Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia COPPE UFRJ

UNSUPERVISED CONCEPT EXTRACTION IN AN INTRODUCTION TO PROGRAMMING COURSE

Laura de Oliveira Fernandes Moraes

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Carlos Eduardo Pedreira

Rio de Janeiro
Fevereiro de 2021

UNSUPERVISED CONCEPT EXTRACTION IN AN INTRODUCTION TO PROGRAMMING COURSE

Laura de Oliveira Fernandes Moraes

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Carlos Eduardo Pedreira

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2021

Moraes, Laura de Oliveira Fernandes

Unsupervised Concept Extraction in an Introduction to Programming Course/Laura de Oliveira Fernandes Moraes. – Rio de Janeiro: UFRJ/COPPE, 2021.

XIII, 86 p.: il.; 29, 7cm.

Orientador: Carlos Eduardo Pedreira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2021.

Referências Bibliográficas: p. 69 – 86.

1. Topic Modeling. 2. Clustering. 3. Educational Data Mining. 4. Computer Science Education. I. Pedreira, Carlos Eduardo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Personally, as an educator, I've struggled to understand which contents my students are mastering and which ones they are having difficulties with. As a data scientist, I value the awareness and consistency given by data-driven decisions. I dedicate this thesis to my students.

Agradecimentos

Meu conselho para quem quer fazer um doutorado: pesquise um assunto que você curta! À primeira vista parece ser um conselho óbvio, mas não custa reforçar. São muitas horas necessárias e dedicadas para fazer esse trabalho, são fins de semanas e feriados, são férias e muitas (muitas!) conversas e trocas de ideias para a pesquisa seguir em uma direção frutífera. Por isso, deixo aqui os meus agradecimentos para quem, direta ou indiretamente, fez parte dessa história.

Agredeço ao meu orientador, Carlos Eduardo Pedreira, que, na medida correta, sabe empurrar, questionar e confiar. Agredeço pela pergunta: "o que VOCÊ quer pesquisar?" feita nos corredores do PESC e que me motivou a buscar um assunto do meu interesse. Agradeço à todos os professores e funcionários do PESC por girarem a roda do departamento e torná-lo um lugar agradabilíssimo e motivador de trabalhar. Agradeço aos meus amigos e colegas do LaSI (ou seria LabIA?) com os quais, além de aprender sobre os mais diversos assuntos, me diverti e aprimorei as minhas habilidades em cantar.

Agradeço à Carla Delgado, que desde o início foi entusiasta desse projeto. Posso dizer com toda certeza que sem seu apoio não teríamos conseguido ir tão longe. Agradeço à todos os professores e alunos que participaram dessa pesquisa tanto utilizando a ferramenta como avaliando os resultados. Agradeço ao Departamento de Ciência da Computação da UFRJ por abraçar esse projeto e torná-lo realidade.

Agradeço aos organizadores e amigos do Data Science for Social Good. Espero ter podido ensinar pelo menos 10% do que aprendi com vocês em intensos 3 meses. Agredeço à Sherry Sahebi e seus alunos que me receberam de braços abertos na SUNY Albany e me proporcionaram o pedaço que faltava para fechar a tese.

Agradeço aos meus sócios e amigos da TWIST: Fernando Ferreira e Felipe Grael. Não consigo mensurar o quanto o companheirismo de vocês contribuiu para eu chegar até aqui. Essa tese leva um pedacinho de tudo que contruímos na TWIST esses anos. Aproveito e agradeço à todos que já trabalharam com a gente, deixando seu tijolinho na história da empresa.

Agradeço ao João Paixão, meu grande companheiro, que contribuiu não somente com apoio e paciência, mas com ideias, experiência e conhecimento. Agradeço aos meus pais Altamirando e Mônica por sempre terem me dado a segurança e independencia de buscar meus objetivos. À minha irmã, ao meu sobrinho, à minha afilhada, aos meus avós, tios, primos e à toda a minha família pelos prazerosos fins de semana em Itaipava, almoços de domingo, aniversários, viagens ou qualquer outra desculpa para nos reunirmos e jogarmos conversa fora. À todos os meus amigos por estarem presentes nas dificuldades e nas alegrias.

