

$$p_i = \frac{1}{m}\sum_{j=1}^{m} s(t_i, t_j),$$

where m is the number of texts used for representing the initial writing style. Since the first m texts are part of the initial writing style, $p_1, p_2, ..., p_m$ are not independent, and thus we exclude the first $m-1$ texts, and re-index such that $p_j = p_{i-m+1}$. Furthermore, for each text, we let τ_j denote the time in months since t_m was written, i.e. the time since p_0, with $\tau_0 = 0$. Now, the final profile becomes the sequence consisting of pairs (τ_j, p_j) of length $|T_\alpha| - m + 1$. Note that the profile now describes a curve.

These profiles are now clustered using a slightly modified k-means clustering. Before clustering, for each profile P_α, an approximate profile $\hat{P}_\alpha$ is constructed by interpolating values between any two consecutive pairs (τ_j, p_j) and (τ_{j+1}, p_{j+1}), in intervals of 0.05 months. Thus $\hat{P}_\alpha$ becomes a vector $\hat{P}_\alpha \in [0, 1]^{\ell_\alpha}$ consisting of similarities for every 0.05 month, with length ℓ_α.

These approximate profiles are then clustered. The clustering is complicated by profiles having variable length: $\hat{P}_\alpha$ has length ℓ_α depending on $\tau_{|T_\alpha|-m+1}$ (the time span between

t_m and t_{T_α}), specific to α. Hence, distance computation used in the clustering algorithm is modified slightly; we compute the distance $dist(\hat{P}_\alpha, \hat{P}_\beta)$ between two profiles $\hat{P}_\alpha$ and $\hat{P}_\beta$ by computing the Euclidean distance between the prefixes of length $\ell = \min\{\ell_\alpha, \ell_\beta\}$ of the two profiles:

$$dist(\hat{P}_\alpha, \hat{P}_\beta) = dist_E(\hat{P}_\alpha[1...\ell], \hat{P}_\beta[1...\ell]),$$

where $dist_E$ denotes the Euclidean distance, and $v[1...n]$ denotes the prefix of length n of vector v.

Similarly, when computing centroid C_r for cluster $\mathcal{C}_r$, profile $\hat{P}_\alpha$ contributes only to the ℓ_α first entries of C_r. Thus, with $\mathcal{C}_r^j = \left\{\hat{P}_\alpha | \hat{P}_\alpha \in \mathcal{C}_r, \ell_\alpha \leq j\right\}$, the j'th entry of C_r is then computed as:

$$C_r[j] = \frac{1}{|\mathcal{C}_r^j|} \sum_{\hat{P}_\alpha \in \mathcal{C}_r^j} \hat{P}_\alpha[j]$$

where $v[j]$ denotes the j'th entry of vector v.

The clustering is initiated by selecting k profiles at random as the initial clusters, and then continually reassigning profiles and recomputing centroids for clusters. Having reassigned the profiles, the $E_\mathcal{C}$ is computed:

$$E_\mathcal{C} = \frac{1}{|\mathcal{A}|} \sum_{r=1}^{k} \sum_{\alpha \in \mathcal{C}_r} dist(\hat{P}_\alpha, C_r)$$

The algorithm iterates until the change in cluster error $E_\mathcal{C}$ is sufficiently small ($E_\mathcal{C} \leq 10^{-6}$), or until a set number of maximum iterations (100) is reached.

Selecting the number of clusters k is an inherent problem in all unsupervised learning task. One approach is to base the decision on domain knowledge, and select the "right" number of clusters. We will instead make use of the so called *elbow heuristic* which relies on looking at how the error decreases with the number of cluster and pick at the "elbow" in the resulting curve [26].

Having determined the parameter k and found k clusters, we compute a few statistics and writing quality indicators for each cluster. Specifically, we will compute the average *noun and verb phrases*, defined as the ratio between nouns and sentences, and the ratio between main verbs and sentences respectively. These measures, especially verb phrases, have been shown to correlate well with readability, which correlates with text quality [18]. Furthermore, we compute the *simple measure of Gobbledygook* (SMOG) grade [16], a measure estimating the grade level required for understanding the text. The SMOG grade is computed as:

$$\text{SMOG} = 1.0430\sqrt{\frac{30 n_{w*}}{n_s}} + 3.1291,$$

where n_{w*} is the number of words of 3 or more syllables, and n_s is the number of sentences [16].

Note that the study showing correlation between noun and verb phrases, and readability, is done on English texts. The SMOG grade as well is defined with the purpose of evaluating English texts. Hence, one must be careful when basing conclusion on these measures when used on Danish. However, we believe they can still provide information about the development, even if the exact computed value might be hard to interpret.

3. EXPERIMENTS AND RESULTS

In this section, we present the data, the experimental setup, and the results obtained. Section 3.1 presents the data, how it is preprocessed and split for training and analysis and some basic statistics. Section 3.2 describes the training of the Siamese neural network, while Section 3.3 describes the clustering and shows the resulting clusters.

3.1 Data

The full data set made available to us by MaCom contains around 130K essays by approximately 10K students, with an average length of about 6K characters. The data set was cleaned by removing very short (≤ 400) and very long ($\geq 30,000$) texts, in order to get rid of outliers/invalid essays (blank hand-ins, garbled texts, etc.). Furthermore, proper pronouns were substituted with placeholder tokens and the first 200 characters of each text were removed, in an effort to remove any data identifying the real author of the text, as such clues could be picked up by the neural network and lead to overfitting. Finally, authors with less than 5 texts were removed. Following this cleaning, the data set contains a total of 131,095 Danish essays, written by 10095 authors, with an average 13.0 texts per author, and an average text length of 5894.8 characters.

We partition the clean data into two author disjoint sets: $T_{network}$ used for training the neural network, and $T_{analyze}$, which we analyze using the trained similarity function. $T_{network}$ is further split into a training set T_{train} and a validation set T_{val} (also author disjoint), used for early stopping when training the network. As the analysis relies heavily on a strong similarity function, the majority of the data (around two thirds) is used for $T_{network}$. The exact sizes are given in Table 1.

Data set	#students	#texts	#SIM
T_{train}	5418	70432	934720
T_{val}	989	12997	173536
$T_{analyze}$	3688	47666	N/A
Total	10095	131,095	1108256

Table 1: Data set overview. The table lists the number of students and texts, as well as the number of problem instances #Sim for training the Siamese neural network.

Data for network training

For training and evaluating the Siamese neural network, we require problem instances consisting of a pair of texts, and a label indicating whether they are by the same author (positive sample) or by different authors (negative sample). We refer to these instances as SIM-instances, and generate them for the training set T_{train} and the validation set T_{val}.

Positive SIM-instances are generated by using $t_i, t_j \in T_\alpha$ with $i \neq j$, while negative instances are generated by using $t_i \in T_{\beta_1}$ and $t_j \in T_{\beta_2}$, where i, j, β_1, β_2 are selected at random, with $\beta_1 \neq \beta_2$. A balanced 50:50 data set is generated by generating the maximum number of positive instances

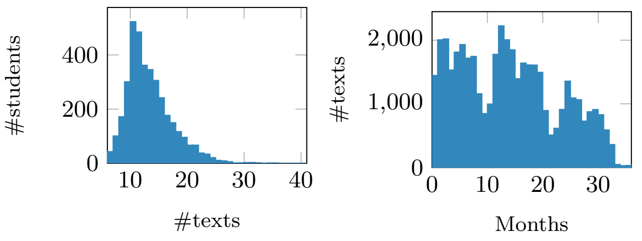


Figure 2: Statistics for $T_{analyze}$. Distribution of students according to number of essays written (left) and total number of essays written at any time during a students stay in high school (right).

for each student, and an equal number of negative instances. The final numbers of SIM-instances for T_{train} and T_{val} are shown in Table 1.

Note, that in generating these samples, we assume all claimed authors in the data to be the real authors; in reality, several students may use ghostwriters or plagiarism, in which case the labels will be wrong. However, we expect that the number of invalid labels is low.

Data for clustering and analysis

The clustering is performed on the remaining data in $T_{analyze}$. Each data point consists of a single student and their texts. As mentioned, an author has around 13 texts in average, distributed over three years; the actual distribution is shown in Figure 2 (left)[3].

Figure 2 (right) shows the number of essays handed in during the three years of high school. The summer vacations are clearly visible in the plot. Note also, that the number of hand-ins drops during the third year. A few students spend more than three years (not shown in the figure), but as only a few students hand-in after 30 months, we consider only the data within 30 months in the experiments[4].

3.2 Neural Network Training

The similarity network described in Section 2.1 was implemented using TensorFlow. We generate SIM-instances for T_{train} and T_{val}, and optimize the network for cross entropy using the Adam optimizer. The final network obtains a training loss of 0.5026 and a validation loss of 0.5357. Rounding the computed similarity to 0 or 1, we can compute an accuracy of 0.7451 for the training set and an accuracy of 0.7178 for the validation set. Figure 3 shows a plot of the loss and accuracy, as the network was trained.

3.3 Clustering

Using the similarity network to compute the similarity function s, we construct profiles as described in Section 2.2. We found that using $m = 2$ texts for determining the initial

[3]Recall that, students with less than 5 essays is not considered in this study.
[4]The time span considered is smaller than three years (36 months), since we measure the time from first hand-in until the last. Combining this with vacation and finals, most students appear to only be active within the 30 month period.

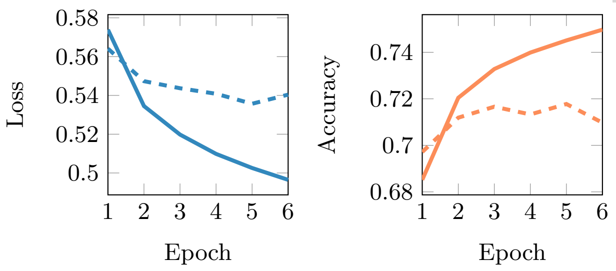


Figure 3: Plot of training (solid) and validation (dashed) loss (left) and accuracy (right), the latter computed by rounding the output. Minimum validation loss was obtained at epoch 5.

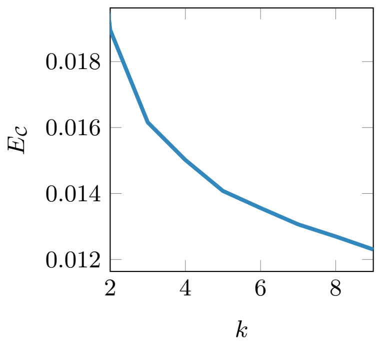


Figure 4: The cluster error $E_{\mathcal{C}}$ obtained for various values of k.

writing style yielded good results. Thus, profile P_α consists of $|T_\alpha| - 1$ pairs (τ_j, p_j) with:

$$p_j = \frac{s(t_{j+1}, t_1) + s(t_{j+1}, t_2)}{2}.$$

and τ_j being the time in months since hand-in of t_2. With a single profile constructed per student, the total number of profiles is equal to the number of students, as given in Table 1. As mentioned, the lengths of the profiles depend on the number of texts written during the time, they spend in high school. Thus the distribution of the lengths of profiles follows that presented in Figure 2 (left).

We now apply the elbow method in order to determine the optimal number of clusters k. We compute a clustering for $k = 2, 3, ..., 9$, and plot the resulting cluster error $E_{\mathcal{C}}$ in Figure 4.

Based on Figure 4, we select $k = 5$, as the curve flattens considerable for $k = 6$. The final clustering is performed, obtaining five clusters: $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$, and $\mathcal{C}_5$, with a cluster error of $E_{\mathcal{C}} = 0.01407$. The curves representing the final clusters are shown in Figure 5, while Table 2 lists the number of members in each cluster. Furthermore, we sampled two million random pairs of texts with random (different) authors, and computed the similarity for these samples, obtaining an average of 0.3470. This average is also plotted in Figure 5, while the similarity with respect to time is plotted as a heat map in Figure 7.

Finally, Figure 6 shows a more detailed view for each cluster.

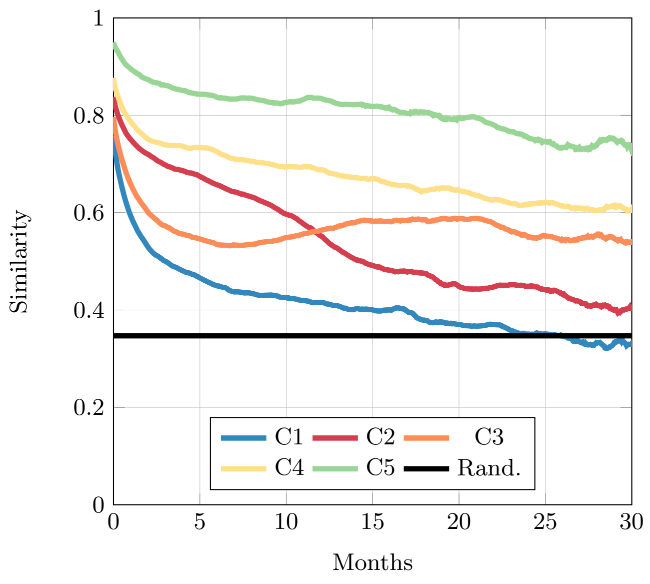


Figure 5: The curves representing the five clusters found. The average similarity between random texts by different students is also plotted.

Cluster	#students
$\mathcal{C}_1$	603
$\mathcal{C}_2$	720
$\mathcal{C}_3$	884
$\mathcal{C}_4$	969
$\mathcal{C}_5$	512

Table 2: The number of students in each cluster.

The similarity curve plots include a plot of the middle 90% of profiles in each cluster. The SMOG score, the noun and verb phrases, and the average text length (in words) are also plotted, as indicators for writing quality changes for the given cluster, see also Section 2.2.

Note, that in visualizing and inspecting the clusters, we consider only data until 30 months, since, as mentioned, only few students are active after 30 months, and the number of data points contributing to that part of the cluster curve becomes small.

4. ANALYSIS AND DISCUSSION

This section presents our analysis and discussion of the five clusters found. Section 4.1 describes and discusses the characteristics of each cluster. Section 4.2 discussed how similarity between random students behaves with time.

4.1 Cluster Analysis

When analyzing the clusters, three properties are important in order to understand a cluster: the initial value of the curve, the shape of the curve, and the total change from start to end. The *initial value of the curve* describes the similarity between the second text of a student and their initial writing style, which is based on the first two texts of the student. Thus a smaller initial value indicates a high initial variance in writing style, which could be an indication of a developing writing style. The *shape of the curve* describes the rate of change in writing style. And finally the *total change* tells us how much the writing style has evolved.

While the similarity curves themselves give no information about *the quality* of the writing, we will use the indicators of writing style, described in Section 2, in the discussion: the SMOG grade, and the noun and verb phrases per sentences. For each of these indicators, the average curves for each cluster are plotted in Figure 6.

Before discussing each cluster in detail, we note some patterns common for all clusters. Across all clusters, it seems the number of words written increases (with the exception of $\mathcal{C}_5$), and the increase seems to be correlated with the corresponding decrease in similarity. Furthermore, on average, students in all clusters appear to be improving with respect to the quality metrics. While positive, some clusters see a smaller increase than others, indicating that these clusters represents suboptimal development profiles. Finally, we note for the SMOG grade, that the maximum increase, occurring in $\mathcal{C}_1$, is only slightly above 1, which might not seem impressive across three years. However, as discussed in Section 2.2, the SMOG grade is a measure designed for readability of English texts, and thus may not be entirely accurate for Danish texts.

Below follow detailed descriptions of each cluster:

- $\mathcal{C}_1$ The initial similarity of $\mathcal{C}_1$ is the lowest among the clusters found. Furthermore, the similarity drops rapidly during the first year, and continues the decline, leading to $\mathcal{C}_1$ having the lowest final similarity with the initial writing style, among all the clusters. In fact, the similarity between the first and the last assignments for students in this cluster is so low, that they could just as well have been written by different students, as can be seen when comparing to the average similarity between random students plotted in Figure 5. Thus, $\mathcal{C}_1$ contains students with a significant change in writing style, happening mostly during the first year of high school. Considering the other metrics plotted in Figure 6, we first note the increase in number of words written, as it is particular extreme in the case of $\mathcal{C}_1$, increasing by almost a factor 2 from start till end. This increase also helps explain the decrease in similarity in two ways: a) length is itself a part of writing style recognized by the network, and b) it seems that writing style changes is correlated with when you start writing more. Looking at the SMOG grade, we see an overall large increase, indicating that the students of $\mathcal{C}_1$ does indeed improve, especially compared to the other clusters. Nouns and verbs per sentence are also both increasing, which also indicates that the students in this cluster write longer sentences.

- $\mathcal{C}_2$ The initial value of cluster $\mathcal{C}_2$ of about 0.84 (third highest among the clusters) indicates an initial low variance in writing style, but the following drop to about 0.4 is quite significant, indicating that the writing style of students in $\mathcal{C}_2$ change a lot during high school, similar to $\mathcal{C}_1$. However, where $\mathcal{C}_1$ had a sudden drop in

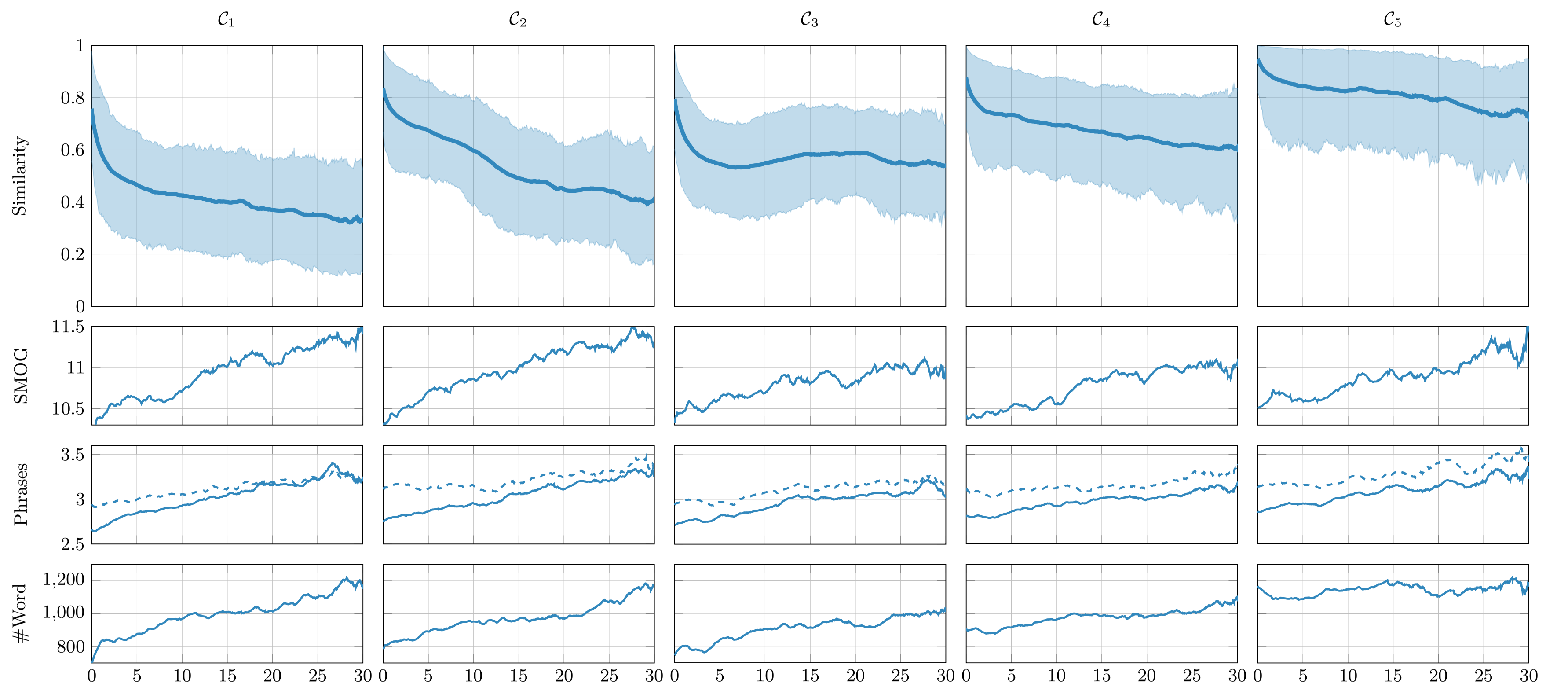


Figure 6: Detailed plots of the five clusters found. The top plot shows the similarity curve, with the middle 90% of profiles in each cluster plotted as well. The second plot from the top shows the development of the SMOG grade, while the third plot shows noun (solid) and verb (dashed) phrases. Finally the bottom plot shows the development of number of words written.

similarity, the change in $\mathcal{C}_2$ is more constant.

Considering the other metrics, we see the number of words written is increasing from about 800 to almost 1200, while the SMOG grade is again showing a large increase from about 10.3 to 11.3. Noun and verb phrases see modest increases. All in all, the metrics indicate a good development of writing style among students in $\mathcal{C}_2$, similar to $\mathcal{C}_1$. However, the more gradual change in similarity of $\mathcal{C}_2$ is preferable to that of $\mathcal{C}_1$, as the development does not stagnate already after the first year.

$\mathcal{C}_3$ After a significant initial drop in similarity in $\mathcal{C}_3$, the similarity actually increases again after the first year, showing the students in this cluster actually reverts to writing style more similar to their original work, before dropping a bit again in the last months. This corresponds well with a smaller improvement in e.g. SMOG grade (around 0.5) compared to the other clusters.

The setback seems to start around the first summer vacation. While not necessarily bad (as students could be reverting back from a worsened writing style), the increase in similarity could indicate reverting to a worse writing style. As such, students in $\mathcal{C}_3$ may be at risk. Many remedies for helping these students could be imagined, from simply encouraging the student to write during their vacation, to going to summer school.

$\mathcal{C}_4$ The similarity of $\mathcal{C}_4$ drops slightly at first, but then decreases slowly at a constant pace, until it reaches a similarity of about 0.6. The total change is smaller than several of the other clusters, as is the improvement in both SMOG grade and noun/verb phrases, indicating that students in the cluster improve less than students in e.g. $\mathcal{C}_1$ and $\mathcal{C}_2$.

This indicates suboptimal development among students in $\mathcal{C}_4$; while we do not see students reverting back, as in $\mathcal{C}_3$, the lower increase in SMOG grade is alarming, indicating students in this cluster may be at risk, and in need of extra attention or encouragement. As for $\mathcal{C}_3$, the total number of words also increases only slightly at a steady pace, from around 900 to 1100.

$\mathcal{C}_5$ Cluster $\mathcal{C}_5$ seems quite distinct from the other clusters. Most notably, students in this cluster have the highest initial similarity, while also decreasing the least amount, ending with a very high similarity of about 0.75. Furthermore, the number of words written is quite high and remains fairly stable, which is quite different from the other clusters, and might be part of the reason the decrease in similarity is as low as it is. Despite the fall in similarity being so low, we still see an increase in SMOG score from about 10.5 to a bit below 11.5, indicating that students are, in fact, improving. A similar pattern occurs for the noun and verb phrases.

The higher-than-average initial SMOG grade and number of words written, indicates that students in $\mathcal{C}_5$ are the initially strong students. While they do develop their writing style, they do not improve as much as students in $\mathcal{C}_1$ and $\mathcal{C}_2$; this could be an indication, that schools do not manage to properly encourage/teach students, who are initially strong.

While not included in the plots, we also investigated several other metrics for the clusters and the set of students in general. Most notably, the average word length increases with time for all clusters. A similar trend was seen in [4], although the study was in a very different setting and time frame.

Summarizing the clusters, the development in SMOG grade was greatest for $\mathcal{C}_1$ and $\mathcal{C}_2$, making those clusters appear the most beneficial for writing style development. While students in $\mathcal{C}_5$ also increased their SMOG grade, they started higher than the students in the other clusters, and did not manage to improve as much as $\mathcal{C}_1$ and $\mathcal{C}_2$. As to $\mathcal{C}_3$ and $\mathcal{C}_4$, they seem to be suboptimal with regards to writing style development, and students in these clusters may need attention.

Looking at Table 2, we see that $\mathcal{C}_3$ and $\mathcal{C}_4$ are the largest individually, indicating that quite a few students are exhibiting suboptimal writing style development. However, the majority of students included in our data are located in $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_5$, indicating optimal or at least fair development through high school.

4.2 Investigating Similarity Between Random Students

As mentioned, we also investigated how the similarity develops between different students, across the time spent in high school. Based on roughly 2 million sampled text pairs from different students, we computed the average similarity between random students to be 0.3470. As seen in Figure 5, the similarity observed among students in $\mathcal{C}_1$ actually drops below this value. This motivated a further investigation of how the similarity between different authors behave on average, conditioned on how long time they each have spend in high school. Based on the samples, we constructed the heat map shown in Figure 7.

The plot shows students starting out similar in writing style and then becoming less similar as time passes. The most surprising thing to notice is that a student in their first year and a student in their third year are equally or even more similar in writing style on average, compared to two different students in their third year. One explanation could be that the initial space of possible writing styles start out small and grows as students are educated, i.e. writing styles among students coming from primary school are fairly similar, but grow more diverse during high school. One would expect some writing styles to diminish or even disappear, but from this data it looks like more new and diverse writing styles develop, than disappear. And not only that; the amount of possible directions for the writing style to develop is so large, that we see first and third year students as equally or more similar on average, than two students both within their third year.

Education is sometimes accused of destroying individuality and/or creativity; these findings indicate the opposite to such claims, at least in regards to writing style.

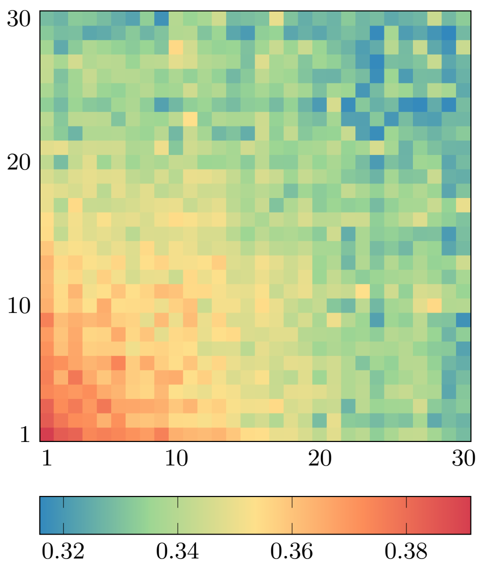


Figure 7: Heat map showing the average similarity between different authors, depending on how long time the two authors have been in high school.

5. CONCLUSIONS AND FUTURE WORK

We trained a Siamese neural network to be able to tell people apart by their writing, and used this network as a similarity function for analyzing the development of writing style in Danish high schools. Writing style development profiles were constructed for 3688 students, and five clusters were found and discussed. Based on quality indicated by noun/verb phrases and SMOG grade, two were found to be optimal, while three were found to be suboptimal, especially two clusters exhibited limited improvement.

The optimal clusters both exhibited a large degree of change in writing style, although at different rates, while the suboptimal clusters showed less development, with one cluster even reverting back to an earlier writing style. The setback in similarity occurred around the summer vacation after the first year. The effect of summer vacation on student learning is highly discussed topic among researchers, teachers, and parents [15]; in the case of the found cluster, the effect appears to be negative.

One tendency, we saw in all clusters, was that writing style changed more when students start writing more words in their essays. It does not seem surprising that your writing style changes as you write more, but it could be an indication of even more: writing style changes, when students are pushed out of their comfort zone, i.e. in the end of their assignments, when they write more than what they usually do. It could be interesting to investigate the scenario, where a student starts writing longer texts: does changes in writing style occur in the entire text, or only near the end, where the student is literally writing more than before?

Furthermore, we saw from Figure 7 how students become less alike, as they go through high school. Specifically, we saw how first year and third year students had higher or equal writing style similarity than two students both in third year, indicating that as Danish students go through high school, their writing styles diverge and become more individual.

5.1 Future Work

It is easy to pose several new questions based on the clusters found and the conclusions made above.

With regards to improving the analysis, using different quality measures tailored to Danish instead of SMOG would be interesting. Another way would be to consider the grades given to the students (as many essays in Danish high school are graded individually), although good writing style is only a requirement for a good grade, but not sufficient.

As mentioned above, one could also consider a more fine grained analysis, by investigating style changes within texts, and maybe even being able to pinpoint exactly where in a text the writing style develops/changes. One could easily imagine drawing inspiration from studies of style breach detection [10, 11, 20].

One could also investigate prediction of writing style development, possibly based on the methods used in this study. This would allow for an early warning system, allowing identification of at-risk students, e.g. students likely to have a setback in writing style due to summer vacation.

The methods used in this study build upon methods used for authorship verification, in which Siamese networks are utilized directly in order to verify authorship [25]. While a sudden deviation in writing style could be an indication of a ghost writer, detecting these reliably using our method will probably not be able to compete with the more direct methods. However, the results obtained here could potentially be used to improve authorship verification techniques, with respect to the fairness perspective: The fact, that the clusters found show such different similarity development, is of interest from a fairness perspective. Fairness is a general issue in machine learning algorithms where the predictions have severe consequences [6, 13]. In the setting of ghost writing detection in high school it is extremely difficult to get non-artificial negative samples and even guaranteeing correctness of labels is rare in large scale data sets. Which makes fairness even more difficult to measure than usual. It could be interesting to check that clusters such as $\mathcal{C}_5$, which would be the cluster most likely to be classified as a false negative, have a representative distribution in regards to gender, race, social status, etc.

Another interesting course of study, would be to further investigate the fact that students seem to become less similar during high school. It could be interesting to pursue this on a larger timescale, perhaps all the way from primary school and on through college. Another take could be to look at how similarity in writing among people behaves with age after they have finished their education. Will the trend continue?

Finally, one could investigate how similarity in writing develops among the genders. Several studies have shown, with some success, that gender can be predicted from writing [17, 21, 22], but no one has settled whether this is due to biology or environment. One could try to answer this question by looking at how similarity in writing style changes with age, while considering three groups: female-female, male-male, female-male. If the cross gender similarity changes faster than same gender similarity, it would be an indication that the differences in writing style are taught, more than it is something you are born with.

6. ACKNOWLEDGMENTS

The work is supported by the Innovation Fund Denmark through the Danish Center for Big Data Analytics Driven Innovation (DABAI) project. The authors would like to thank MaCom for the cooperation.

7. REFERENCES

[1] Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. Gender, genre, and writing style in formal written texts. *TEXT*, 23:321–346, 2003.

[2] Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. Automatically profiling the author of an anonymous text. *Commun. ACM*, 52(2):119–123, February 2009.

[3] Paul B. Diederich. Measuring growth in english. 01 1974.

[4] Fazli Can and Jon M. Patton. Change of writing style with time. *Computers and the Humanities*, 38(1):61–82, Feb 2004.

[5] Jun Chu and Sargur Srihari. Writer identification using a deep neural network. In *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, ICVGIP '14, pages 31:1–31:7, New York, NY, USA, 2014. ACM.

[6] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 797–806, New York, NY, USA, 2017. ACM.

[7] William Weidong Du, Michael Fang, and Ming Shen. Siamese convolutional neural networks for authorship verification. https://www.semanticscholar.org/paper/Siamese-Convolutional-Neural-Networks-for-Du-Fang/c5d1c54511d7f688963cd29a8556d0cf02595890, 2017. Accessed April 2019.

[8] Marjorie Freedman, Alex Baron, Vasin Punyakanok, and Ralph Weischedel. Language use: What can it tell us? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 341–345, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[9] Niels Dalum Hansen, Christina Lioma, Birger Larsen, and Stephen Alstrup. Temporal context for authorship attribution: a study of Danish secondary schools. In *Multidisciplinary information retrieval*, pages 22–40. Springer, 2014.

[10] Daniel Karaś, Martyna Śpiewak, and Piotr Sobecki. OPI-JSA at CLEF 2017: Author Clustering and Style Breach Detection—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland*. CEUR-WS.org, September 2017.

[11] Jamal Ahmad Khan. Style Breach Detection: An Unsupervised Detection Model—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland*. CEUR-WS.org, September 2017.

[12] Moshe Koppel, Jonathan Schler, and Kfir Zigdon. Determining an author's native language by mining a text for errors. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 624–628, New York, NY, USA, 2005. ACM.

[13] Joshua R. Loftus, Chris Russell, Matt J. Kusner, and Ricardo Silva. Causal reasoning for algorithmic fairness. *CoRR*, abs/1805.05859, 2018.

[14] Annie Louis and Ani Nenkova. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics*, 1:341–352, 2013.

[15] Andrew Mceachin and Allison Atteberry. The impact of summer learning loss on measures of school performance. *Education Finance Policy*, 12, 05 2016.

[16] Harry G. McLaughlin. SMOG grading - a new readability formula. *Journal of Reading*, pages 639–646, May 1969.

[17] Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. Predicting age and gender in online social networks. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, SMUC '11, pages 37–44, New York, NY, USA, 2011. ACM.

[18] Emily Pitler and Ani Nenkova. Revisiting readability: A unified framework for predicting text quality. In *EMNLP 2008*, 2008.

[19] Chen Qian, Tianchang He, and Rao Zhang. Deep learning based authorship identification. https://www.semanticscholar.org/paper/Deep-Learning-based-Authorship-Identification-Qian-He/ab0ebe094ec0a44fb0013d640b344d8cfd7adc81, 2018. Accessed April 2019.

[20] Kamil Safin and Rita Kuznetsova. Style Breach Detection with Neural Sentence Embeddings—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland*. CEUR-WS.org, September 2017.

[21] Kosgi Santosh, Romil Bansal, Mihir Shekhar, and Vasudeva Varma. Author profiling: Predicting age and gender from blogs - notebook for pan at clef 2013. In *CLEF*, page 10, 2013.

[22] Anat Rachel Shimoni, Moshe Koppel, and Shlomo Argamon. Automatically Categorizing Written Texts by Author Gender. *Literary and Linguistic Computing*, 17(4):401–412, 11 2002.

[23] V. Spandel. *Creating Writers: Through 6-trait Writing Assessment and Instruction*. Longman, 2001.

[24] Efstathios Stamatatos. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556, March 2009.

[25] Magnus Stavngaard, August Sørensen, Stephan Lorenzen, Niklas Hjuler, and Stephen Alstrup.

Detecting Ghostwriters in High Schools. In *27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2019.

[26] Robert L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, Dec 1953.

[27] Yuting Yang, Juan Cao, Mingyan Lu, Jintao Li, and Chia-Wen Lin. How to write high-quality news on social network? predicting news quality by mining writing style. *CoRR*, abs/1902.00750, 2019.

A.6. On PAC-Bayesian Bounds for Random Forests

Authors:	Stephan Sloth Lorenzen
	Christian Igel
	Yevgeny Seldin
Presented at:	European Conference on Machine Learning (ECML)
Journal:	Machine Learning
Year:	2019
Vol:	108
Number:	8-9
Pages:	1-20
DOI:	http://dx.doi.org/10.1007/s10994-019-05803-4
arXiv:	`https://arxiv.org/abs/1810.09746`
Source code:	`https://github.com/StephanLorenzen/RF-with-Bounds`

Machine Learning manuscript No.
(will be inserted by the editor)

On PAC-Bayesian Bounds for Random Forests

Stephan S. Lorenzen · Christian Igel · Yevgeny Seldin

Received: date / Accepted: date

Abstract Existing guarantees in terms of rigorous upper bounds on the generalization error for the original random forest algorithm, one of the most frequently used machine learning methods, are unsatisfying. We discuss and evaluate various PAC-Bayesian approaches to derive such bounds. The bounds do not require additional hold-out data, because the out-of-bag samples from the bagging in the training process can be exploited.

A random forest predicts by taking a majority vote of an ensemble of decision trees. The first approach is to bound the error of the vote by twice the error of the corresponding Gibbs classifier (classifying with a single member of the ensemble selected at random). However, this approach does not take into account the effect of averaging out of errors of individual classifiers when taking the majority vote. This effect provides a significant boost in performance when the errors are independent or negatively correlated, but when the correlations are strong the advantage from taking the majority vote is small. The second approach based on PAC-Bayesian C-bounds takes dependencies between ensemble members into account, but it requires estimating correlations between the errors of the individual classifiers. When the correlations are high or the estimation is poor, the bounds degrade.

In our experiments, we compute generalization bounds for random forests on various benchmark data sets. Because the individual decision trees already perform well, their predictions are highly correlated and the C-bounds do not lead to satisfactory results. For the same reason, the bounds based on the analysis of Gibbs classifiers are typically superior and often reasonably tight. Bounds based on a validation set coming at the cost of a smaller training set gave better performance guarantees, but worse performance in most experiments.

Keywords PAC-Bayesian analysis · Random Forests · majority vote

The work is supported by the Innovation Fund Denmark through the Danish Center for Big Data Analytics Driven Innovation (DABAI) project.

S. Lorenzen · C. Igel · Y. Seldin
Department of Computer Science, University of Copenhagen
E-mail: {lorenzen,igel,seldin}@di.ku.dk

1 Introduction

A *random forest* is one of the most successful machine learning algorithms (Breiman, 2001). It is easy to use and to parallelize and often achieves high accuracies in practice (Fernández-Delgado et al., 2014). In a survey on the machine learning competition website kaggle.com[1], 46% of 16.000 surveyed users claimed to use the algorithm in their daily work. A random forest for classification predicts based on the (possibly weighted) *majority vote* of a set (an *ensemble*) of weaker classifiers, concretely *decision trees*. The model was first presented by Breiman (2001), who provides an initial analysis and some theoretical bounds, showing that the strength of the random forest depends on the strength of individual trees and their correlation. Despite its popularity in practice, the algorithm is still not well understood theoretically (Arlot and Genuer, 2014; Biau, 2012; Denil et al., 2014), the main reason being that the model is difficult to analyse because of the dependencies between the induced partitions of the input space and the predictions within the partitions (Arlot and Genuer, 2014). The conceptually simpler *purely random forests* (Breiman, 2002) avoids these dependencies by creating a random partitioning independent of the training data. This is done by selecting features and splits at random. Biau et al. (2008) show the purely random forests to be *consistent* under some assumptions on the distribution of the input variables. Several modification of the random forest have been introduced in the literature, most of them being in between the standard random forest and the purely random forest in the sense that extra randomness is added to get independent partitions (Geurts et al., 2006; Genuer, 2010). For instance, Wang et al. (2016) introduce the *Bernoulli random forests*, which relies on Bernoulli random variables for randomly choosing the strategy for partitioning the input space, and prove this model to be consistent. Likewise, Denil et al. (2014) give a variant based on sampling of predictions in a partition for determining best splits, and prove this variant to be consistent. Theoretical bounds on the expected loss have been considered by Genuer (2010) in the case of regression tasks when the input space is one-dimensional. Arlot and Genuer (2014) consider the generalization error for the purely random forest in relation to the number of trees. All these have nice analytical properties, but these come at the expense of degradation in empirical performance compared to the standard random forest. Accordingly, the original random forest still remains the best choice in practice (Wang et al., 2016), despite the lack of strong theoretical guarantees.

This study considers the application of theoretical bounds based on PAC-Bayesian analysis to the standard random forest as given by Breiman (2001). Here PAC stands for the Probably Approximately Correct frequentist learning model (Valiant, 1984). PAC-Bayesian approaches are usually used for analysing the expected loss of *Gibbs classifiers*. Gibbs classifiers are randomized classifiers that make predictions by applying a hypothesis drawn from a hypothesis class $\mathcal{H}$ according to some distribution ρ on $\mathcal{H}$ (McAllester, 1998; Seeger, 2002; Thiemann et al., 2017). While generalization bounds for Gibbs classifier may at first not seem directly applicable to majority vote classifiers, they are in fact closely related. It can be shown that the loss of a ρ-weighted majority vote classifier is at most twice that of the associated Gibbs classifier, meaning that any bound for a Gibbs classifier leads to a bound for the majority vote (Germain et al., 2015). However, such adaptation of the bounds for Gibbs classifiers typically provides relatively weak bounds for majority vote classifiers, because the bounds for Gibbs classifiers do not take into account dependencies between individual classifiers. One of the main reasons for the good performance of majority vote classifiers is that when the errors of individual classifiers are independent they tend to av-

[1] http://www.kaggle.com/surveys/2017

erage out (Breiman, 2001). Therefore, the majority vote may perform very well even when the individual classifiers are weak (i.e., only slightly better than random guessing). In this case, application of PAC-Bayesian bounds for the Gibbs classifier to the majority vote yields suboptimal results.