Agradeço à UFRJ, minha casa pelos últimos 15 anos, e ao CNPq pelo suporte financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

EXTRAÇÃO NÃO-SUPERVISIONADA DE CONCEITOS EM UM CURSO INTRODUTÓRIO DE PROGRAMAÇÃO

Laura de Oliveira Fernandes Moraes

Fevereiro/2021

Orientador: Carlos Eduardo Pedreira

Programa: Engenharia de Sistemas e Computação

Determinar manualmente os conceitos presentes em um grupo de perguntas é um processo desafiador e demorado. No entanto, este processo é uma etapa essencial durante a modelagem de um ambiente virtual de aprendizagem, uma vez que é necessário o mapeamento entre conceitos e perguntas para avaliação automática de conhecimento, produção de *feedback* e recomendação de exercícios. Esta tese fornece ferramentas para auxiliar no ciclo de aprendizagem com foco na extração de conceitos. Nós investigamos modelos semânticos não supervisionados para apoiar professores de ciência da computação nesta tarefa e propomos um método para transformar soluções de código fornecidas por professores em documentos de texto, incluindo as informações da estrutura do código. Primeiro, projetamos, implementamos e implantamos um ambiente de aprendizado para coletar dados de professores e alunos. Este ambiente foi construído baseado em uma metodologia de ensino. Em seguida, extraímos a relação latente entre os exercícios e avaliamos os resultados usando um conjunto de dados externo. Consideramos a interpretabilidade dos conceitos recuperados utilizando dados de 14 professores, e os resultados confirmaram seis clusters semanticamente coerentes, atingindo 0,75 na métrica normalizada de informação mútua pontual. Esta métrica está positivamente correlacionada com a percepção humana, tornando o método proposto útil na anotação e agrupamento semântico de códigos. Por fim, comparamos este método com métodos focados no conhecimento do aluno de modo a extrair a relação semântica latente das questões através da perspectiva do aluno. Por fim, propusemos uma visualização que agregue a performance dos alunos para acompanhar seu desempenho e identificar possíveis pontos de dificuldade.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

UNSUPERVISED CONCEPT EXTRACTION IN AN INTRODUCTION TO PROGRAMMING COURSE

Laura de Oliveira Fernandes Moraes

February/2021

Advisor: Carlos Eduardo Pedreira

Department: Systems Engineering and Computer Science

Manually determining concepts present in a group of questions is a challenging and time-consuming process. However, the process is an essential step while modeling a virtual learning environment since a mapping between concepts and questions using mastery level assessment and recommendation engines are required. This thesis provides tools to assist in closing the learning feedback loop focused on concept extraction. We investigated unsupervised semantic models (known as topic modeling techniques) to assist computer science teachers in this task and propose a method to transform Computer Science 1 teacher-provided code solutions into representative text documents, including the code structure information. First, we projected, implemented, and deployed a learning environment based on a teaching methodology to collect professors and student data. Then, we extracted the underlying relationship between questions and validated the results using an external dataset by applying non-negative matrix factorization and latent Dirichlet allocation techniques. We considered the interpretability of the learned concepts using 14 university professors' data, and the results confirmed six semantically coherent clusters, achieving 0.75 in the normalized pointwise mutual information metric. The metric correlates with human ratings, making the proposed method useful and providing semantics for large amounts of unannotated code. Finally, we compare this method with methods focused on the students' knowledge to extract the questions' latent semantic relationship through the students' perspective. As we could not find a significant relationship between concepts found using the student-focused methods and the ones provided by the professors, we proposed a new visualization to track student performance and pinpoint students' difficulties and achievements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It has been a while since educators have been challenged by the idea of using technology to support affordable and scalable personalized educational experiences while keeping learning quality. Even before the COVID-19 pandemic, technology and pedagogical strategies for distance learning were already being broadly discussed [1–5]. The Google Trends engine shows that the search for "distance education" has doubled between February and March 2020 in the world [6]. More than ever, there is a demand for tools to support delivering these personalized learning experiences while still incorporating students' environments' difficulties. Besides individual and targeted support for each student, some other needed features are asynchronicity and ubiquity, letting students learn at their own pace in whichever device they have available for them [4, 7, 8].