This has motivated the development of PAC-Bayesian bounds designed specifically for averaging and majority vote classifiers (Germain et al., 2015; McAllester, 1999; Oneto et al., 2018). One such bound is the *C-bound*, given by Germain et al. (2015), which is based on the *margin* of the classifier. In contrast to the bounds for Gibbs classifiers, the C-bound takes the correlations between the individual classifiers into account and could potentially yield tighter bounds in the case described above. However, in the case with strong individual classifiers and high correlation (as is the case for random forests), the C-bound deteriorates (Germain et al., 2015) – in contrast to the Gibbs classifier bounds.

In this study, several of the above mentioned bounds are applied to the standard random forest setting, where trees are trained using *bagging*, that is, using different random subsets of the training data (Breiman, 2001, 1996a). Since validation sets for individual trees are constructed as part of the training procedure when using bagging, the theoretical bounds come "for free" in the sense that no separate data needs to be reserved for evaluation. We compare the quality of bounds obtained in this setting with bounds obtained by leaving out a validation set for evaluation. We also consider optimization of the weighting of the voters by minimization of the theoretical bounds (Thiemann et al., 2017; Germain et al., 2015).

2 Background

We consider supervised learning. Let $S = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be an independent identically distributed sample from $\mathcal{X} \times \mathcal{Y}$, drawn according to an unknown distribution D. A hypothesis is a function $h : \mathcal{X} \to \mathcal{Y}$, and $\mathcal{H}$ denotes a space of hypotheses. We evaluate a hypothesis h by a bounded loss function $\ell : \mathcal{Y}^2 \to [0, 1]$. The expected loss of h is denoted by $L(h) = \mathbb{E}_{(X,Y)\sim D}[\ell(h(X), Y)]$ and the empirical loss of h on a sample S is denoted by $\hat{L}(h, S) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(X_i), Y_i)$. In this study, we focus on classification. Given a set of hypotheses $\mathcal{H}$ and a distribution ρ on $\mathcal{H}$, the *Gibbs classifier* h_G is a stochastic classifier, which for each input X randomly draws a hypothesis $h \in \mathcal{H}$ according to ρ and predicts $h(X)$ (Seeger, 2002). The expected loss of the Gibbs classifier is given by $L^{\text{Gibbs}}(h_G) = \mathbb{E}_{h\sim\rho}[L(h)]$, and the empirical loss of h_G on a sample S is given by $\hat{L}^{\text{Gibbs}}(h_G, S) = \mathbb{E}_{h\sim\rho}\left[\hat{L}(h, S)\right]$.

Closely related to the random Gibbs classifier are *aggregate classifiers*, whose predictions are based on weighted aggregates over $\mathcal{H}$. The ρ-weighted *majority vote* h_M predicts $h_M(X) = \operatorname{argmax}_{Y\in\mathcal{Y}} \sum_{h\in\mathcal{H}\wedge h(X)=Y} \rho(h)$. When discussing majority vote classifiers, it is convenient to define the *margin* realised on a pattern (X, Y) (Breiman, 2001):

$$\mathcal{M}_\rho(X, Y) = \mathbb{P}_{h\sim\rho}[h(X) = Y] - \max_{j\neq Y} \mathbb{P}_{h\sim\rho}[h(X) = j], \tag{1}$$

and the expected value of the margin $\mathcal{M}_\rho = \mathbb{E}_{(X,Y)\sim D}\left[\mathcal{M}_\rho(X, Y)\right]$. Note, that a large margin indicates a strong classifier. The expected loss of h_M is then given by $L^{\text{MV}}(h_M) = \mathbb{P}_{(X,Y)\sim D}\left[\mathcal{M}_\rho(X, Y) \leq 0\right]$, and the empirical loss $\hat{L}^{\text{MV}}(h_M, S) = \mathbb{P}_{(X,Y)\sim S}\left[\mathcal{M}_\rho(X, Y) \leq 0\right]$, where we use $(X, Y) \sim S$ to denote a uniform distribution over the sample.

The Kullback-Leibler divergence between two distributions π and ρ is denoted by $\text{KL}(\rho\|\pi)$ and between Bernoulli distributions with biases p and q by $\text{kl}(p\|q)$. Furthermore, let $\mathbb{E}_D[\cdot]$ denote $\mathbb{E}_{(X,Y)\sim D}[\cdot]$ and $\mathbb{E}_\rho[\cdot]$ denote $\mathbb{E}_{h\sim\rho}[\cdot]$. Finally, u denotes the uniform distribution.

2.1 Random Forests.

Originally described by Breiman (2001), the random forest is a majority vote classifier, where individual voters are decision trees. In the standard random forest setting, every voter has equal weight (i.e., $\rho = u$). Let $T \subset \mathcal{X} \times \mathcal{Y}$ denote training patterns drawn according to D. A random forest is constructed by independently constructing *decision trees* $h_1, h_2, \ldots, h_m$ (Hastie et al., 2009), where each h_i is trained on $T_i \subseteq T$. A tree is constructed recursively, starting at the root. At each internal node, a threshold θ and a feature j are chosen, and the data set T' corresponding to the current node is then split into $\{X \mid X_j \leq \theta\}$ and $\{X \mid X_j > \theta\}$. θ and j are chosen according to some splitting criterion, usually with the goal of maximizing the *information gain* for each split, making the new subsets more homogeneous (Hastie et al., 2009). Splitting is stopped when a node is completely *pure* (only one class is present) or by some other stopping criterion (e.g., maximum tree depth). The tree construction is randomized (Breiman, 1996a, 2001). First, the selection of data sets T_i for training individual trees is randomized. They are generated by the bagging procedure described below. Second, only a random subset of all features is considered when splitting at each node (Breiman, 2001; Hastie et al., 2009).

Bagging is a general technique used for aggregated predictors (Breiman, 1996a), which generates the training sets T_i for the individual predictors by sampling $|T|$ points from T with replacement. The individual training sets $T_1, T_2, \ldots$ are known as *bootstrap samples*. Because of sampling with replacement, not all patterns of T are expected to be in each T_i. Let $\bar{T}_i = T \setminus T_i$ denote the patterns not sampled for T_i. $\bar{T}_i$ can now be used to give an unbiased estimate of the individual classifier h_i. The expected number of unique patterns in T_i is approximately $\left(1 - \frac{1}{e}\right)|T| \simeq 0.632|T|$, leaving us with slightly more than one third of the training patterns for the validation sets (Breiman, 1996a).

Bagging also allows us to compute *out-of-bag (OOB)* estimates. For each training pattern $(X, Y) \in T$, a majority vote prediction is computed over all voters h_i with $(X, Y) \notin T_i$. The empirical loss computed over these predictions is known as the OOB estimate, which we denote by $\hat{L}^{\mathrm{MV}}_{\mathrm{OOB}}(h_M, T)$. Empirical studies have shown that the OOB estimate on the training data is a good estimator of the generalization error (Breiman, 1996b).

Furthermore, the sets $\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_m$ can be used to compute the empirical error of the associated ρ-weighted Gibbs classifier h_G by

$$\hat{L}^{\mathrm{Gibbs}}_{\mathrm{OOB}}(h_G, T) = \mathbb{E}_\rho \left[\frac{1}{|\bar{T}_i|} \sum_{(X,Y) \in \bar{T}_i} \ell(h_i(X), Y) \right],$$

and by considering $\bar{T}_i \cap \bar{T}_j$, we can also estimate the correlation between trees h_i and h_j, an important ingredient in bounds for majority vote classifiers.

3 PAC-Bayesian Bounds for Majority Vote Classifiers

We now give an overview of the PAC-Bayesian bounds we apply to bound the expected loss of random forests. PAC-Bayesian bounds have a form of a trade-off between ρ-weighted empirical loss of hypotheses in $\mathcal{H}$ and the complexity of ρ, which is measured by its Kullback-Leibler divergence from a prior distribution π. The prior must be selected before the data is observed and can be used to incorporate domain knowledge, while the posterior ρ can be chosen based on the data.

3.1 PAC-Bayesian Bounds for Gibbs Classifiers.

The ρ-weighted majority vote classifier h_M is closely related to the ρ-parameterized Gibbs classifier h_G. Whenever the majority vote makes a mistake, it means that more than a ρ-weighted half of the voters make a mistake. Thus, the expected loss of the majority vote classifier $L^{\mathrm{MV}}(h_M)$ is at most twice the expected loss of the Gibbs classifier $L^{\mathrm{Gibbs}}(h_G)$,

$$L^{\mathrm{MV}}(h_M) \leq 2L^{\mathrm{Gibbs}}(h_G) \tag{2}$$

(Mcallester, 2003; Langford and Shawe-Taylor, 2002; Germain et al., 2015). Therefore, any bound on $L^{\mathrm{Gibbs}}(h_G)$ provides a bound on the corresponding $L^{\mathrm{MV}}(h_M)$. We consider the following inequality originally due to Seeger (2002), which we refer to as the PBkl-bound (**P**AC-**B**ayesian **kl**):

Theorem 1 (PBkl-bound, Seeger, 2002) *For any probability distribution π over $\mathcal{H}$ that is independent of S and any $\delta \in (0,1)$, with probability at least $1-\delta$ over a random draw of a sample S, for all distributions ρ over $\mathcal{H}$ simultaneously:*

$$\mathrm{kl}\left(\hat{L}^{\mathrm{Gibbs}}(h_G, S) \middle\| L^{\mathrm{Gibbs}}(h_G)\right) \leq \frac{\mathrm{KL}(\rho\|\pi) + \ln\frac{2\sqrt{n}}{\delta}}{n}.$$

A slightly tighter bound can be obtained by using

$$\xi(n) = \sum_{k=0}^{n} \binom{n}{k} \left(\frac{k}{n}\right)^k \left(1 - \frac{k}{n}\right)^{n-k} \tag{3}$$

instead of $2\sqrt{n}$ in the bound above, because we have $\sqrt{n} \leq \xi(n) \leq 2\sqrt{n}$ (Maurer, 2004).

In order to use Theorem 1 in the bagging setting, we need to make a small adjustment. The empirical Gibbs loss $\hat{L}^{\mathrm{Gibbs}}(h_G, T)$ is computed using $\bar{T}_1, \bar{T}_2, \ldots$, and since these sets have different sizes, in order to apply the PBkl-bound, we use $n = \min_i \left(|\bar{T}_i|\right)$. That this is a valid strategy can easily be seen by going through the proof of Theorem 1, see Thiemann et al. (2017).

Theorem 1 may also be applied to the final majority vote classifier h_G if a separate validation set is left out. In this case, $|\mathcal{H}| = 1$, $\mathrm{KL}(\rho\|\pi) = 0$, and $\hat{L}^{\mathrm{Gibbs}}(h_M, S) = \hat{L}^{\mathrm{MV}}(h_M, S)$. A separate validation set implies that the data available at training time has to be split, and, therefore, the actual training set gets smaller. While $\hat{L}^{\mathrm{Gibbs}}(h_M, S)$ may be larger due to the smaller training data set size, the bound does no longer suffer from the factor 2 in (2). We will consider this way of bounding $L^{\mathrm{MV}}(h_M)$ as an additional baseline denoted as *SH-bound* (**S**ingle **H**ypothesis). Note that this bound requires the separate validation set and, thus, cannot be applied in the bagging setting.

3.2 PAC-Bayesian Bounds for Majority Vote Classifiers.

The PBkl-bound provides a tight bound for the Gibbs classifier, but the associated bound for the majority vote classifier may be loose. This is because the bound for the Gibbs classifier does not take correlation between individual classifiers into account. The individual classifiers may be weak (i.e., $L(h_i)$ close to $\frac{1}{2}$) leading to a weak Gibbs classifier, but if the correlations between the classifiers are low, the errors tend to cancel out when voting, giving a stronger majority vote classifier (Germain et al., 2015). The generalization bounds for Gibbs classifiers do not capture this, as they depend only on the strength of the individual classifiers. In order to get stronger generalization guarantees for majority vote classifiers, we need bounds that incorporate information about the correlations between errors of classifiers as well, as already pointed out by Breiman (2001).

Germain et al. (2015) propose to use the C-bound for this purpose, which is based on the margin of the majority vote classifier. They consider only the case where the output space is binary, $\mathcal{Y} = \{-1, 1\}$, and (1) becomes $\mathcal{M}_\rho(X,Y) = Y\left(\sum_{h\in\mathcal{H}} \rho(h)h(X)\right)$. With the first moment $\mathcal{M}_\rho = \mathbb{E}_D\left[\mathcal{M}_\rho(X,Y)\right]$, the second moment is given by $\mathcal{M}_{\rho^2} = \mathbb{E}_D\left[\left(\mathcal{M}_\rho(X,Y)\right)^2\right] = \mathbb{E}_{h_1,h_2\sim\rho^2}\left[\mathbb{E}_D\left[h_1(X)h_2(X)\right]\right]$. Then the C-bound for the expected loss of the ρ-weighted majority vote classifier reads:

Theorem 2 **(C-bound, Germain et al., 2015)** *For any distribution ρ over $\mathcal{H}$ and any distribution D on $\mathcal{X}\times\{-1,1\}$, if $\mathcal{M}_\rho > 0$, we have*

$$L(h_M) \le 1 - \frac{\mathcal{M}_\rho^2}{\mathcal{M}_{\rho^2}}.$$

The theorem follows from the one-sided Chebyshev inequality applied to the loss of h_M. As the first and second moments are usually not known, Germain et al. offer several ways to bound them empirically. They start by showing that

$$\mathcal{M}_\rho = 1 - 2L^{\text{Gibbs}}(h_G) \tag{4}$$

meaning that the first moment of the margin can be bounded by the use of the PBkl-bound (Theorem 1). For the second moment, we have that

$$\mathcal{M}_{\rho^2} = 1 - 2d_\rho, \tag{5}$$

where $d_\rho = \frac{1}{2}\left[1 - \mathbb{E}_D\left[\left(\mathbb{E}_\rho\left[h(X)\right]\right)^2\right]\right]$ is the *disagreement* between individual classifiers. Together with the C-bound, the relations above confirm the observations made by Breiman (2001): The strength of h_M depends on having strong individual classifiers (low $L^{\text{Gibbs}}(h_G)$, i.e., large $\mathcal{M}_\rho$) and low correlation between classifiers (high disagreement d_ρ, i.e., low $\mathcal{M}_{\rho^2}$).

By (4), the first moment of the margin can be bounded by the use of the PBkl-bound (Theorem 1), while by (5) the second moment can be bounded using a lower bound on d_ρ. With $\hat{d}_\rho^S$ denoting the empirical disagreement computed on S, d_ρ can be lower bounded by the smallest d satisfying (Germain et al., 2015)

$$\mathrm{kl}\left(\hat{d}_\rho^S \middle\| d\right) \le \frac{2\,\mathrm{KL}(\rho\|\pi) + \ln\frac{\xi(n)}{\delta}}{n}.$$

Like in the case of Theorem 1, solutions to the above inequality can be computed by a root-finding method. This leads to the following empirical C-bound, which we denote the $C1$-bound:

Theorem 3 **($C1$-bound, Germain et al., 2015)** *For any probability distribution π over $\mathcal{H}$ that is independent of S and any $\delta\in(0,1)$, with probability at least $1-\delta$ over a random draw of a sample S, for all distributions ρ over $\mathcal{H}$ simultaneously*

$$L^{\text{MV}}(h_M) \le 1 - \frac{(1-2b)^2}{(1-2d)}.$$

Here b is an upper bound on $L^{\text{Gibbs}}(h_G)$, which can be found by Theorem 1, and d is a lower bound on d_ρ.

The $C1$-bound allows direct bounding of $L^{\text{MV}}(h_M)$. However, Germain et al. (2015) also provide another bound based on Theorem 2, which does not require bounding $L^{\text{Gibbs}}(h_G)$ and d_ρ separately. First, we let $e_\rho = \mathbb{E}_{h_1,h_2\sim\rho^2}\left[\mathbb{E}_D\left[I(h_1(X)\neq Y)I(h_2(X)\neq Y)\right]\right]$ denote the *expected joint error* and $\hat{e}^S_\rho$ denote the empirical joint error computed on S. Then the loss of the associated Gibbs classifier can be written as $L^{\text{Gibbs}}(h_G) = \frac{1}{2}\left(2e_\rho + d_\rho\right)$. The next bound is then based on bounding d_ρ and e_ρ simultaneously, by bounding the KL-divergence between two *trivalent* random variables. A variable X is trivalent if $\mathbb{P}(X = x_1) = p_1$, $\mathbb{P}(X = x_2) = p_2$ and $\mathbb{P}(X = x_3) = 1 - p_1 - p_2$, and similar to $\text{kl}(\cdot\|\cdot)$, $\text{kl}(p_1, p_2\| q_1, q_2)$ denotes the KL-divergence between two trivalent random variables with parameters $(p_1, p_2, 1-p_1-p_2)$ and $(q_1, q_2, 1-q_1-q_2)$.

Using the above and a generalization of the PAC-Bayes inequality to trivalent random variables, Germain et al. derive the following bound, which we refer to as the $C2$-bound:

Theorem 4 **($C2$-bound, Germain et al., 2015)** *For any probability distribution π over $\mathcal{H}$ independent of S and any $\delta \in (0,1)$, with probability at least $1-\delta$ over a random draw of a sample S, for all distributions ρ over $\mathcal{H}$ simultaneously*

$$L^{\text{MV}}(h_M) \leq \sup_{d,e}\left(1 - \frac{(1-(2e+d))^2}{1-2d}\right),$$

where the supremum is over all d and e satisfying

$$\text{kl}\left(\hat{d}^S_\rho, \hat{e}^S_\rho \middle\| d, e\right) \leq \frac{2\,\text{KL}(\rho\|\pi) + \ln\frac{\xi(n)+n}{\delta}}{n}, \qquad d \leq 2\left(\sqrt{e} - e\right), \qquad 2e + d < 1. \tag{6}$$

Again, we need to make adjustments in order to apply the $C1$-bound and $C2$-bound in the bagging setting. When lower bounding the disagreement and the joint error in Theorem (4), we consider the empirical disagreement $\hat{d}^T_\rho$ (and joint error $\hat{e}^T_\rho$) between h_i and h_j estimated on $\bar{T}_i \cap \bar{T}_j$ and choose $n = \min_{i,j}\left(|\bar{T}_i \cap \bar{T}_j|\right)$ accordingly.

3.3 Optimizing the Posterior Distribution.

Aside from providing guarantees on expected performance, PAC-Bayesian bounds can be used to tune classifiers. The prior distribution π must be chosen before observing the data, but we are free to choose the posterior distribution ρ afterwards, for instance one could choose ρ such that the empirical loss $\hat{L}^{\text{MV}}(h_M, S)$ is minimized.

Breiman has applied *boosting* (Schapire and Singer, 1999) to random forests in order to optimize the weighting of the vote, finding that it improved the accuracy in some cases (Breiman, 2001). We instead consider optimization of the posterior by minimizing the theoretical bounds (Thiemann et al., 2017; Germain et al., 2015). However, none of the bounds provided above can easily be used to directly optimize ρ, because they are non-convex in ρ (Thiemann et al., 2017). Thiemann et al. (2017) and Germain et al. (2015) came up with two different ways to resolve the convexity issue.

Thiemann et al. apply a relaxation of Theorem 1 based on Pinsker's inequality, which leads to the following result that we refer to as the λ-bound:

Theorem 5 **(λ-bound, Thiemann et al., 2017)** *For any probability distribution π over $\mathcal{H}$ that is independent of S and any $\delta \in (0,1)$, with probability at least $1-\delta$ over a random draw of a sample S, for all distributions ρ over $\mathcal{H}$ and $\lambda \in (0,2)$ simultaneously*

$$L^{\text{Gibbs}}(h_G) \le \frac{\hat{L}^{\text{Gibbs}}(h_G,S)}{1-\frac{\lambda}{2}} + \frac{\mathrm{KL}(\rho\|\pi) + \ln\frac{2\sqrt{n}}{\delta}}{\lambda\left(1-\frac{\lambda}{2}\right)n}. \tag{7}$$

They show that the λ-bound is convex in ρ and in λ (but not jointly convex). They give an iterative update procedure, which alternates between updating λ and ρ, and prove that, under certain conditions, the procedure is guaranteed to converge to the global minimum.

Germain et al. (2015) state a version of the C-bound that is suited for optimization of ρ. However, the bound is restricted to *self-complemented hypothesis classes* and posteriors *aligned* on the prior. A hypothesis class is said to be self-complemented, if $h \in \mathcal{H} \Leftrightarrow -h \in \mathcal{H}$, where $-h$ is a hypothesis that always predicts the opposite of h (in binary prediction). A posterior ρ is said to be aligned on π if $\rho(h)+\rho(-h) = \pi(h)+\pi(-h)$. Thus, the final statement of the bound, which we denote by *$C3$-bound*, becomes:

Theorem 6 **($C3$-bound, Germain et al., 2015)** *For any self-complemented hypothesis set $\mathcal{H}$, any probability distribution π over $\mathcal{H}$ independent of S and any $\delta \in (0,1)$, with probability at least $1-\delta$ over a random draw of a sample S, for all distributions ρ aligned with π simultaneously*

$$L^{\text{MV}}(h_M) \le 1 - \frac{(1-2r)^2}{(1-2d)},$$

where

$$r = \min\left(\frac{1}{2}, \hat{L}^{\text{Gibbs}}(h_G,S) + \sqrt{\frac{\ln\frac{2\xi(n)}{\delta}}{2n}}\right), \qquad d = \max\left(0, \hat{d}_\rho^S - \sqrt{\frac{\ln\frac{2\xi(n)}{\delta}}{2n}}\right).$$

The authors show how to minimize the bound in Theorem 6 over the posterior ρ, by solving a quadratic program. The quadratic program requires a hyperparameter μ, used to enforce a minimum value of the first moment of the margin. μ can be chosen by cross validation (Germain et al., 2015). Furthermore, they note how the restriction to aligned posteriors acts as regularization.

For both the λ-bound and the $C3$-bound, we need to make the same adjustments as for the PBkl-bound and the C-bound, that is, we choose $n = \min_{i,j}\left(|\bar{T}_i \cap \bar{T}_j|\right)$. When applying the optimization procedure of the $C3$-bound, we also need to make sure that the $\mathcal{H}$ is self-complemented; given a set of hypotheses, this can be done by copying all hypotheses and inverting their predictions.

4 Experiments

We have applied the bounds of Section 3, summarized in Table 1, in different random forest settings. First, we considered the **standard setting with bagging** and used the sets $\bar{T}_1, \bar{T}_2, \ldots$ for evaluation and computation of the bounds as described in Section 3. The posterior distribution ρ was taken uniform and not optimized. Then we considered a **setting with a separate validation set** T_{val}. The majority vote bounds suffer from the low number of

Table 1 Overview of the bounds that we apply in Section 4 with references to the corresponding theorems in Section 3.

Name	Bound	Theorem	Reference
PBkl-bound	$\mathrm{kl}\left(\hat{L}^{\text{Gibbs}}(h_G, S) \Vert L^{\text{Gibbs}}(h_G)\right) \leq \frac{\mathrm{KL}(\rho \Vert \pi) + \ln \frac{2\sqrt{n}}{\delta}}{n}$	Thm. 1	Seeger, 2002
C1-bound	$L^{\text{MV}}(h_M) \leq 1 - \frac{(1-2b)^2}{(1-2d)}$	Thm. 3	Germain et al., 2015
C2-bound	$L^{\text{MV}}(h_M) \leq \sup_{d,e}\left(1 - \frac{(1-(2e+d))^2}{1-2d}\right)$	Thm. 4	Germain et al., 2015
SH-bound	$\mathrm{kl}\left(\hat{L}^{\text{MV}}(h_M, S) \Vert L^{\text{MV}}(h_M)\right) \leq \frac{\ln \frac{2\sqrt{n}}{\delta}}{n}$		PBkl with $\lvert\mathcal{H}\rvert = 1$
λ-bound	$L^{\text{Gibbs}}(h_G) \leq \frac{\hat{L}^{\text{Gibbs}}(h_G,S)}{1-\frac{\lambda}{2}} + \frac{\mathrm{KL}(\rho \Vert \pi) + \ln \frac{2\sqrt{n}}{\delta}}{\lambda\left(1-\frac{\lambda}{2}\right)n}$	Thm. 5	Thiemann et al., 2017
C3-bound	$L^{\text{MV}}(h_M) \leq 1 - \frac{(1-2r)^2}{(1-2d)}$	Thm. 6	Germain et al., 2015

training patterns available for evaluating the correlation between classifiers. Therefore, we also evaluated the quality of the bounds when a separate validation set T_{val} was set aside before training. T_{val} was then used in addition to $\bar{T}_1, \bar{T}_2, \ldots$, when evaluating and computing the bounds. Again, the posterior distribution $\rho = u$ was not optimized. Finally, we looked at **random forests with optimized posteriors**. We used bagging and the bounds in Theorems 5 and 6 to optimize the posterior distribution ρ.

For all settings, the accuracy of the final majority vote classifier h_M is also of interest. Hence, a separate test set T_{ext} is left out in each setting. This set is used only for evaluating the final classifier by $\hat{L}^{\text{MV}}(h_M, T_{\text{ext}})$. We are mainly concerned with the tightness of the bounds when individual voters are strong. Therefore, all features are considered in each split during the training of the random forest (using Gini impurity as splitting criterion), and trees are trained until all leaves are pure (see, e.g., Gieseke and Igel, 2018, for arguments why this can be beneficial).

To study how the bounds depend on the strengths of the individual classifiers, we varied the maximum tree depth and the number of features considered in each split in the first two settings. This allows us to investigate the evolution of the bounds as the strength of individual classifiers increases by going from decision stumps to full-grown trees. We either set the number of random features considered for splitting to the maximum number or, to further weaken the classifiers, to a single random feature. We restricted these experiments to two data sets.

Experiments were run on several binary UCI data sets (see left part of Table 2). For each data set, all patterns with one or more missing features were removed. Since the C-bound is only analysed for binary classification, we restrict ourselves to binary tasks. The number of trees m for any data set of size N was chosen as the largest value from $\{100, 200, 500, 1000\}$ that was smaller than $N/4$.

For each setting, $N/2$ patterns were randomly sampled for T_{ext}. In the first and third settings, all remaining patterns were used for training. In the second setting, a further $N/4$ patterns were sampled for T_{val}, with the remaining patterns used for training, see Figure 1 for an illustration.

When evaluating the bounds, we chose $\pi = u$ and $\delta = 0.05$.

Table 2 The PBkl-bound, $C1$-bound, $C2$-bound and SH-bound computed for the binary UCI data sets in the bagging and validation set settings. In both settings, the majority vote loss on T_{ext} is given as an estimate of the accuracy of the trained classifier denoted as *test score*. The best bound within an experiment is marked **bold**, while *italics* is used to indicate trivial bounds (≥ 0.5).

			Bagging				Validation set				
Data set	n	d	Test score	PBkl	$C1$	$C2$	Test score	PBkl	$C1$	$C2$	SH
Adult	45222	14	0.152	**0.428**	*0.525*	*0.520*	0.154	0.426	0.500	0.479	**0.169**
Credit-A	653	15	0.144	***0.603***	*0.853*	*0.990*	0.135	*0.563*	*0.790*	*0.788*	**0.294**
Haberman	306	3	0.333	*>1*	*>1*	*>1*	0.333	*>1*	*>1*	*>1*	***0.577***
Heart	297	13	0.228	***0.960***	*>1*	*>1*	0.282	*0.822*	*0.973*	*0.986*	**0.412**
ILPD	579	10	0.307	*>1*	*>1*	*>1*	0.307	*0.955*	*0.999*	*>1*	**0.441**
Ionosphere	351	34	0.108	***0.691***	*0.920*	*>1*	0.125	*0.649*	*0.878*	*0.890*	**0.299**
Letter:AB	20000	16	0.010	**0.124**	0.244	0.408	0.015	0.140	0.242	0.186	**0.035**
Letter:DO	20000	16	0.045	**0.216**	0.391	*0.530*	0.067	0.227	0.362	0.294	**0.072**
Letter:OQ	20000	16	0.051	**0.288**	0.491	*0.605*	0.059	0.323	0.474	0.404	**0.119**
Mushroom	8124	22	0.000	**0.011**	0.025	0.078	0.001	0.013	0.028	0.037	**0.008**
Sonar	208	60	0.221	*>1*	*>1*	*>1*	0.250	*>1*	*>1*	*>1*	***0.510***
Tic-Tac-Toe	958	9	0.088	***0.627***	*0.847*	*0.934*	0.142	*0.749*	*0.892*	*0.805*	**0.221**
USvotes	232	16	0.034	***0.526***	*0.810*	*>1*	0.052	0.497	*0.764*	*0.750*	**0.228**
WDBC	569	30	0.053	**0.430**	*0.696*	*0.945*	0.063	0.327	*0.543*	0.489	**0.102**

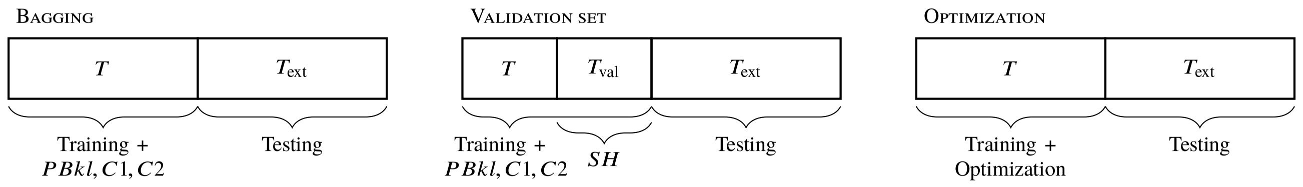


Fig. 1 Overview of data split for each of the three settings. Note, that the bounds computed on T, as well as the optimization in the third setting, are only based on the hold-out sets, $\bar{T}_i$.

4.1 Random forest with bagging.

We started with the original random forest setting, where an individual tree h_i is trained on a bootstrap sample T_i of size $|T|$, drawn with replacement from the training set T consisting of half the data with the other half T_{ext} used for evaluating the final classifier. As mentioned, the posterior distribution ρ was chosen uniform. In this experiment, T comprised all available data. The empirical Gibbs loss was evaluated using $\bar{T}_i = T \setminus T_i$ and the empirical disagreement and joint error between two trees h_i and h_j using $\bar{T}_i \cap \bar{T}_j$.

We considered the PBkl-bound and the two empirical C-bounds, $C1$-bound and $C2$-bound, with sample sizes n calculated as described in Section 3. Furthermore, the trained classifier h_M was evaluated on T_{ext}.

Table 2 (middle) lists the results. The test score $\hat{L}^{\text{MV}}(h_M, T_{\text{ext}})$ provides an estimate of the accuracy of the classifier. The PBkl-bound always gave the tightest bounds. For 6 out of the 14 data sets the bound was below 0.5. The better performance of the PBkl-bound is explained by the high accuracy of the individual trees. As mentioned by Germain et al. (2015) and discussed in Section 3, the C-bound degrades when the individual classifiers are strong. Thus, the PBkl-bound including the factor of 2 coming from (2) is tighter. Furthermore, for the C-bounds the bagging setting is particularly difficult, because there is only a small amount of data available for estimating correlations. This is especially true for the $C2$-bound, since it relies only on the intersection between the two samples $\bar{T}_i$ and $\bar{T}_j$, which may be small.

In the bagging setting we get the bounds "for free" in the sense that all evaluations are based on the $\bar{T}_i$ sets, which are by-products of the training, and we do not have to set aside a separate validation set. Thus, more data is available for selecting the hypothesis.

Figure 2 shows the evolution of the bounds as the strength of the individual voters varies for the data sets Letter:AB and Mushroom. Voter strength was controlled by increasing the maximum allowed tree depth until only pure trees were obtained, and by feature selection during splits, that is, using either the best feature (stronger voters) or a random feature (weaker voters).

From the figure, we see that the PBkl-bound is tighter than both the $C1$-bound and the $C2$-bound, even though the C-bounds are expected to perform better, when individual classifiers are weak. However, this theoretical advandtage is outweighed by the low amount of data available for bounding the disagreement/joint error, that is, $n = \min_{i,j}\left(|\bar{T}_i \cap \bar{T}_j|\right)$ is very small, leading to loose bounds.

4.2 Random forest with a separate validation set.

As a reference, we considered the scenario where a separate validation set T_{val} was set aside before the random forest was trained, which allows for a better estimate of the correlations in the C-bounds. Recall that a separate test set T_{ext} was set aside for evaluating the classifier beforehand. Now the remaining half of the data set was split into two equal sized parts, T and T_{val}. The random forest was then trained on T as before using bagging, but the empirical Gibbs loss and disagreement were now measured on the sets $\bar{T}_1, \bar{T}_2, \ldots$ combined with T_{val}. We also considered the setting in which only T_{val} was used for computing the bounds. This, as expected, led to slightly worse bounds. The results can be found in Table 4 in the appendix. As in the previous setting, we had to take care when applying the bounds. Again, the sample sizes n for the theorems were calculated as described in Section 3, but now with extra $N/4$ points available. As before, we applied the PBkl-bound, the $C1$-bound and the $C2$-bound,

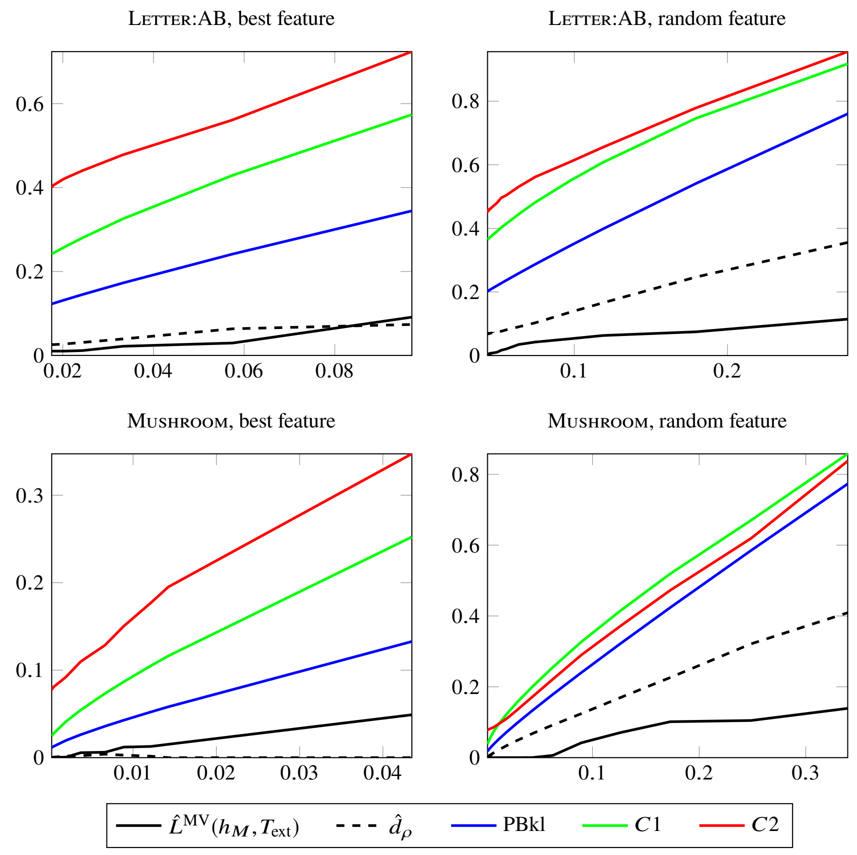


Fig. 2 Evolution of bounds depending on voter strength (measured by Gibbs risk) on data sets LETTER:AB (top) and MUSHROOM (bottom) in the bagging setting, using the best feature for splits (left) or a random feature (right). In addition, the empirical disagreement $\hat{d}_\rho$ between the trees is plotted.

but now with the addition of the SH-bound. Having the separate validation set allowed us to apply this single hypothesis bound, which is based only on T_{val}.

Table 2 (right) lists the results. Again, the loss of h_M on T_{ext} is given as an estimate of the accuracy of the classifier. As before, we see that the PBkl-bound was tighter than the C-bounds in almost all cases, and again the explanation lies in the strength of the individual classifiers. We also see that the $C2$-bound was tighter than $C1$-bound. This is in accordance with the observation by Germain et al. (2015) that the $C2$-bound is often tighter when there is an equal amount of data available for estimating the empirical Gibbs loss and the empirical correlation between any two classifiers. However, we see in all cases that the single hypothesis bound gives the best guarantees. This indicates that the PBkl-bound does indeed suffer from not taking correlations into account, even if it outperforms the C-bounds.

Comparing the results to the bounds obtained in the previous experiment, we see that, with the exception of the SH-bound, the bounds overall were very similar, some bounds better, some worse. This can be explained by the trade-off between using data for training the classifier and using data for evaluating the classifier as part of computing the bounds. In the previous experiment, more data was used to train the random forest, which typically gives a better classifier (as also indicated by the performance on the test set T_{ext}), resulting in a lower empirical Gibbs loss. Still, in this experiment the bound can be tighter because more data is used to evaluate the classifiers. This is demonstrated in Figure 6 in Appendix B which shows a comparison of the two settings on two exemplary data sets, LETTER:DO and ADULT. The figure illustrates the difference in tightness of the bounds.

The SH-bound provides the best guarantees for all data sets across both experiments, indicating that the other bounds are still too loose. The SH-bound does not come for free though, as data must be set aside, whereas the bounds computed in the bagging setting often provide useful guarantees and a better classifier.

The dependence of the bounds on the strengths of the individual voters is shown in Figure 3 for the data sets LETTER:AB and MUSHROOM. As in the previous setting, maximum tree depth and feature selection at splits (using the best or a random feature) were used to control voter strength.

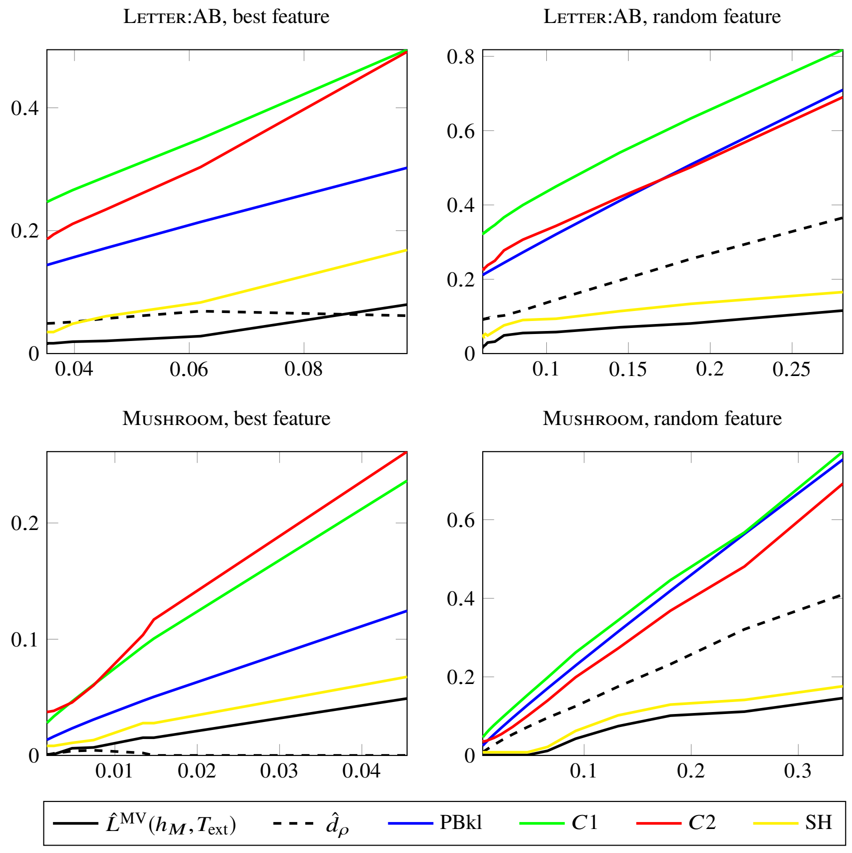


Fig. 3 Evolution of bounds depending on voter strength (measured by Gibbs risk) on data sets LETTER:AB (top) and MUSHROOM (bottom) in the setting with a separate validation set, using the best feature for splits (left) or a random feature (right). In addition, the empirical disagreement $\hat{d}_\rho$ between the trees is plotted.

Even when individual voters got weaker, the SH-bound remained tighter. As expected, the $C2$-bound now outperformed the PBkl-bound when individual voters are weak and

Table 3 Loss on T_{ext} obtained when ρ is chosen by minimizations of the λ-bound ($\rho = \rho_\lambda$) and the $C3$-bound ($\rho = \rho_C$), compared to loss obtained with $\rho = u$.