This thesis is inserted in the educational data mining (EDM) context, an emerging research area centered on developing methods to better understand students by analyzing large-scale data from educational settings [9]. It derives from regular data mining, the field of discovering novel and potentially meaningful patterns from large amounts of data in automatic or semiautomatic ways [10]. However, EDM differs from it by integrating psychometric methods to consider the statistical data dependence and the multiple levels of information hierarchy in educational data [11, 12].

In regular classrooms, educators perform "knowledge discovery" (in the EDM sense) every day by observing student behavior and outcomes in assignments and tests. From their observations, they can determine which concepts students have mastered or are still struggling with and consequently adapt teaching strategies. In distance learning, this observation is limited, and, therefore, we need enhanced tools to support this task. Also, student log analysis can provide insights to improve teaching and learning strategies, course curricula and support other sense-making processes. However, for it to be useful, this amount of data has to be presented in a way for stakeholders to act on it. Recent learning analytics reports pointed out that we need to approximate the already rich published research to actual deployment and practice [13–15]. Also, to further advance the field, hybrid approaches should be considered, inserting the instructors on the learning loop to interpret the outcomes of the algorithms and take appropriate actions [13, 16, 17]. Rosé *et al.* [14] advocate the use of explanatory learner models to provide actionable insights, in addition to accurate predictions.

This thesis uses as a case study the introduction to programming classes at the Universidade Federal do Rio de Janeiro (UFRJ). Learning to program is admittedly a difficult task, already studied by several authors [18, 19]. Understanding the cognitive processes involved in this task significantly improves teaching-learning proposals [19, 20]. Lately, there have been attempts to mine virtual learning environments students' interaction data [21] to obtain insights about students' behavior, but few of them have shown how to approximate the research findings with practice.

1.1 Objective

According to Lan et al. [22], the learning feedback loop consists of (1) continuous analysis of learners' interaction with learning resources to monitor learning evolution and (2) react to the results of the monitor and analysis by providing timely feedback, recommendations, and remediation to students. Previous intelligent tutoring systems (ITSs) and personalized learning systems (PLSs) used rule-based approaches to close the learning feedback loop [22]. We would like to add a third step into this loop that consists of a longitudinal analysis of the results to improve pedagogical methodologies [23, 24].

This thesis's general goal is to provide tools to assist in closing the learning feedback loop applied to computer science education. Aiming to improve step (1), we reviewed the existing literature on tools to collect learners' interaction data on computer science and proposed and implemented an enhanced architecture scheme that better suits our objective. Concerning step (2), we built an early dropout prediction model, revisiting and consolidating results from exploratory research found in the literature. A common requirement in every step of the learning feedback loop consists of mapping the concepts needed in each learner interaction since students' mastery level assessment, feedback production, and next steps recommendation depend on these mappings. However, manually identifying the concepts required to answer questions can be time-consuming and difficult, increasing the need for tools to assist teachers in the tasks. Desmarais [25] suggested that even partial automation of the process can be highly desirable. Besides decreasing the manual labeling required from the experts, the process automation also results in a more objective and replicable mapping.