Data set	u	ρ_λ	ρ_C
Adult	0.152	0.170	0.152
Mushroom	0.000	0.000	0.000
Letter:AB	0.010	0.010	0.010
Letter:DO	0.044	0.051	0.044
Letter:OQ	0.051	0.061	0.051
Tic-Tac-Toe	0.079	0.069	0.086
Credit-A	0.144	0.153	0.144

disagreement is high. However, to observe this effect it was necessary to consider a single random feature for splitting. Considering the best feature with shallow trees also results in weak voters, but because of lower disagreement (due to trees being very similar), the C-bounds are still loose.

Comparing to the bagging setting, we see the impact of having extra data from the left out validation set, T_{val}, when evaluating the bounds. Still, the PBkl-bound remained tighter for strong voters, that is, when the Gibbs risk is close to zero.

4.3 Random Forest with Optimized Posterior.

Finally, we optimized the posteriors based on the λ-bound (Theorem 5) and the $C3$-bound (Theorem 6). The former was updated by iterative application of the update rules given by Thiemann et al. (2017). For the latter, we made sure that the hypothesis set is self-complemented (Germain et al., 2015) by adding a copy of all trained trees with their predictions reversed. The quadratic program was then solved using the solver CVXOPT (Dahl and Vandenberghe, 2007).

For each experiment, we split the data set into a training set T and an external test set T_{ext} *not used in the model building process*, see Figure 1. We only considered larger benchmark data sets, because T and T_{ext} needed to be of sufficient size. The random forest was then trained using bagging, and the posteriors were then optimized using the individual sets $\bar{T}_1, \bar{T}_2, \ldots$. We selected the hyperparameter μ for the quadratic program of Theorem 6 that minimized the OOB estimate. Once the optimal ρ was found, the random forest with optimized weights was evaluated on T_{ext}. A random forest with uniform posterior was trained and evaluated in the same setting as a baseline.

Table 3 lists the loss on T_{ext} for the seven largest data sets when optimizing ρ by minimzation of the λ-bound and $C3$-bound. ρ_λ and ρ_C denotes the optimal posteriors found using the optimization with the λ-bound and the $C3$-bound respectively. Note that for ρ_C, the hypothesis set is modified such that it is self-complemented.

For the optimization using the λ-bound, we see that, except on the Tic-Tac-Toe data set, the test loss for the optimized posterior was equal or slightly higher. The reason is that, because the λ-bound does not consider interactions between ensemble members, it tends to put most weight on only a few trees. Thus, the effect of cancellation of errors vanishes.

Figure 4 demonstrates that indeed most of the probability mass was centered on the few trees. However, recomputing the PBkl-bound, $C1$-bound and $C2$-bound using posterior ρ_λ,

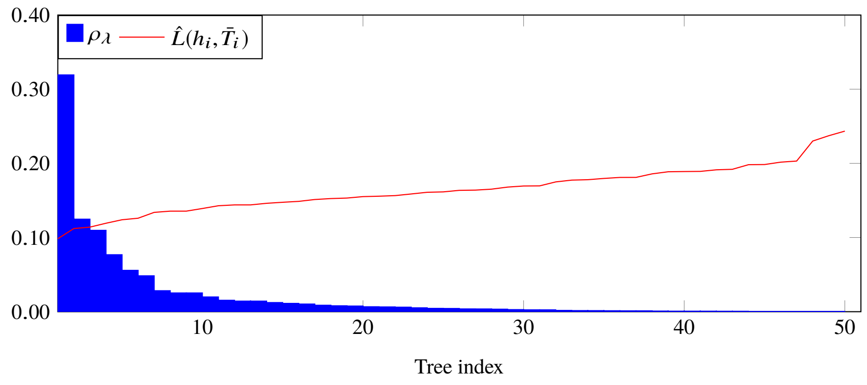


Fig. 4 Optimized distribution ρ_λ and for each tree i the error on the subset $\bar{T}_i$ of the training data not used for building the tree. Shown are the results for the CREDIT-A data set using 50 trees.

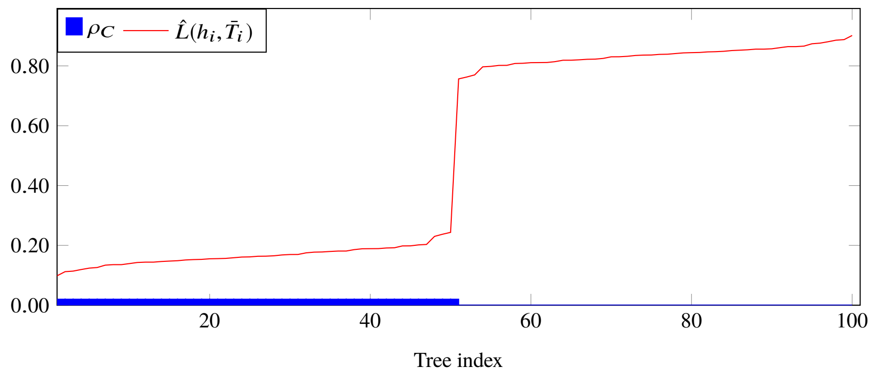


Fig. 5 Optimized distribution ρ_C and for each tree i the error on the subset $\bar{T}_i$ of the training data not used for building the tree. Shown are the results for the CREDIT-A data set using 100 self-complemented trees.

we observed that the PBkl-bound (and actually also the C-bounds) became tighter, indicating that the bounds are still quite loose.

Optimizing using the C-bound in Theorem 6 does not suffer from the probability mass being concentrated on very few tress, because of the restriction to posteriors aligned on the prior (which is uniform in our case) and the fact that the individual trees were rather strong. The probability mass can only be moved between a tree and its complement. If h_i has a small loss, $\rho_C(h_i)$ is close to $1/m$, since $-h_i$ is very weak. Figure 5 shows an example. The algorithm selected almost exclusively the strong classifiers, and due to the required alignment, the ρ_C was basically the uniform distribution on the original (non-self-complemented) hypothesis set, explaining the similarities in accuracy and bounds.

5 Conclusions

PAC-Bayesian generalization bounds can be used to obtain rigorous performance guarantees for the standard random forest classifier used in practice. No modification of the algorithm is necessary and no additional data is required because the out-of-bag samples can be exploited. In our experiments using the standard random forest, bounds inherited from the corresponding Gibbs classifiers clearly outperformed majority vote bounds that take correlations between ensemble members into account. The reason is that the individual decision trees are already rather accurate classifiers, which makes it difficult to estimate the correlations of errors. As expected, we could observe the opposite result when using weaker individual classifiers. However, this required enough disagreement between the classifiers (which we enforced by increasing randomization) and using a separate validation set, because the out-of-bag samples alone provided not enough data for reliably estimating the correlation between two voters. We also replaced the majority vote by a weighted majority vote and optimized the weights by minimizing the PAC-Bayesian bounds. This led to better performance guarantees, but weaker empirical performance.

When we split the data available at training time into a training and a validation set, we can use the hold-out validation set to compute a generalization bound. In our experiments, this led to considerably tighter bounds compared to the PAC-Bayesian approaches. However, because less data was available for training, the resulting classifiers performed worse on an external test set in most cases. Thus, using a validation set gave us better performance guarantees, but worse performance.

Our conclusion is that existing results that are derived for ensemble methods and take correlations of predictions into account are not sufficiently strong for guiding model selection and/or weighting of ensemble members in majority voting of powerful classifiers, such as decision trees. While the C-bounds are empirically outperformed by the generalization bounds based on the Gibbs classifier, the latter ignore the effect of cancellation of errors in majority voting and, thus, are of limited use for optimizing a weighting of the ensemble members and guiding model selection. Therefore, more work is required for tightening the analysis of the effect of correlations in majority voting. Nevertheless, to our knowledge, the PAC-Bayesian approach in this study provides the tightest upper bounds for the performance of the canonical random forest algorithm without requiring hold-out data.

Acknowledgements

We acknowledge support by the Innovation Fund Denmark through the *Danish Center for Big Data Analytics Driven Innovation* (DABAI).

References

Arlot S, Genuer R (2014) Analysis of purely random forests bias. ArXiv e-prints `1407.3939`
Biau G (2012) Analysis of a random forests model. Journal of Machine Learning Research 13(1):1063–1095
Biau G, Devroye L, Lugosi G (2008) Consistency of random forests and other averaging classifiers. Journal of Machine Learning Research 9:2015–2033
Breiman L (1996a) Bagging Predictors. Machine Learning 24(2):123–140
Breiman L (1996b) Out-of-bag estimation URL `https://www.stat.berkeley.edu/users/breiman/OOBestimation.pdf`
Breiman L (2001) Random Forests. Machine Learning 45(1):5–32
Breiman L (2002) Some infinity theory for predictor ensembles. Journal of Combinatorial Theory A 98:175–191
Dahl J, Vandenberghe L (2007) CVXOPT. Http://mloss.org/software/view/34/

Denil M, Matheson D, Freitas ND (2014) Narrowing the gap: Random forests in theory and in practice. In: Proceedings of the 31st International Conference on Machine Learning (ICML), PMLR, Proceedings of Machine Learning Research, vol 32, pp 665–673

Fernández-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research 15:3133–3181

Genuer R (2010) Risk bounds for purely uniformly random forests. Tech. rep., Institut National de Recherche en Informatique et en Automatique, France

Germain P, Lacasse A, Laviolette F, Marchand M, Roy JF (2015) Risk bounds for the majority vote: From a PAC-bayesian analysis to a learning algorithm. Journal of Machine Learning Research 16:787–860

Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. Machine Learning 63(1):3–42

Gieseke F, Igel C (2018) Training big random forests with little resources. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM Press, pp 1445–1454

Hastie T, Tibshirani R, Friedman J (2009) The Elements of Statistical Learning, 2nd edn. Springer

Langford J, Shawe-Taylor J (2002) PAC-Bayes & Margins. In: Proceedings of the 15th International Conference on Neural Information Processing Systems, MIT Press, pp 439–446

Maurer A (2004) A note on the PAC-Bayesian theorem. www.arxiv.org

Mcallester D (2003) Simplified PAC-Bayesian margin bounds. In: Proceedings of the 16th Annual Conference on Computational Learning Theory (COLT), Springer, LNCS, vol 2777, pp 203–215

McAllester DA (1998) Some PAC-Bayesian theorems. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT), ACM, pp 230–234

McAllester DA (1999) PAC-Bayesian Model Averaging. In: Proceedings of the Twelfth Annual Conference on Computational Learning Theory (COLT), ACM, pp 164–170

Oneto L, Cipollini F, Ridella S, Anguita D (2018) Randomized learning: Generalization performance of old and new theoretically grounded algorithms. Neurocomputing 298:21 – 33

Schapire RE, Singer Y (1999) Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning 37(3):297–336

Seeger M (2002) PAC-Bayesian generalization error bounds for Gaussian process classification. Journal of Machine Learning Research 3:233–269

Thiemann N, Igel C, Wintenberger O, Seldin Y (2017) A strongly quasiconvex PAC-Bayesian bound. In: Proceedings of the International Conference on Algorithmic Learning Theory (ALT), PMLR, Proceedings of Machine Learning Research, vol 76, pp 466–492

Valiant LG (1984) A Theory of the Learnable. Communications of the ACM 27(11):1134–1142

Wang Y, Tang Q, Xia ST, Wu J, Zhu X (2016) Bernoulli Random Forests: Closing the Gap between Theoretical Consistency and Empirical Soundness. In: Proceedings of the 17th International Conference on Machine Learning (ICML), Morgan Kaufmann, pp 2167–2173

A Extra Results for Second Setting

Table 4 lists the bounds and losses obtained in the validation set setting using only T_{val} for computing the bounds.

Table 4 The PBkl-bound, $C1$-bound, $C2$-bound and SH-bound computed for the binary UCI data sets in the validation set setting, where only T_{val} is used for computing the bounds. The majority vote loss on T_{ext} is given as an estimate of the accuracy of the trained classifier denoted as *test score*. The best bound is marked with **bold**, while *italics* is used to indicate trivial bounds (≥ 0.5).

Data set	n	d	Test score	PBkl	$C1$	$C2$	SH
Adult	45222	14	0.154	0.432	*0.510*	0.483	**0.169**
Credit-A	653	15	0.135	*0.632*	*0.854*	*0.812*	**0.294**
Haberman	306	3	0.333	*>1*	*>1*	*>1*	***0.577***
Heart	297	13	0.282	*0.893*	*0.992*	*0.990*	**0.412**
ILPD	579	10	0.307	*>1*	*>1*	*>1*	**0.441**
Ionosphere	351	34	0.125	*0.728*	*0.931*	*0.910*	**0.299**
Letter:AB	20000	16	0.015	0.152	0.266	0.192	**0.035**
Letter:DO	20000	16	0.067	0.237	0.383	0.300	**0.072**
Letter:OQ	20000	16	0.059	0.352	*0.519*	0.418	**0.119**
Mushroom	8124	22	0.001	0.017	0.036	0.041	**0.008**
Sonar	208	60	0.250	*>1*	*>1*	*>1*	***0.510***
Tic-Tac-Toe	958	9	0.142	*0.765*	*0.908*	*0.807*	**0.221**
USvotes	232	16	0.052	*0.513*	*0.784*	*0.739*	**0.228**
WDBC	569	30	0.063	0.342	*0.567*	0.490	**0.102**

B Comparison Plots for the Bagging and Validation Set Settings

Figure 6 shows the comparison of the bounds obtained for the Letter:DO and Adult data set. The figure includes all three settings: using only the hold-out sets from bagging ($\bar{T}$), using only the validation set (T_{val}), and using a combination of both ($\bar{T}$+T_{val})

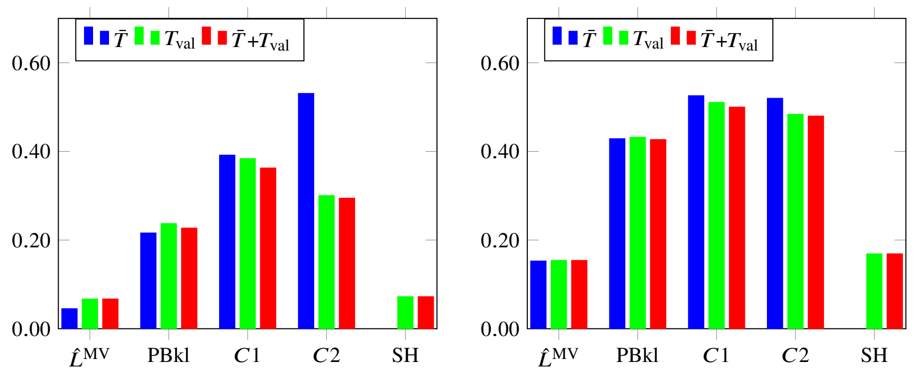


Fig. 6 Comparison of the bounds obtained for a random forest with 500 trees trained on the Letter:DO data set. Comparison of the bounds obtained for a random forest with 500 trees trained on the Letter:DO data set (left) and for a random forest with 1000 trees trained on the Adult data set (right).

A.7. Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search

Authors:	Stephan Sloth Lorenzen Ninh Pham (University of Auckland)
Submitted to:	23'rd International Conference on Extending Database Technology (EDBT)
Year:	2019
arXiv:	`https://arxiv.org/abs/1908.08656`

Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search

Ninh Pham
School of Computer Science
University of Auckland
Auckland, New Zealand
ninh.pham@auckland.ac.nz

Stephan Lorenzen
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
lorenzen@di.ku.dk

ABSTRACT

Top-k maximum inner product search (MIPS) is a central task in many machine learning applications. This paper extends top-k MIPS with a budgeted setting, that asks for the *best* approximate top-k MIPS given a limit of $\mathcal{B}$ computational operations. We investigate recent advanced sampling algorithms, including wedge and diamond sampling to solve it. Though the design of these sampling schemes naturally supports budgeted top-k MIPS, they suffer from the linear cost from scanning *all* data points to retrieve top-k results and the performance degradation for handling negative inputs.

This paper makes two main contributions. First, we show that diamond sampling is essentially a combination between wedge sampling and basic sampling for top-k MIPS. Our theoretical analysis and empirical evaluation show that wedge is competitive (often superior) to diamond on approximating top-k MIPS regarding both efficiency and accuracy.

Second, we propose a series of algorithmic engineering techniques to deploy wedge sampling on budgeted top-k MIPS. Our novel *deterministic* wedge-based algorithm runs significantly faster than the state-of-the-art methods for budgeted and exact top-k MIPS while maintaining the top-5 precision at least 80% on standard recommender system data sets.

1 INTRODUCTION

Maximum inner product search (MIPS) is the task of, given a point set $\mathbb{X} \subset \mathbb{R}^d$ of size n and a query point $\mathbf{q} \in \mathbb{R}^d$, finding the point $\mathbf{p} \in \mathbb{X}$ such that,

$$\mathbf{p} = \arg\max_{\mathbf{x}\in\mathbb{X}} \mathbf{x} \cdot \mathbf{q} \ .$$

MIPS and its variant top-k MIPS, which finds the top-k largest inner product points with a query, are central tasks in the retrieval phase of standard collaborative filtering based recommender systems [7, 16]. They are also algorithmic ingredients in a variety of machine learning tasks, for instance prediction tasks on multi-class learning [8, 25] and neural network [6, 28], and as a black-box procedure to speed up learning and inference algorithms [21].

Modern real world online recommender systems, e.g. Xbox or Netflix often deal with very large-scale data sets and limited amount of response time [2, 4]. Such collaborative filtering based systems often present users and items as low-dimensional vectors. A large inner product between these vectors indicates that the item is relevant to the user preferences and should be in the recommendation list to the user. The recommendation is often performed in the *online* manner since the user vector is updated online with ad-hoc contextual information only available during the interaction [2, 15, 16]. For instance, new users in a recommendation system often want to see the list of recommended items immediately when they already input their preferences. A personalized recommender needs to infer user preferences based on online user behavior, e.g. recent search queries and browsing history, as implicit feedback in order to return relevant results [13, 24]. Since the retrieval of recommended items is only performed online, the result of this task might not be "perfect" given a small amount of waiting time but its accuracy/relevance should be improved given more waiting time. Hence, it is challenging to not only speed up the MIPS process, but to trade the search efficiency for the search quality for improving performance of recommender systems.

Motivated by the computational bottleneck in the retrieval phase of modern recommendation systems, this work investigates the *budgeted* MIPS problem, a natural extension of MIPS with a computational limit for the search efficiency and quality trade-off. Given that $d = O(\log n)$, our budgeted MIPS addresses the following question:

Given a data structure built in $\tilde{O}(n)$ time[1] *and a budget $\mathcal{B}$ computational operations, can we have an algorithm to return the best approximate top-k MIPS?*

In our budgeted setting, we limit the time complexity of building a data structure to $\tilde{O}(n)$ since when a context is used in a recommender system, the learning phase cannot be done entirely offline [2, 15]. In other words, the items vectors are also computed online and hence a high cost of constructing the data structure will degrade the performance. Furthermore, since user preferences often change over time, a recommender system needs to frequently update its factorization model to address such *drifting* user preferences. This means that the items vectors and our data structure will be updated frequently.

It is worth noting that the budgeted MIPS has been recently studied in [31] given a budget of $\mathcal{B} = \eta n$ inner product computations where η is a small constant, e.g. 5%. Furthermore, such budget constraints on the number of computational operations or on accessing a limit number of data points are widely studied not only on search problems [18, 23] but also on clustering [19, 26] and other problems [11, 32] when dealing with large-scale complex data sets.

1.1 Prior art on solving MIPS and its limit on budgeted MIPS

It is well-known that due to the "curse of dimensionality", any exact solution for MIPS based on data or space partitioning indexing data structures generally degrades when dimensionality increases. It is no better than a simple sequential scanning when dimensionality is larger than 10 [17, 30]. Hence recent work on solving MIPS focuses on speeding up sequential scanning by

[1]Polylogarithmic factors, e.g. $\log n$ is absorbed in the $\tilde{O}$-notation.

pruning the search space [17, 29]. Even though such methods can solve MIPS exactly, it requires $\Theta(n)$ operations. Hence it does not fit well to the budgeted MIPS setting since we might need $o(n)$ operations for some query. Furthermore, these methods do not provide any trade-off between the search quality and efficiency for online queries.

Another research direction is investigating approximation solutions which trade accuracy for efficiency. Since *locality-sensitive hashing* (LSH) [1, 12] has emerged as a basic algorithmic tool for similarity search in high dimensions due to the sublinear query time guarantee, several approaches have followed this direction to obtain sublinear solutions for approximate MIPS [14, 22, 27]. Due to the inner product not being a metric, these LSH-based solutions have to convert MIPS to near neighbor search problem by applying order preserving transformations, in order to exploit the LSH framework.