This thesis is focused on concept extraction, and our main goal is to explore the problem of extracting the skills that students need to solve the learning activities, in which single or multiple skills are required. Our proposal focuses on extracting unsupervisedly these latent skills combining two different approaches: deriving it from the solutions for each problem and students' performance. The first one is more suitable for human-interpretability, whereas the second considers student knowledge and can grasp easier and harder concepts. Therefore, combining both allows us to navigate between interpretable concepts and student knowledge estimation of each concept. We propose unsupervised semantic methods, known as topic modeling techniques [26–29], as more interpretable methods for experts, to be applied in introductory computer science problems.

1.2 Contribution

This thesis researches different improvements around the learning feedback loop, providing contributions to different areas. The main contributions of this thesis are:

- On the data collection and availability domain:
  1. Use of methodological teaching theory to construct a learning environment.
  2. Improvement of personalized learning environment architecture proposed in the literature.
  3. An anonymous open-source student interaction database for multiple classes and a Python problems database with corresponding solutions and categories.
  4. Real user studies from multiple CS1 courses at the Federal University of Rio de Janeiro.
  5. Deployment of a large-scale functional system at the Federal University of Rio de Janeiro.
  6. Early dropout prediction, revisiting results from exploratory research found in the literature.
- On the unsupervised concept extraction domain:
  1. A tokenization structure to transform raw code snippets into a document-term matrix.
  2. A code-clustering method to optimize positively correlated metrics for human-interpretability.
  3. Experts validation, illustrating how the proposed method can support questions' exercise labeling using each topic's terms.

1.3 Document Organization

The text is organized as follows:

- Chapter 2 reviews the literature on computer science intelligent tutoring systems, personalized learning systems, and question attribute extraction.
- Chapter 3 describes the proposed architecture and developed system. The system requirements are detailed along with the collected data and their possible use.
- Chapter 4 performs an exploratory data analysis, providing a better understanding of the most relevant features that determine student outcomes. Part of this chapter was presented at the 4th Educational Data Mining in Computer Science Education (CSEDM) Workshop [30].
- Chapter 5 presents the used methodology to extract concepts unsupervisedly. The concepts are extracted from code solutions provided by professors and from student performance data. Part of this and the next chapters were published at the IEEE Transactions on Learning Technologies [31].
- Chapter 6 presents the results obtained using the methodology described in Chapter 5. This chapter also presents the students' evaluation of the learning environment described in Chapter 3.
- Chapter 7 discusses the general conclusions and presents the next steps.

Chapter 2

Related Work

This thesis proposes improvements around the learning feedback loop as introduced in Chapter 1. The main results are the two different perspectives to discover which latent skills are involved in a given learning task. Our first approach uses textual solutions for the problems as input and clusters them accordingly to the used terms. Our second approach extracts the skills by observing students' outcomes at each problem. Therefore, this chapter begins by reviewing learning environments that support automated assessment for Python exercises and drop-out prediction literature findings. Then, we review methods to associate general attributes to each exercise. These attributes can vary from question style to the concepts needed to solve a problem. We then review approaches to cluster code and extract its underlying concepts (or skills). Finally, we review methods to derive unsupervisedly these concepts based on the observed student outcomes.

2.1 Programming Learning Environments

In this section, intelligent tutoring systems and personalized learning systems to support learning the Python language are revised. Personalized Learning Systems (PLS) are systems designed to assist students' tutoring on a personalized level. They have been around since the 1960s [32], having its first formal definition in the 1990s [33]. Nowadays, several different PLSs cover a broad range of subjects with success used by hundreds of thousands of students a year. Concerning the subject of PLSs in programming languages, Table 2.1 shows a selected list gathered from the 2015 and 2018 reviews [34, 35]. Even though different environments can support students in learning, this list is restricted to those that provide an automated Python exercises assessment. Python is a general-purpose language, which means it can be used in a large variety of projects and can be great to stimulate students since they can work on projects they relate to. Python also is user-friendly, and for the past seven years, it has been the fastest-growing programming language [36], being correlated with trending careers, such as DevOps and Data Scientist [37]. According to the 2015 literature review on educational data mining and learning analytics in programming [34], only 11% of the papers about programming courses reported using Python as the course language. However, this is changing. A 2017 review on the introductory programming courses in Australasia universities [38] reported a shift from Java to Python in the past years. The 2018 review on introductory programming literature [35] already presents a higher number of papers using Python as the course language.