Although LSH-based approaches can guarantee sublinear query time, the top-k inner product values are often very small compared to the vector norms in high dimensions. This means that the distance gap between "close" and "far apart" points in the LSH framework is arbitrary small. That leads to not only the space usage (i.e. number of hash tables) blow up, but also degrading LSH performance. Furthermore, the LSH trade-off between search quality and search efficiency is somewhat "fixed" for any query since it is governed by specific parameters of the LSH data structure, e.g. number of hash tables. More important, the learning phase of a recommender system has to be executed in the online manner when a context is used [2, 15]. Hence, the subquadratic cost of building LSH tables will be a bottleneck for handling online recommendations.

An alternative efficient solution is applying sampling methods to estimate the vector-matrix multiplication derived by top-k MIPS [3, 5]. The basic idea is to sample a point $\mathbf{x}$ with probability proportional to the inner product $\mathbf{x} \cdot \mathbf{q}$. The larger inner product values the point $\mathbf{x}$ has, the more occurrences of $\mathbf{x}$ in the sample set. By the end of the sampling process, we retrieve top-m budgeted points ($m > k$) with largest occurrences in the sample set via a counting histogram. The top-k points, with the largest inner product with the query among these m points, will be returned as an approximation for top-k MIPS. It is clear that sampling schemes naturally fit to the budgeted setting since the more samples we use, the higher accuracy we can achieve. However, the linear cost of scanning *all* data points to return top-m candidate points limits sampling methods to $o(n)$ budget.

1.2 Our contribution

This work investigates sampling methods for solving the budgeted MIPS since these methods naturally fit to the class of budgeted problems. Sampling schemes provide not only the trade-off between search quality and search efficiency but also a flexible mechanism to control this trade-off via the number of samples. Our contributions are as follows:

(1) We revise popular sampling methods for MIPS, including basic sampling, wedge sampling [5], and the state-of-the-art diamond sampling methods [3]. We show that diamond sampling is essentially a combination between basic sampling and wedge sampling.
(2) Our novel theoretical analysis and empirical evaluation illustrate that wedge is competitive (often superior) to diamond on approximating top-k MIPS regarding both efficiency and accuracy.
(3) In order to deploy wedge sampling on budgeted top-k MIPS, we propose a series of algorithmic engineering techniques, including (1) a simple shifting technique which transforms a MIPS with general inputs to a non-negative one while preserving the inner products order; (2) a novel greedy sampling generator which carefully selects representative *modes* of a discrete distribution, leading to a *deterministic* version of wedge sampling; and (3) a fast wedge-based algorithm running in $O(\mathcal{B})$ time, which completely governs the trade-off between search quality and efficiency in the budgeted setting.
(4) Our empirical results confirm the efficiency of our proposed algorithm on standard recommender system data sets. In particular, our *deterministic* wedge-based method returns top-5 MIPS with the accuracy at least 80%, and runs significantly faster than the state-of-the-art methods for budgeted [31] and exact top-k MIPS [17].

2 NOTATION AND PRELIMINARIES

We use lower-case fonts for scalars, upper-case fonts for random variables, bold lower-case fonts for vectors, and bold upper-case fonts for matrices. For convenience, we present the point set $\mathbb{X}$ as a matrix $\mathbf{X} \subset \mathbb{R}^{n \times d}$ where each point $\mathbf{x}_i$ corresponds to the ith row, and the query point $\mathbf{q}$ as a column vector $\mathbf{q} = (q_1, \ldots, q_d)^T$. We use $i \in [n]$ to index row vectors of $\mathbf{X}$, i.e., $\mathbf{x}_i = (x_{i1}, \ldots, x_{id}) \in \mathbb{R}^d$. Since we will describe our investigated methods using the column-wise matrix-vector multiplication $\mathbf{Xq}$, we use $j \in [d]$ to index column vectors of $\mathbf{X}$, i.e. $\mathbf{y}_j = (x_{1j}, \ldots, x_{nj})^T \in \mathbb{R}^n$. For each column j, we define $c_j = \sum_{i=1}^n x_{ij}$. We also define the minimum and maximum values of $\mathbf{y}_j$ as $\alpha_j = \min(\mathbf{y}_j) = \min(x_{1j}, x_{2j}, \ldots, x_{nj})$ and $\beta_j = \max(\mathbf{y}_j) = \max(x_{1j}, x_{2j}, \ldots, x_{nj})$, respectively.

We briefly review sampling approaches for estimating inner products $\mathbf{x}_i \cdot \mathbf{q}$ and provide corresponding algorithms for MIPS and its potential extension for budgeted MIPS. For simplicity, we first assume that $\mathbf{X}$ and $\mathbf{q}$ are non-negative. Then we show how to extend these approaches to handle negative inputs with their limits. We will present sampling algorithms based on the column-wise matrix-vector multiplication $\mathbf{Xq}$, i.e. sum of d rank-one matrices, as follows.

$$\begin{aligned} \mathbf{Xq} &= \begin{bmatrix} x_{11} \\ \vdots \\ x_{n1} \end{bmatrix} q_1 + \begin{bmatrix} x_{12} \\ \vdots \\ x_{n2} \end{bmatrix} q_2 + \ldots + \begin{bmatrix} x_{1d} \\ \vdots \\ x_{nd} \end{bmatrix} q_d \\ &= \mathbf{y}_1 q_1 + \mathbf{y}_2 q_2 + \ldots + \mathbf{y}_d q_d \end{aligned} \quad (1)$$

2.1 Basic Sampling

Basic sampling is a very straightforward method to estimate the inner product $\mathbf{x}_i \cdot \mathbf{q}$ for the point $\mathbf{x}_i$. For any row i, we sample a column j with probability $q_j / \|\mathbf{q}\|_1$ and return x_{ij}. Define a random variable $Z_i = x_{ij}$, we have

$$\mathbf{E}[Z_i] = \sum_{j=1}^{d} x_{ij} \frac{q_j}{\|\mathbf{q}\|_1} = \frac{\mathbf{x}_i \cdot \mathbf{q}}{\|\mathbf{q}\|_1}$$

The basic sampling suffers large variance when most of the contribution of $\mathbf{x}_i \cdot \mathbf{q}$ are from a few of coordinates. In particular, the variance will be significantly large when the main contributions of $\mathbf{x}_i \cdot \mathbf{q}$ are from a few coordinates $x_{ij} q_j$ and q_j are very small. Note that this basic sampling approach has been used in [10] as an efficient sampling technique for approximating matrix-matrix multiplication.

Negative cases: To handle the negative cases, one can change the sampling probability to $|q_j|/\|\mathbf{q}\|_1$ and return $Z_i = \mathrm{sgn}(q_j)x_{ij}$ where sgn is the sign function, i.e. $\mathrm{sgn}(u) = -1$ if $u < 0$ and $\mathrm{sgn}(u) = 1$ if $u \geq 0$. It is clear that $\mathbf{E}[Z_i] = \mathbf{x}_i \cdot \mathbf{q}/\|\mathbf{q}\|_1$. This scheme also incurs large variance when the contributions of inner product are skewed.

Given that a sample requires a constant cost, the basic sampling needs $s = \Omega(n)$ samples for estimating n inner products and an additional $O(n \log k)$ cost to return top-k MIPS. Hence it is not suitable for budgeted MIPS settings with $o(n)$ required computational resources.

2.2 Wedge Sampling

Cohen and Lewis [5] proposed an efficient sampling approach, called wedge sampling, to approximate matrix multiplication and to isolate the largest inner products as a byproduct. Wedge sampling needs to pre-compute some statistics, including the sum of all inner products, i.e. $z = \sum_i z_i$ where $z_i = \mathbf{x}_i \cdot \mathbf{q}$ and norm-1 of columns $c_j = \|\mathbf{y}_j\|_1$. Since we can precompute c_j in advance, computing $z = \sum_j c_j q_j$ clearly takes $O(d)$ time.

The basic idea of wedge sampling is to randomly sample a row index i corresponding to $\mathbf{x}_i$ with probability z_i/z. Hence, the larger the inner product $z_i = \mathbf{x}_i \cdot \mathbf{q}$, the larger the number of occurrences of i in the sample set. Consider Equation (1), wedge sampling first samples a column j corresponding to $\mathbf{y}_j$ with probability $q_j c_j/z$, and then samples a row i corresponding to $\mathbf{x}_i$ from $\mathbf{y}_j$ with probability x_{ij}/c_j. By Bayes's theorem,

$$\begin{aligned}&\mathbf{Pr}[\texttt{Sampling } i] = \\ &\sum_{j=1}^{d} \mathbf{Pr}[\texttt{Sampling } i | \texttt{Sampling } j] \cdot \mathbf{Pr}[\texttt{Sampling } j] \\ &= \sum_{j=1}^{d} \frac{x_{ij}}{c_j} \cdot \frac{q_j c_j}{z} = \frac{\sum_{j=1}^{d} x_{ij} q_j}{z} = \frac{z_i}{z} \ .\end{aligned}$$

Applying wedge sampling method on $\mathbf{X}\mathbf{q}$, we obtain a sample set where each index i corresponding to $\mathbf{x}_i$ is sampled according to an independent Bernoulli distribution with parameter $p_i = z_i/z$. Next, a counting algorithm will be used to find the points with largest counters. Given s samples and a constant cost for each sample, such counting algorithm runs in $O(s + n \log k)$ time to answer approximate top-k MIPS. Note that we can compute the exact inner products for the $m > k$ points with largest counters for post-processing. The post-processing phase with an additional $O(dm + m \log k)$ computational cost will provide higher accuracy for top-k MIPS in practice.

We note that since wedge sampling uses the term $q_j c_j$ when sampling the column j, it alleviates the effect of skewness of $\mathbf{x}_i \cdot \mathbf{q}$ where large contributions are from a few coordinates. Hence wedge sampling achieves lower variance than the basic sampling in practice. At the query phase, wedge sampling with careful implementation [5] needs only simple counting and *sequential* memory access operations, which are always faster than expensive floating-point multiply-add operations required by any brute-force method.

Negative cases: Again, we can use the sign trick to deal with negative cases. We note that this trick has been also exploited in the diamond sampling approach [3]. In particular, we execute wedge sampling on absolute values of $\mathbf{X}$ and $\mathbf{q}$, and return $Z_i = \mathrm{sgn}(x_{ij})\mathrm{sgn}(q_j)$ for the point $\mathbf{x}_i$. It is clear that $\mathbf{E}[Z_i]$ is proportional to $\mathbf{x}_i \cdot \mathbf{q}$. Hence we can leverage an efficient implementation of wedge sampling [5] to answer top-k MIPS with negative inputs.

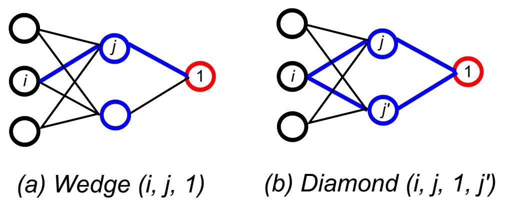


Figure 1: Given that $n = 3, d = 2$, Xq is presented as a weighted tripartite graph above. Diamond sampling randomly picks a wedge $(i, j, 1)$ by first choosing column j with probability $q_j c_j/z$, then choosing x_{ij} for $\mathbf{x}_i$ with probability x_{ij}/c_j. After that, it randomly chooses column j' with probability $q'_j/\|\mathbf{q}\|_1$ to form a diamond $(i, j, 1, j')$.

2.3 Diamond Sampling

Ballard et al. [3] proposed diamond sampling to find the largest *magnitude* elements from a matrix-matrix multiplication $\mathbf{XQ}$ without computing directly the final matrix. The method presents $\mathbf{XQ}$ as a weighted tripartite graph. The number of nodes on the left (right) present the number of rows (columns) of $\mathbf{X}$ ($\mathbf{Q}$) while the number of nodes in the middle correspond to the number of columns (rows) of $\mathbf{X}$ ($\mathbf{Q}$). Then it samples a diamond, i.e. four cycles from such graph with probability proportional to the value $(\mathbf{XQ})^2_{ij}$, which claims to amplify the focus on the largest magnitude elements.

Consider a vector $\mathbf{q}$ as an one-column matrix $\mathbf{Q}$, it is clear that diamond sampling can be applied to solve MIPS. Indeed, for the matrix-vector multiplication $\mathbf{X}\mathbf{q}$, diamond sampling is essentially a combination between wedge sampling and basic sampling, as shown in Figure 1. In particular, diamond sampling first makes use of wedge sampling to return a random row i corresponding to $\mathbf{x}_i$ with probability z_i/z. Given such row i, it then applies basic sampling to sample a random column j' with probability $q'_j/\|\mathbf{q}\|_1$ and return $x_{ij'}$ as a scaled estimate of $(\mathbf{x}_i \cdot \mathbf{q})^2$. Define a random variable $Z_i = x_{ij'}$ corresponding to $\mathbf{x}_i$, using the properties of wedge sampling and basic sampling we have

$$\mathbf{E}[Z_i] = \sum_{j'=1}^{d} x_{ij'} \frac{q_{j'}}{\|\mathbf{q}\|_1} \cdot \frac{z_i}{z} = \frac{(\mathbf{x}_i \cdot \mathbf{q})^2}{z\|\mathbf{q}\|_1}$$

Since diamond sampling builds on basic sampling, it suffers from the same drawback as basic sampling. To answer top-k MIPS, diamond sampling follows the same procedure as wedge sampling; hence it shares the same asymptotic running time, $O(s + n \log k)$.

Negative cases: Handling negative cases using diamond is similar to wedge. We apply diamond sampling on absolute values of $\mathbf{X}$ and $\mathbf{q}$ then return $Z_i = \mathrm{sgn}(q_j)\mathrm{sgn}(x_{ij})\mathrm{sgn}(q_{j'})x_{ij'}$ where j is the column sampled by wedge sampling first and j' is the column sampled by basic sampling later. We can verify that $\mathbf{E}[Z_i]$ is proportional to $(\mathbf{x}_i \cdot \mathbf{q})^2$.

Although diamond sampling can deal with negative inputs, its concentration bound only works on non-negative cases. This is due to the complex analysis of dependent sign random variables. Furthermore, the implementation of diamond sampling [3] requires significant time overhead due to the sampling process from a discrete distribution derived from the query $\mathbf{q}$ and *random* access operations to access $x_{ij'}$.

2.4 Sampling Methods for Budgeted Top-k MIPS

It is clear that the design of sampling algorithms naturally supports budgeted MIPS. Since the more samples used the less variance of the estimate, these methods are able to govern the trade-off between search efficiency and search quality.

Note that in practice advanced sampling algorithms with the post-processing step often achieves higher search quality [3, 5, 31]. Hence, given a budget $\mathcal{B}$ of computational operations, we distribute it into the two procedures, including candidate screening and candidate ranking. The candidate screening will return top-m candidate point indexes with largest estimate values. This step requires $O\left(s + n\log m\right)$ time where s is the total sampling cost. Then a post-processing of $O\left(dm + m\log k\right)$ time manipulates the candidate ranking procedure by computing exactly these m inner products and returning top-k MIPS.

Drawbacks of advanced sampling: In theory, the time complexity of the candidate screening $O\left(s + n\log m\right)$ limits the use of these sampling methods on budgeted MIPS with $\mathcal{B} = o(n)$ operations. In practice, these sampling schemes are not suitable to solve MIPS with general inputs. Diamond sampling indeed solves a different problem, i.e. $\arg\max_i \left(\mathbf{x}_i \cdot \mathbf{q}\right)^2$, which will give a completely different result on negative inputs. Although wedge sampling can handle general inputs, it is executed on absolute values of $\mathbf{X}$ and $\mathbf{q}$, similar to diamond sampling. Hence it essentially uses more samples for the largest *magnitude* dot products since it tends to sample the point $\mathbf{x}_i$ with large $\sum_j |x_{ij}q_j|$.

3 WEDGE SAMPLING FOR BUDGETED TOP-k MIPS

This section presents a series of algorithmic engineering techniques to deploy wedge sampling on budgeted top-k MIPS. We first propose a simple shifting technique to convert a MIPS with general inputs to non-negative MIPS while preserving the inner product order, which enables advanced sampling methods for various MIPS applications. Exploiting the shifting technique, we then propose a fast wedge sampling algorithm which runs in time $O\left(\mathcal{B}\right)$, completely governing the trade-off between search efficiency and quality for the budgeted setting.

Our new theoretical concentration bound shows that wedge sampling requires fewer samples than diamond sampling on top-k MIPS. Finally, we introduce a novel greedy sampling generator which carefully selects representative *modes* of a discrete distribution, leading to efficient *deterministic* versions of both wedge and diamond sampling. The combination of these techniques yields a novel *deterministic* wedge-based algorithm, called *dfsWedge*, which runs significantly faster than the state-of-the-art methods for budgeted and exact top-k MIPS.

3.1 Order Preserving Transformations

This subsection will present a simple shifting technique to map a MIPS with some negative inputs to a non-negative MIPS while preserving the inner product order. This enables proposed advanced sampling methods to solve MIPS with a negligible loss of efficiency.

Considering the column-wise matrix-vector multiplication in Equation (1), our intuition is that *shifting* each vector $\mathbf{y}_j$ a constant factor preserves the order of $\mathbf{x}_i \cdot \mathbf{q}$. Hence, we can transform both $\mathbf{X}$ and $\mathbf{q}$ to non-negative representations while preserving their inner product order.

Definition 3.1. Given a point set $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and a query $\mathbf{q} \in \mathbb{R}^d$, we assume that $\mathbf{x}_1 \cdot \mathbf{q} \leq \mathbf{x}_2 \cdot \mathbf{q} \leq \ldots \leq \mathbf{x}_n \cdot \mathbf{q}$. Asymmetric non-negative transformations $f : \mathbf{x} \mapsto \mathbf{x}' \in \mathbb{R}^d_+$ and $g : \mathbf{q} \mapsto \mathbf{q}' \in \mathbb{R}^d_+$ are called *order preserving* regarding inner product if $\mathbf{x}'_1 \cdot \mathbf{q}' \leq \mathbf{x}'_2 \cdot \mathbf{q}' \leq \ldots \leq \mathbf{x}'_n \cdot \mathbf{q}'$. It follows that $\arg\max_i \mathbf{x}_i \cdot \mathbf{q} = \arg\max_i \mathbf{x}'_i \cdot \mathbf{q}'$.

Note that we restrict the new transformation space to have d dimensions since this will not affect the running time of algorithms in this space. We now describe f and g which transform $\mathbf{X}$ and $\mathbf{q}$, respectively, to non-negative representations while preserving their inner product order.

First, let us consider the case where $\mathbf{X}$ has some negative values. We compute the minimum value for each column j of $\mathbf{X}$, i.e. $\alpha_j = \min(\mathbf{y}_j)$ and consider the mapping

$$\begin{aligned}\mathbf{y}_j \mapsto \mathbf{y}'_j &= \mathbf{y}_j - \alpha_j \mathbf{1} \\ &= \left(x_{1j} - \alpha_j, x_{2j} - \alpha_j, \ldots, x_{nj} - \alpha_j\right)^T \in \mathbb{R}^n_+ \ .\end{aligned}$$

According to this mapping, we have a new representation of $\mathbf{x}_i$ as follows.

$$\mathbf{x}_i \mapsto \mathbf{x}'_i = \left(x_{i1} - \alpha_1, x_{i2} - \alpha_2, \ldots, x_{id} - \alpha_d\right) \in \mathbb{R}^d_+. \qquad (2)$$

It is clear that $\mathbf{x}'_i$ is non-negative. The following lemma states that this transformation is order preserving regarding inner product.

LEMMA 3.2. *Given a query* $\mathbf{q}$*, the non-negative transformation (2) satisfies* $\arg\max_i \mathbf{x}_i \cdot \mathbf{q} = \arg\max_i \mathbf{x}'_i \cdot \mathbf{q}$ *and* $\mathbf{x}_{i_1} \cdot \mathbf{q} - \mathbf{x}_{i_2} \cdot \mathbf{q} = \mathbf{x}'_{i_1} \cdot \mathbf{q} - \mathbf{x}'_{i_2} \cdot \mathbf{q}$ *for any* $i_1, i_2 \in [n]$.

PROOF. Consider the vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d)$, it is clear that $\mathbf{x}'_i \cdot \mathbf{q} = (\mathbf{x}_i - \boldsymbol{\alpha}) \cdot \mathbf{q} = \mathbf{x}_i \cdot \mathbf{q} - \boldsymbol{\alpha} \cdot \mathbf{q}$. Since $\boldsymbol{\alpha}$ is fixed regarding any query $\mathbf{q}$, the lemma holds. □

We now examine the case where $\mathbf{q}$ has some negative values and without loss of generality, let us assume that $q_j < 0$ for some j. Since the Equation (1) can be written as $\mathbf{Xq} = \mathbf{y}_1 q_1 + \ldots + (-\mathbf{y}_j)(-q_j) + \ldots + \mathbf{y}_d q_d$, flipping the sign of both vector $\mathbf{y}_j$ and q_j preserves all inner product values.

We denote by $\beta_j = \max(\mathbf{y}_j)$, the maximum value of each column j of $\mathbf{X}$. Note that $\beta_j = -min(-\mathbf{y}_j)$ and $-\mathbf{y}_j + \beta_j \mathbf{1} \in \mathbb{R}^n_+$. Combine the sign flipping trick with the non-negative transformation (2), the following theorem holds.

THEOREM 3.3. *The non-negative transformation* $f : \mathbf{x} \mapsto \mathbf{x}'$*, such that* $x_j \mapsto x_j - \alpha_j$ *if* $q_j \geq 0$*; otherwise,* $x_j \mapsto -x_j + \beta_j$ *and the non-negative transformation* $g : \mathbf{q} \mapsto \mathbf{q}' = |\mathbf{q}|$ *are order preserving regarding inner product. Moreover,* $\mathbf{x}_{i_1} \cdot \mathbf{q} - \mathbf{x}_{i_2} \cdot \mathbf{q} = \mathbf{x}'_{i_1} \cdot \mathbf{q}' - \mathbf{x}'_{i_2} \cdot \mathbf{q}'$ *for any* $i_1, i_2 \in [n]$.

Since we do not know the sign of q_j in advance, we will keep two mapping values $x_j - \alpha_j$ and $-x_j + \beta_j$. In other words, we maintain two mapping vectors $\mathbf{y}^+_j = \mathbf{y}_j - \alpha_j \mathbf{1}$ and $\mathbf{y}^-_j = -\mathbf{y}_j + \beta_j \mathbf{1}$ corresponding to $q_j \geq 0$ and $q_j < 0$, respectively. Exploiting these order-preserving transformations, we are able to apply advanced sampling methods to solve MIPS with general inputs.

3.2 Fast Wedge-based Algorithm for Budgeted Top-k MIPS

We now describe our novel wedge-based algorithms, including query-independent and query-dependent phases, for answering top-k MIPS with a budget $\mathcal{B}$.

Algorithm 1 Pre-processing

```
Require: Matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ presenting for the data set $\mathbb{X}$, the
    maximum number of pre-samples $s_{max}$
Ensure: Vectors $\mathbf{c}^+, \mathbf{c}^- \in \mathbb{R}^d_+$ contain norm-1 of columns of the
    data in the transform space, matrices $\mathbf{X}^+, \mathbf{X}^- \in \mathbb{N}_+^{s_{max} \times d}$ as
    pre-sample sets for non-negative and negative coordinates of $\mathbf{q}$
1: for each column vector $\mathbf{y}_j$ of $\mathbf{X}$ do
2:     Compute $\alpha_j = \min(\mathbf{y}_j)$ and $\beta_j = \max(\mathbf{y}_j)$.
3:     Compute $\mathbf{y}^+_j = \mathbf{y}_j - \alpha_j \mathbf{1}$ and $\mathbf{y}^-_j = -\mathbf{y}_j + \beta_j \mathbf{1}$.
4:     Compute and store norm-1 $c^+_j = \|\mathbf{y}^+_j\|_1$ and $c^-_j = \|\mathbf{y}^-_j\|_1$
       in the dimension $j$ of $\mathbf{c}^+$ and $\mathbf{c}^-$, respectively.
5:     Generate and store $s_{max}$ random variables from the discrete distributions $\mathbf{y}^+_j$ and $\mathbf{y}^-_j$ as the column $j$ of $\mathbf{X}^+$ and
       $\mathbf{X}^-$, respectively.
6: end for
7: return $\mathbf{X}^+$ and $\mathbf{X}^-$, $\mathbf{c}^+$ and $\mathbf{c}^-$ corresponding to non-negative and negative coordinates of $\mathbf{q}$, respectively.
```

Algorithm 2 fsWedge

```
Require: Matrices $\mathbf{X}^+$ and $\mathbf{X}^-$, vectors $\mathbf{c}^+$ and $\mathbf{c}^-$, the query $\mathbf{q}$,
    and the budget $\mathcal{B}$
Ensure: Approximate top-$k$ MIPS for $\mathbf{q}$
1: Step 1: Determine the total number of samples $s$ based on the budget $\mathcal{B}$.
2: Step 2: Compute statistics values: $z = \sum_i \mathbf{x}'_i \cdot \mathbf{q}'$ and the number of samples $s_j$ for each dimension $j$ based on $s$.
3: Step 3: For each dimension $j$, randomly access $s_j$ point indexes from the column $j$ of $\mathbf{X}^+$ if $q_j \geq 0$; otherwise, $\mathbf{X}^-$, and insert them into the counting and tracking histograms.
4: Step 4: Return top-$m$ points with largest estimate values from the histograms.
5: Step 5 (post-processing): Compute these $m$ inner products and return top-$k$ points with largest inner product values.
```

In the *offline* query-independent phase, we execute order-preserving transformations to convert to non-negative MIPS, and pre-compute some statistics needed for wedge sampling. Then we pre-sample point indexes based on specific discrete distributions for speeding up the query-dependent process.

In the *online* query-dependent phase, we determine the total number of samples s based on the budget $\mathcal{B}$. Then we compute the number of samples s_j for each dimension j and access these s_j samples from the pre-sample set. For each sampled point $\mathbf{x}_i$, we insert it into additional data structures which support fast retrieval of the top-m points with the largest counter values in only $O(s)$ time. A post-processing step for candidate ranking will compute these m inner products and return top-k MIPS.

3.2.1 ***Query-independent constructions:*** We perform sequential operations by traversing column vectors $\mathbf{y}_j$ of $\mathbf{X}$, as shown in Algorithm 1. First, we execute order-preserving transformations which generate column vectors $\mathbf{y}^+_j$ and $\mathbf{y}^-_j$ corresponding to non-negative and negative values of $\mathbf{q}$, respectively. We also store their norm-1 in vectors $\mathbf{c}^+$ and $\mathbf{c}^-$ in order to compute the statistics needed for wedge sampling in $O(d)$ time.

We sample the point $\mathbf{x}_i$ based on the discrete distribution presented by $\mathbf{y}^+_j$ (and $\mathbf{y}^-_j$) and store them as the columns j of the matrix $\mathbf{X}^+$ (and $\mathbf{X}^-$) for non-negative (and negative) coordinates of $\mathbf{q}$. One of the most popular and fastest method to generate random variables from a discrete distribution $\boldsymbol{\kappa} \in \mathbb{R}^d_+$ is the so-called *alias* method [9], which requires $O(d)$ time for setup and $O(1)$ for each sample. This alias method has been used in both diamond and wedge sampling [3]. Given the maximum number of samples s_{max} for each dimension, the query-independent step takes $O(dn + ds_{max})$ time and requires $O(ds_{max})$ additional space to store the two pre-sample sets. In practice, we often set the budget $\mathcal{B} = o(n)$, hence the setting $s_{max} = n$ suffices and the space overhead and time complexity of this step is $O(dn)$.

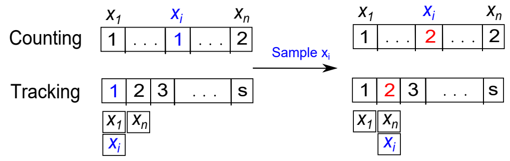


Figure 2: Updating histograms when processing $\mathbf{x}_i$. Before processing $\mathbf{x}_i$, its counter value is 1, hence $\mathbf{x}_i$ locates in the 1st cell of the tracking histogram. After processing $\mathbf{x}_i$, its counter value is updated to 2, hence $\mathbf{x}_i$ moves to the 2nd cell of the tracking histogram.

3.2.2 ***Query-dependent construction:*** Given a query with a budget $\mathcal{B}$ of computational operations, we execute Algorithm 2, called *fsWedge*, to answer budgeted top-k MIPS. Step 1 computes the total number of samples s based on the budget $\mathcal{B}$ and specific settings. In Step 2, we compute some statistics for wedge sampling. We compute the sum of all inner products in the new transformed space $z = \sum_i \mathbf{x}'_i \cdot \mathbf{q}' = \sum_j c_j |q_j|$ and number of samples $s_j = \lceil s c_j |q_j| / z \rceil$ required for each dimension j in $O(d)$ time. Note that $c_j = \|\mathbf{y}^+_j\|_1$ if $q_j \geq 0$ and $c_j = \|\mathbf{y}^-_j\|_1$ otherwise. Then, we randomly access s_j point indexes from the column j of $\mathbf{X}^+$ if $q_j \geq 0$; otherwise of $\mathbf{X}^-$, and insert them into the counting and tracking histograms (Step 3).

In Step 4, we extract the top-m points with largest estimate values using these histograms. In Step 5, we compute these m inner products, and use the standard priority queue to return more accurate top-k MIPS.

Counting and tracking histograms to return top-m points in Step 4: We now describe new data structures that enable us to retrieve top-m points with largest occurrences in $O(s)$ time, as illustrated in Figure 2. First, we need a counting histogram of size n to count up-to-date occurrences of n points. Second, we need a tracking histogram to keep track of all point indexes with the same number of occurrences. Such tracking histogram can be implemented as an array and its cell is a hash table containing all point indexes with the same counter value. When sampling $\mathbf{x}_i$, we get its old occurrences in the counting histogram in order to find it in the tracking histogram. Then, we increase the counter of $\mathbf{x}_i$ by 1 and move it into the hash table of the next cell. When we finish Step 3, we simply traverse the tracking array from end to front and return the m different points with largest occurrences. It is clear that we only spend $O(1)$ cost for each sample, which leads to $O(s)$ time for Step 3 and $O(m)$ time for Step 4.

3.3 Parameter Settings and Time complexity

3.3.1 ***Parameter settings:*** If the post-processing is not allowed, we simply set the total samples $s = \mathcal{B}$. In practice, we observe that the post-processing often increases the search quality. Hence we will exploit this step with additional $O(dm + m \log k)$

time. For the time balance between the post-processing step and the other steps, we set $s = dm = \mathcal{B}/2$. Note that this setting has been used on the very recent work on budgeted MIPS [31].

3.3.2 ***Time complexity:*** It is clear that the cost of Step 1 is negligible and Step 2 takes $O(d)$ time. Using our proposed histograms with $O(s+n)$ space overhead, Step 3 and Step 4 run in $O(s)$ and $O(m)$ time, respectively. Hence fsWedge runs in $O(s)$ time, which completely depends on the budget $\mathcal{B}$ using the above settings.

3.4 Analysis of Wedge Sampling

Given a query $\mathbf{q}$, we denote by $\delta = \sum_{j=1, q_j \geq 0}^{d} \alpha_j q_j + \sum_{j=1, q_j < 0}^{d} \beta_j q_j$ the shifting value for any point $\mathbf{x}_i$. This shifting value is fixed given a query $\mathbf{q}$. Applying wedge sampling in the new transformation space, we have the sampling probability for the point $\mathbf{x}_i$ is $p'_i = \frac{\mathbf{x}_i \cdot \mathbf{q} - \delta}{\sum_i \mathbf{x}_i \cdot \mathbf{q} - n\delta}$.

For simplicity, we will show the analysis on the transform space since the input values are now non-negative. One can easily derive the analysis in the original space with the shifting value δ defined above. Consider the counting histogram *counter* of n counters corresponding to n point indexes, the following theorem states the number of samples required to distinguish between two inner product values τ_1 and τ_2.

Theorem 3.4. *Fix two thresholds $\tau_1 > \tau_2 > 0$ and suppose $s \geq \frac{3z \ln n}{(\sqrt{\tau_1} - \sqrt{\tau_2})^2}$ where $z = \sum_i \mathbf{x}_i \cdot \mathbf{q}$. With probability at least $1 - \frac{1}{n}$, the following holds for all pairs $i_1, i_2 \in [n]$: if $\mathbf{x}_{i_1} \cdot \mathbf{q} \geq \tau_1$ and $\mathbf{x}_{i_2} \cdot \mathbf{q} \leq \tau_2$, then $counter[i_1] > counter[i_2]$.*

Proof. Define $p_1 = \frac{\mathbf{x}_{i_1} \cdot \mathbf{q}}{z} \geq \frac{\tau_1}{z}$ and $p_2 = \frac{\mathbf{x}_{i_2} \cdot \mathbf{q}}{z} \leq \frac{\tau_2}{z}$. For $l = 1, \ldots, s$, we consider independent pair of random variables (X_l, Y_l) where

$$X_l = \begin{cases} 1 & \text{if } \mathbf{x}_{i_1} \text{is chosen at } l\text{th sample;} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_l = \begin{cases} 1 & \text{if } \mathbf{x}_{i_2} \text{is chosen at } l\text{th sample;} \\ 0 & \text{otherwise} \end{cases}$$

Define $X = \sum_{l=1}^{s} X_l$ and $Y = \sum_{l=1}^{s} Y_l$. We only consider the failure case where $Y - X \geq 0$. Applying Markov inequality for any $\lambda > 0$, we have

$$\begin{aligned} \mathbf{Pr}[Y - X \geq 0] &= \mathbf{Pr}\left[e^{\lambda(Y-X)} \geq 1\right] \leq \mathbf{E}\left[e^{\lambda(Y-X)}\right] \\ &= \mathbf{E}\left[e^{\lambda(\sum_l Y_l - \sum_l X_l)}\right] = \prod_{l=1}^{s} \mathbf{E}\left[e^{\lambda(Y_l - X_l)}\right] . \end{aligned}$$

We also have

$$\begin{aligned} \mathbf{E}\left[e^{\lambda(Y_l - X_l)}\right] &= e^{\lambda} p_2 + (1 - p_1 - p_2) + e^{-\lambda} p_1 \\ &\geq 2\sqrt{p_1 p_2} + 1 - p_1 - p_2 = 1 - \left(\sqrt{p_1} - \sqrt{p_2}\right)^2 . \end{aligned}$$

The equality holds when $\lambda = \ln \sqrt{p_1/p_2} > 0$. In other words, by choosing $\lambda = \ln \sqrt{p_1/p_2}$, we have

$$\mathbf{Pr}[Y - X \geq 0] \leq \left(1 - \left(\sqrt{p_1} - \sqrt{p_2}\right)^2\right)^s \leq e^{-s\left(\sqrt{p_1} - \sqrt{p_2}\right)^2} .$$

By choosing $s \geq \frac{3z \ln n}{(\sqrt{\tau_1} - \sqrt{\tau_2})^2} \geq \frac{3 \ln n}{(\sqrt{p_1} - \sqrt{p_2})^2}$ and the union bound, the theorem holds with probability at least $1 - 1/n$. □

Algorithm 3 GreedySam

```
Require: Number of samples $s$, and a discrete distribution $\boldsymbol{\kappa} = \{\kappa_1, \ldots, \kappa_n\} \in [0,1]^n$ and $\|\boldsymbol{\kappa}\|_1 = 1$
Ensure: A pre-sample set $\mathbf{s}$ of size $s$
1: $\mathbf{s} \leftarrow [0]^s$
2: For all indexes $i \in [n]$, insert $(i, \kappa_i)$ into a max heap H based on the value $\kappa_i$
3: for $j = 1$ to $s$ do
4:     Extract from H the element $(l, \kappa_l)$
5:     $\mathbf{s}[j] \leftarrow l$
6:     Insert $(l, \kappa_l - 1/s)$ into H
7: end for
8: return  Return $\mathbf{s}$
```

3.4.1 ***Trade-off between search quality and efficiency:*** By choosing s as Theorem 3.4, we have $\sqrt{\tau_1} - \sqrt{\tau_2} \geq \sqrt{\frac{3z \ln n}{s}}$. Assume that the top-$k$ value is τ_1, it is clear that the number of samples s controls the trade-off between search quality and efficiency. That is that the larger the budget $\mathcal{B}$ of computational operations (i.e. the larger s) we have, the smaller gap between the top-k largest inner product value and the other values we are able to distinguish.

3.4.2 ***Comparison to diamond sampling:*** For a fair theoretical comparison, we consider the similar setting as in [3] where we want to distinguish $\mathbf{x}_{i_1} \cdot \mathbf{q} \geq \tau$ and $\mathbf{x}_{i_2} \cdot \mathbf{q} \leq \tau/4$, and all entries in $\mathbf{X}$ and $\mathbf{q}$ are non-negative[2]. Applying Theorem 3.4, wedge sampling needs $s_w \geq 12z \ln n/\tau$. Diamond sampling [3, Theorem 4] needs $s_d \geq 12K\|\mathbf{W}\|_1 \ln n/\tau^2$ where all entries in $\mathbf{X}$ are at most K, and $\|\mathbf{W}\|_1 = z\|\mathbf{q}\|_1$. Since $K\|\mathbf{q}\|_1 \geq \tau$ for any τ, wedge sampling requires less samples than diamond sampling.

3.5 A Greedy Sampling Generator for a Discrete Distribution

This subsection presents implementation details of advanced sampling approaches, including wedge and diamond sampling for budgeted top-k MIPS. We first discuss a significant drawback on generating random samples from a discrete distribution, the core operation of these methods, when applying these schemes with budget $\mathcal{B} = o(n)$. Then, we introduce a greedy approach to carefully select these samples, which leads to *deterministic* versions of both sampling schemes. For simplicity, we assume that $\mathbf{q}$ and $\mathbf{X}$ are non-negative.

One big advantage of wedge sampling compared to diamond sampling is the ability of generating the pre-sample vectors from discrete distributions presented by column vectors of $\mathbf{X}$ (i.e. line 5 in Algorithm 1) on the query-independent phase. Furthermore, we can avoid the randomness provided by the Step 3 of Algorithm 2 by simply accessing the top-s_j point indexes from these pre-sample vectors $\mathbf{y}_j$. This suggested modification in [5] leads to a very efficient implementation which requires only simple counting and *sequential* memory access operations.

It is obvious that the more accurately these s_j sampled points represent their corresponding discrete distribution $\mathbf{y}_j$, the higher accuracy wedge sampling provides. However, in the budgeted setting with a very limited budget, i.e. $\mathcal{B} = o(n)$, the number of sample points s_j required for each dimension j is $\mathcal{B}/d \ll n$ in expectation. Since the number of samples s_j is much smaller than the size of the distribution, n, and since the data set is often dense,

[2]Diamond sampling's analysis only works on non-negative inputs.

it is impossible to approximate the distribution well. Hence the performance of both wedge and diamond sampling dramatically degrades, as can be seen in the empirical evaluation section.

To overcome this drawback, we propose a greedy approach to select s_j samples to present the *modes* of $\mathbf{y}_j$, instead of $\mathbf{y}_j$ itself. In other words, our greedy strategy selects indexes i corresponding to the largest values x_{ij} of $\mathbf{y}_j$. Since these local maxima reflect the distinguishable contributions of the dimension j on n inner products, they work well for differentiating the largest inner product values. Our approach can be seen as a greedy simulation of the condensed table lookup method [20] to select s samples from a discrete distribution $\mathbf{y}_j$. We first fill the table with s values from the vector $\lceil s\mathbf{y}_j/\|\mathbf{y}_j\|_1\rceil$. Then we greedily choose samples corresponding to the largest value, and continue to the next largest ones.

Algorithm 3 shows how our greedy approach, called *GreedySam*, selects $s \ll n$ samples from a discrete distribution $\boldsymbol{\kappa} = \{\kappa_1, \ldots, \kappa_n\}$ where $\|\boldsymbol{\kappa}\|_1 = 1$. We use a max heap H to keep track the largest value κ_i in order to priority sample them (Step 2). When we sample an index l with the largest value κ_l, we decrease κ_l by $1/s$ and insert it into the heap again (Step 6). The intuition behind the heuristic approach of subtracting $1/s$ for each sample, is that we expect any $\kappa_l \geq 1/s$ will be sampled at least one time.