Our proposed system (Machine Teaching) and the presented systems were analyzed, comparing their theoretical foundation, system functionalities, log granularity, and evaluation, producing the benchmark from Table 2.1. About half of the systems are not built upon a teaching or pedagogical methodology; they are mostly an environment to write and correct code. All of the systems support students and professors use. For students, the systems usually work like an integrated development environment (IDE), where students can write code and receive impromptu feedback. However, the functionalities for professors may vary. In UUhistle, for example, the professors use the system to run simulations and examples. Most of the systems allow the professors to see if the student passed or not an individual question. However, just the URI Online Judge and PCRS provide in-depth analysis of student submissions through learning analytics dashboards.

Ihantola *et al.* [34] also define a typical architecture for these systems in their 2015 review. This architecture will be detailed in Section 3.3 with an enhanced proposed architecture.

Table 2.1: List of programming PLSs that support automated assessment. Adapted from [34, 35]

System	**Is it developed based on a teaching or pedagogical methodology? Which one?**	**In-depth analysis of student submissions**	**Log granularity**	**System evaluation**
CloudCoder [39, 40]	No	No	Keystrokes	Not evaluated
CodeWorkout [41]	Drill-and-practice	No	Keystrokes	Survey
PCRS [42]	Peer Instruction	Yes	Submissions	Not evaluated
URI Online Judge [43, 44]	No	Yes	Submissions	Survey
UUhistle [45]	Visual Program Simulation	No	N/A	Survey and control group
Pythy [46]	No	No	Keystrokes	Survey
Web-CAT [47]	No	No	Submissions	Control group
Machine Teaching [30]	Imperative programming (functions first)	Yes	Submissions	Survey

2.2 Early Dropout Prediction

Early prediction in the educational context is the ability to predict students' outcomes based on their performance in the course's first weeks to provide professors enough time to plan and execute interventions. Models to predict student performance and course non-completion (dropout) using data from the first weeks have been widely studied in the literature [48–55].

On the dropout prediction matter, initial research had been limited to single courses, which led to questions about the generalizability and replicability of these findings [56]. Therefore, Andres *et al.* [56] proposed a replication framework (MORF) to evaluate the previous findings across a multitude of massive open online courses (MOOCs). They were corroborated: forum participation and time dedication for the course (either finishing assignments or browsing in forums) were good predictors for course completion, as studies before have stated. Alamri *et al.* [51] focused on using only two features to predict student dropout in a group of five different MOOCs. They concluded that the time spent and the number of accesses were the best predictors for course completion.

Al-Shabandar *et al.* [57] also use MOOC data to investigate if geographical and behavioral features are correlated with course completion. They found out that location matters, probably due to language and educational level barriers. Also, they noted that clicks and the number of unique days that learners interact with the system are the most important features to predict student course withdrawal.

Damasceno *et al.* [21] cross-checked literature results in course completion using a Brazilian dataset. They used random forest and decision trees and combined different features. They found that assignment completion is a good predictor of course completion. Pereira *et al.* [54] proposed using CART4.5 to predict dropout in a face-to-face introduction to programming course at a Brazilian university. They used data from the first two weeks of the course from an online judge system used in class to predict the outcome. Unlike the MOOCs research, the dataset is small to medium-sized due to its course restriction in this study. Therefore, the same verified results could not be previously assumed. Their relevant features were the percentage of correct test cases for each submission, the access frequency, and the total number of submissions. In this thesis, we aim to verify these literature results by reproducing this experiment in our dataset.

2.3 Extracting Questions General Attributes

The existing methods to identify concepts from a set of CS1 exercises involve manual work and input from experts [58, 59]. For example, Sheard *et al.* [58] characterized introductory programming examination questions according to their concept areas, question style, and required skills. Participants manually classified the questions and the determined topics covered alongside the necessary skill levels to solve them. Nonetheless, applying a successful approach in a different set of exercises requires a new manual labeling stage, which may not be achievable.

One strategy to overcome this issue and minimize the domain experts' workload is to apply supervised learning. Previous research in question classifications used supervised learning to classify questions according to the level of difficulty [60], Bloom's taxonomy [61], answer type [62], and subject [63]. In Godea *et al.* [62], the features are derived from the questions, using part-of-speech tags, word embeddings, inter-class correlations, and manual annotation. Supraja *et al.* [61] use a grid search to analyze different combinations of weight schemes and methods to find the best set of parameters to build a supervised model to classify questions given Bloom's Taxonomy. Its main cost is the manual annotation of all labels, impractical when applying to large datasets. Unsupervised learning can group similar items without a predefined label, but it is harder to ascertain the results since there is no objective goal to analyze, and evaluating the clustering outcomes becomes a subjective task. Unsupervised learning techniques have been used to address EDM problems [64–68]. For example, Trivedi *et al.* [69] use spectral clustering with linear regression to predict student performance. In the questions' classification context, an unsupervised approach using K-means, as a clustering algorithm [70], was proposed to group similar learning objects (such as handouts, exercises, complementary readings, and suggested activities). Still, K-means does not provide a list of features that best characterize each cluster, making the expert infer them manually by reading a sample of each cluster's exercises.

2.4 Extracting Concepts from Code

It is possible to use the code provided as answers to the exercises as input while restricting the concept identification to the CS1 domain. Unsupervised learning using code as input is achieved by calculating their abstract syntax trees (ASTs) and clustering the most similar ones; besides, various strategies calculate similarity among trees. Huang *et al.* [71] and Nguyen *et al.* [72] use edit tree distance, Paasen *et al.* [73] apply sequential clustering algorithms, and Price *et al.* [74] and Mokbel *et al.* [75] fragment each AST into subgraphs, pair similar subgraphs and compute the distance between the subgraphs within a pair. Concepts identification in these scenarios requires experts to read samples of exercises from each cluster to infer the relevant features. It is also unclear how to generalize it to other domains where the answers are not constructed directly using a tree or a graph.

We propose a CS1 concept identification method to support experts in labeling concepts by providing the relevant features in each cluster in Section 5.2. This method could potentially be extended to different domains since it only requires text inputs. We develop a code tokenizer that uses its code structure to augment and improve the corpus. We apply topic modeling techniques where the documents are the code provided as answers to the exercises. In the software engineering field, concepts are extracted from code using programmers' annotations [76]. Similarly, code clustering, identifying similar features/concepts in a repository, predicting bugs, analyzing/predicting source code evolution, tracing links between modules, and detecting clones are widely applied in the same field [77]. For the EDM, Azcona *et al.* [78] created a Python code submission tokenizer to setup features to generate code embeddings. Unlike the software engineering context, code snippets in CS1 are small in size and lack annotations.

Although the LDA in Blei *et al.* [27] is a common technique in topic modeling, it does not perform well in short texts (code in this context) because the traditional way of extracting terms does not provide enough textual words to characterize a specific topic [79, 80]. It is necessary to decrease the latent document-topic or word-topic spaces, making them more specific for each context. Hsiao *et al.* [81, 82] propose a topic-facet LDA model using sentence LDA (SLDA) with a facet representing a more specific topic and all words from a sentence belonging to the same facet. Zhao *et al.* [83] decrease the latent space by creating a common word distribution with denominated background words, which are the same for every topic. Steyers *et al.* [84] and Rosen-Zvi *et al.* [85] adopted a similar strategy. In their method, the generative process to create a document decreases the space by choosing an author and then choosing a topic. Li *et al.* [86] use a distribution over tags to restrict the latent topic space before inferring the documents' topic distribution.