It is clear that GreedySam runs in $O(n \log n + s \log n)$ time using the priority queue. Since we need to generate $s_{max} = n$ samples for each dimension, the pre-sampling process takes $O(dn \log n)$ time. GreedySam leads to deterministic versions of wedge and diamond sampling, called *dWedge* and *dDiamond*, respectively. These deterministic versions provide much higher accuracy than the randomized schemes with the alias generator, as can be seen in the experiment section.

3.6 A Fast and Deterministic Wedge-based Algorithm: dfsWedge

The combination of our series of algorithmic engineering techniques, including shifting technique, GreedySam and the fast wedge-based algorithm, yields a novel *deterministic* wedge-based algorithm, called *dfsWedge* for budgeted top-k MIPS. Our empirical evaluation shows that dfsWedge runs significantly faster than the state-of-the-art methods for budgeted and exact top-k MIPS while achieving at least 80% accuracy for top-5 MIPS on standard recommender system data sets.

4 EXPERIMENT

We implemented sampling schemes and other competitors in C++ and conducted experiments on a 2.80 GHz core i5-8400 32GB of RAM. We first show the empirical evaluation to compare GreedySam and the alias method on wedge and diamond sampling. Then we show a comparison between wedge and diamond on both efficiency and accuracy of top-k MIPS to confirm our theoretical findings. Finally, we compare the performance of our proposed *deterministic* wedge-based algorithms, *dfsWedge*, with other state-of-the-art methods, including Greedy [31] as a budgeted top-k MIPS and FEXIPRO [17] as an exact top-k MIPS

Data set	Movielens1M	Movielens10M	Netflix	Yahoo
d	150	150	200	50
n	3,952	65,133	17,770	624,961

Table 1: Overview of the data sets.

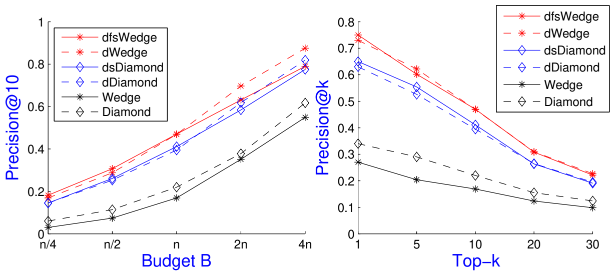


Figure 3: Accuracy comparison between wedge/diamond-based schemes using GreedySam/Alias generators on Movielens1M when fixing $k = 10$ and varying $\mathcal{B}$ (left); and fixing $\mathcal{B} = n$ and varying k (right).

on standard recommender system data sets. For measuring the accuracy and efficiency, we used the standard *Precision@k* and the speedup over the brute force algorithm, defined as follows.

$$Precision@k = \frac{|\texttt{Retrieved top-}k \cap \texttt{True top-}k|}{k}$$

$$Speedup = \frac{\texttt{Running time of bruteforce}}{\texttt{Running time of algorithm}}$$

4.1 Experiment Setup

Since both shifting technique and GreedySam can apply to both wedge and diamond, we implement their corresponding versions. Below is the list of all implemented algorithms used in our experiment.

- **Wedge-based schemes**: the randomized wedge sampling with the alias generator (*Wedge*), the deterministic version with GreedySam (*dWedge*), the fast and deterministic version with GreedySam and shifting (*dfsWedge*).
- **Diamond-based schemes**: the randomized diamond sampling with the alias generator (*Diamond*), the deterministic version with GreedySam (*dDiamond*), and the deterministic version with shifting (*dsDiamond*).
- **Greedy**: A greedy approach for approximate top-k MIPS on the budgeted setting [31].
- **FEXIPRO**: The start-of-the-art approach for exact top-k MIPS [17].
- **Bruteforce**: We use the Eigen library[3] which provides extremely fast C++ matrix-vector multiplication. Our bruteforce algorithm runs nearly 2 times faster than FEXIPRO for top-5 MIPS on the Yahoo data set.

We do not implement LSH-based approaches since they are not suitable for budgeted MIPS. Instead, we compare our sampling method to Greedy, a recent work on budgeted MIPS, which empirically outperforms LSH-based methods on budgeted MIPS [31].

Parameter settings: For the budgeted methods, including wedge, diamond and greedy, we simply set $\mathcal{B}/2$ for candidate screening and $\mathcal{B}/2$ candidate ranking. This means that we need $s = \mathcal{B}/2$ samples for sampling schemes and compute $\mathcal{B}/2d$ inner products to answer approximate top-k MIPS. Since diamond is a combination between wedge and basic sampling, its number of samples s_d is double that of wedge sampling s_w. For a fair comparison between wedge and diamond, we set $s_d = s_w/2 = \mathcal{B}/4$. We note that the candidate screening of Greedy is extremely fast. For a fair comparison between wedge-based schemes and

[3]http://eigen.tuxfamily.org/index.php?title=Main_Page

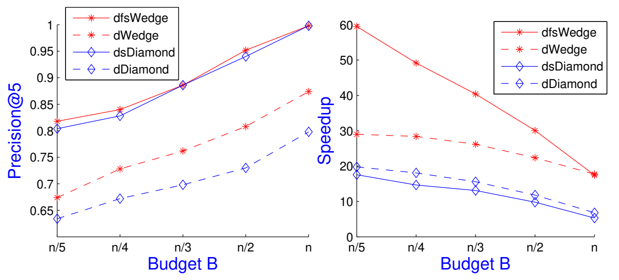


Figure 4: Comparison of *Precision*@5 (left) and speedup (right) between wedge/diamond-based algorithms when varying $\mathcal{B}$ on Movielens10M.

Greedy, we reduce the number of samples $s_w = \mathcal{B}/20$ to achieve similar running time.

4.2 Data Sets

We conducted experiments on standard recommender system data sets, including Movielens1M, Movielens10M, Netflix, and Yahoo, as shown in Table 1.

- **Movielens1M** and **10M**: We execute PureSVD [7] to generate user and item matrices of 150 dimensions.
- **Netflix**: We use the version of 200 dimensions provided by Greedy [31].
- **Yahoo**: We use the version of 50 dimensions provided by Greedy [31].

For all data sets, we randomly pick 100 items from the item matrices as the query sets.

4.3 Comparison between GreedySam and the alias method

This subsection conducted experiments on evaluating the peformance of GreedySam and the alias generator on both wedge and diamond. We computed the exact values of top-10 MIPS and executed the wedge-based and diamond-based schemes with GreedySam and alias generator. Note that $s_d = s_w/2$ for all comparison settings between wedge and diamond. Due to the similar results on other data sets, we only report the representative results of Movielens1M here.

Figure 3 shows that GreedySam provides superior performance compared to Alias since the *Precision*@10 of all sampling schemes with GreedySam are mostly 2 times higher than that of Alias. These gaps are persistent for both settings when we vary $\mathcal{B}$ (left) and the k (right). It is worth noting that the wedge-based algorithms, including dfsWedge and dWedge, achieve higher accuracy than diamond-based algorithms, including dsDiamond and dDiamond.

4.4 Comparison between wedge and diamond based schemes

In this subsection, we compare the top-k performance of wedge-based and diamond-based schemes with GreedySam since GreedySam outperforms the alias methods on these sampling schemes. For each sampling scheme, we consider two versions with and without the shifting technique, including dfsWedge, dWedge and dsDiamond, dDiamond. We measured their performance on *Precision*@5 value and speedup over the bruteforce search where we varied $\mathcal{B}$ and k. The parameter setting was the same as described in the previous subsection, i.e $s_d = s_w/2 = \mathcal{B}/4$, for the sake of comparison. Figure 4 reveals the performance of dWedge and dfsWedge compared to dDiamond and dsDiamond on Movilens10M.

It is clear that the shifting technique provides superior performance as illustrated on the left figure. In particular, both dfsWedge and dsDiamond achieve at least 80% when using $\mathcal{B} = n/5$ and reach the exact solution with $\mathcal{B} = n$, whereas without shifting, dWedge and dDiamond's accuracy are at most 80% even with $\mathcal{B} = n$.

Regarding the efficiency, wedge-based schemes significantly outperform diamond-based ones, as shown on the right figure. Note that we set $s_d = s_w/2$ since diamond sampling requires 2 times larger number of samples than wedge sampling. Diamond is still much slower than wedge because it has to execute the basic sampling on the query phase. This step requires costly random access operations while wedge only needs sequential access. We note that on Yahoo, the larger data set, the speedup gap of these methods is very significant, but not reported here.

In general, regarding accuracy and efficiency, dfsWedge illustrates substantial advantages with the highest accuracy and largest speedup on a wide range $\mathcal{B}$. In particular, dfsWedge runs several orders of magnitude faster than bruteforce, approximately 2 times faster than dWedge and 3 times faster than diamond-based methods. When $\mathcal{B} = n$, the speedup of dWedge is similar to that of dfsWedge. This is due to the fact that the complexity of dfsWedge $O(\mathcal{B}) = O(n)$ is nearly matching with the bottleneck of the candidate screening process of dWedge, i.e. $O(n \log n)$ times in practice.

4.5 Comparison between wedge, Greedy and FEXIPRO

This subsection presents experiments to measure the performance of wedge-based algorithms, including dWedge and dfsWedge, and Greedy and FEXIPRO on the Netflix, Movilens10M and Yahoo data sets. We again use *Precicision*@5 and speedup as our standard measures.

We note that Greedy has very fast candidate screening process. Hence, we decreased the number of samples for wedge $s_w = \mathcal{B}/20$ to achieve similar running time. For candidate ranking, we computed $m = \mathcal{B}/2d$ inner products for both wedge-based and Greedy methods. For a fair comparison, we investigated two different settings for Greedy. The first setting is standard with the same budget $\mathcal{B}$ as wedge. For the second setting, we increase the budget $\mathcal{B}$ by a factor of 2 for Netflix, called *Greedy2B*, and a factor 1.5 for Movielens10M, called *Greedy1.5B*. This means that Greedy2B and Greedy1.5B have, respectively, 2 times and 1.5 times more dot product computations than the wedge-based methods. Figure 5 shows the accuracy and efficiency of dfsWedge, dWedge and the two versions of Greedy, including Greedy and Greedy2B for Netflix, and Greedy and Greedy1.5B for Movielens10M.

On Netflix, wedge-based approaches achieve dramatically higher accuracy than both Greedy and Greedy2B although dWedge provides slightly higher accuracy than dfsWedge. In particular, Greedy methods suffer from very low accuracy when varying $\mathcal{B}$ on Figure 5(a) and (b). With $\mathcal{B} = 6n$, both Greedy approaches provide up to 40% *Precision*@5, whereas wedge-based algorithms return top-5 with nearly 70% accuracy. Figure 5(b), when fixing $\mathcal{B} = 4n$, the accuracy gap of these methods is at least 30% for a wide range of k.

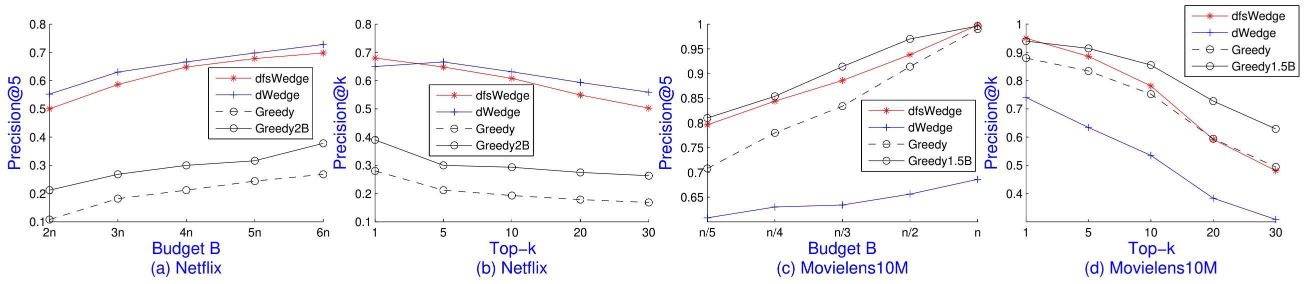


Figure 5: Comparison between dfsWedge, dWedge, Greedy and Greedy2B on Netflix when fixing $k = 5$ and varying $\mathcal{B}$ (a); fixing $\mathcal{B} = 4n$ and varying k (b); and between dfsWedge, dWedge, Greedy and Greedy1.5B on Movielens10M when fixing $k = 5$ and varying $\mathcal{B}$ (c); fixing $\mathcal{B} = n/3$ and varying k (d).

On Movielens10M, the accuracy of dfsWedge is consistenly between that of Greedy and Greedy1.5B while dWedge suffers from prominent loss compared to the other methods. When fixing $\mathcal{B} = n$ on Figure 5(c), dWedge provides up to 70% *Precision*@5, while dfsWedge and Greedy reach the exact result. When fixing $\mathcal{B} = n/3$ on Figure 5(d), dfsWedge and Greedy achieve at least 80% *Precision*@10 while dWedge only gives 55%. Both dfsWedge and Greedy1.5B present very high accuracy, at least 80% and higher than 90% *Precision*@5 when $\mathcal{B} = n/5$ and $\mathcal{B} = n/2$, respectively.

Figure 6 reveals the comparison of speedup between wedge-based and greedy-based approaches. While all methods run several orders of magnitudes faster than the bruteforce algorithm, the speedup of dfsWedge is consistently between the two Greedy versions on both data sets. For Netflix, Greedy2B is the slowest algorithm with $\mathcal{B} \geq n$ while dWedge is the slowest one with $\mathcal{B} \geq 3n$. It is natural since the bottleneck cost of dWedge is the candidate screening, i.e. $O(n \log m)$, to find top-m for candidate ranking. This observation is also consistent with Greedy1.5B on Movilens10M when $\mathcal{B} = n$. dfsWedge and Greedy offer significantly larger speedup compared to dWedge. In general, regarding both accuracy and efficiency, dfsWedge outperforms both dWedge and Greedy on Netflix and is comparable to Greedy on Movilens10M.

Figure 7 shows the observed *Precision*@k and speedup of dfsWedge, dWedge and Greedy on Yahoo, the large-scale data sets. dfsWedge still outperforms both dWedge and Greedy when varying $\mathcal{B}$ on top-5 MIPS. While Greedy obtains significantly higher *Precision*@5 than dWedge, the gap between dfsWedge and Greedy is considerable. The most dramatic difference is around 10% with $\mathcal{B} = 3n$ when dfsWedge achieves over 80% *Precision*@5 and Greedy offers less than 70%. Regarding the speedup, dfsWedge also runs faster than Greedy and up to 3 times faster than dWedge when $\mathcal{B} = n/3$.

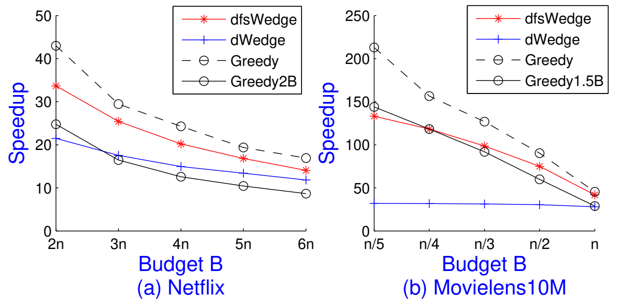


Figure 6: Comparison of speedup between dfsWedge, dWedge, Greedy and Greedy2B on Netflix and Movielens10M when varying $\mathcal{B}$ for top-5 MIPS.

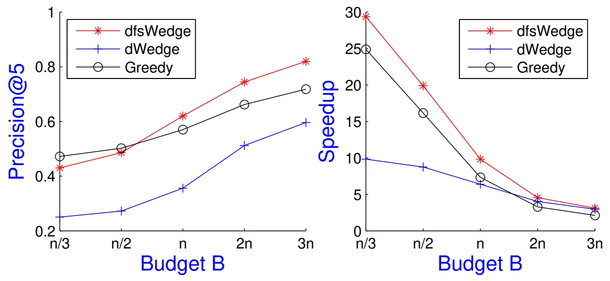


Figure 7: Comparison of *Precision*@5 (left) and speedup (right) between dfsWedge, dWedge, Greedy on Yahoo when varying $\mathcal{B}$ for top-5 MIPS.

We conclude our experiment by comparing dfsWedge, Greedy and FEXIPRO, an exact MIPS method on both accuracy and efficiency. Table 2 show that given at least 80% *Precision*@5, dfsWedge runs significantly faster than both Greedy and FEXIPRO. Furthermore, dfsWedge also achieves higher accuracy than Greedy on these data sets. In conclusion, regarding both accuracy and efficiency on the budgeted setting, dfsWedge has superior performance compared to dWedge and Greedy on the three data sets.

Data sets	dfsWedge		Greedy		FEXIPRO	
	Acc	Time	Acc	Time	Acc	Time
Netflix ($\mathcal{B} = 10n$)	0.80	**9×**	0.60	7×	1	1×
Movie10M ($\mathcal{B} = n/4$)	0.84	**118×**	0.84	**118×**	1	101×
Yahoo ($\mathcal{B} = 3n$)	0.82	**3×**	0.72	2×	1	_

Table 2: Comparison of accuracy and speedup between dfsWedge, Greedy and FEXIPRO on the three data sets.

5 CONCLUSIONS

This paper studies top-k MIPS given a limit of $\mathcal{B}$ computational operations and investigates recent advanced sampling algorithms, including wedge and diamond sampling to solve it. We theoretically and empirically show that wedge is competitive (often superior) to diamond on approximating top-k MIPS regarding both efficiency and accuracy. We also propose a series of algorithmic engineering techniques to deploy wedge sampling on budgeted top-k MIPS. Our novel deterministic wedge-based algorithm runs significantly faster than the state-of-the-art methods on budgeted MIPS and exact MIPS while maintaining the accuracy at least 80% on standard real-word recommender system data sets.

REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 2008.

[2] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*, pages 257–264, 2014.

[3] G. Ballard, T. G. Kolda, A. Pinar, and C. Seshadhri. Diamond sampling for approximate maximum all-pairs dot-product (MAD) search. In *ICDM*, pages 11–20, 2015.

[4] J. Bennett and S. Lanning. The netflix prize, 2007.

[5] E. Cohen and D. D. Lewis. Approximating matrix multiplication for pattern recognition tasks. *J. Algorithms*, 30(2):211–252, 1999.

[6] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[7] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.

[8] T. L. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100, 000 object classes on a single machine. In *CVPR*, pages 1814–1821, 2013.

[9] L. Devroye. *Non-Uniform Random Variate Generation(originally published with.* Springer-Verlag, 1986.

[10] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices I: approximating matrix multiplication. *SIAM J. Comput.*, 36(1):132–157, 2006.

[11] E. Fetaya, O. Shamir, and S. Ullman. Graph approximation and clustering on a budget. In *AISTATS*, 2015.

[12] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

[13] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[14] Q. Huang, G. Ma, J. Feng, Q. Fang, and A. K. H. Tung. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *KDD*, pages 1561–1570, 2018.

[15] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172, 2011.

[16] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[17] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD*, pages 835–850, 2017.

[18] W. Li, Y. Zhang, Y. Sun, W. Wang, W. Zhang, and X. Lin. Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement (v1.0). *CoRR*, 2016.

[19] S. T. Mai, X. He, N. Hubig, C. Plant, and C. Böhm. Active density-based clustering. In *ICDM*, pages 508–517, 2013.

[20] G. Marsaglia, W. W. Tsang, and J. Wang. Fast Generation of Discrete Random Variables. *Journal of Statistical Software*, 11(i03), 2004.

[21] S. Mussmann and S. Ermon. Learning and inference via maximum inner product search. In *ICML*, pages 2587–2596, 2016.

[22] B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. In *ICML*, pages 1926–1934, 2015.

[23] P. Ram, D. Lee, and A. G. Gray. Nearest-neighbor search on a time budget via max-margin trees. In *SDM*, pages 1011–1022, 2012.

[24] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[26] O. Shamir and N. Tishby. Spectral clustering on a budget. In *AISTATS*, pages 661–669, 2011.

[27] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *NIPS*, pages 2321–2329, 2014.

[28] R. Spring and A. Shrivastava. Scalable and sustainable deep learning via randomized hashing. In *KDD*, pages 445–454, 2017.

[29] C. Teflioudi and R. Gemulla. Exact and approximate maximum inner product search with LEMP. *TODS*, 42(1):5:1–5:49, 2017.

[30] R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[31] H. Yu, C. Hsieh, Q. Lei, and I. S. Dhillon. A greedy approach for budgeted maximum inner product search. In *NIPS*, pages 5459–5468, 2017.

[32] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.