Another approach to overcome the lack of textual words is to increase the representation, so Weng *et al.* [87] proposed an LDA to cluster subjects' twitterers (people who tweet). Their method increases document representation by aggregating all tweets from a single user into one document. Abolhassani and Ramaswamy [80] enhance short semi-structured texts by augmenting the corpus' structure, which is similar to the method we adopted in this thesis. In comparison with the standard literature tokenizer, our proposed tokenizer increased the vocabulary size (total terms) from 287 to 2388 and the average of terms per document from 23 to 137. We maintain a 95% sparsity, which agrees with the sparsity of the long-text documents from Syed and Spruit [88] and Zhao *et al.* [83], i.e., successfully clustered using topic modeling techniques.

2.5 Extracting Concepts from Student Performance

A second approach to extract the skills a student needs to solve a problem is to use actual student data. Student performance data concern all data acquired on a student's level while interacting with an intelligent learning environment. They are primarily used to create a student model to assess students' knowledge. The most widely used model families for this task are the bayesian knowledge tracing (and derivatives) [89–97] and logistic models [94, 98–100]. In both model families, this process is done in two steps: define the skills needed in each question and then train a supervised model to infer the students' mastery of each skill based on their outcomes for each question. However, associating each question's skills, as stated in Section 2.3, requires extensive effort and time, being static for the set of problems to which they were defined.

2.5.1 Unsupervised Student Performance Models

Researchers have attempted to extract the question-skill association automatically using as input student item-response data. According to Barnes [101], Patrick Brewer started investigating questions' underlying relationship extraction methods directly from student data. When using student data, the goal is to capture students' differential performance (patterns) when answering questions [101].

The question-skill association table is called Q-matrix, and it is represented as a binary matrix where the rows are the latent concepts/skills and the columns are the questions. If the concept is needed to solve the question, the corresponding cell is filled with one; otherwise, it is filled with zero [92, 101, 102]. In his master's dissertation, Brewer started with a predefined Q-matrix and created simulated students with different knowledge levels. Then, he generated students' outcomes based on their knowledge and the Q-matrix. He then used the "Q-matrix method" [101] and factor analysis [103] to retrieve Q-matrices based only on student-generated outcomes. Factor analysis is a data reduction technique to find latent factors from the underlying data structure. It searches for the maximum common variance between the variables. The Q-matrix is a method based on the hill-climbing optimization technique [101]. It starts by creating a random Q-matrix and improves it at each iteration by flipping random bits on the previous Q-matrix. If the total error after flipping the bits is smaller than the previous iteration, the modification is saved. The total error is calculated by comparing students' real outcomes and their expected/predicted outcome, derived from the generated Q-matrix. Barnes [101] compares the Q-matrix method with factor analysis and k-means clustering using the Hamming distance. The Q-matrix method performed better than factor analysis and similar to k-means. Compared to the k-means clustering approach, the Q-matrix method provides an interpretable model, being a more appropriate tool for teachers to understand students' knowledge and misconceptions. However, when compared with expert Q-matrices, they often do not correspond. Barnes predicted this behavior because the Q-matrix method's primary motivation was to reflect students' differential patterns that expert Q-matrices were already not reflecting. Nevertheless, is it possible to extract and connect unsupervisedly both Q-matrices, one that contains students' differential patterns and another for the experts' intuition?

Researchers kept trying to find this Q-matrix using theory from other domains. Estimating the latent concepts from questions and students' outcomes can be posed as an item recommendation problem. If we use a movie recommendation system as an example, each movie's user ratings are represented by the student outcome in each question. They are both the observable information we have about a user preference/student knowledge. In both scenarios, we understand the latent characteristics involving the movies/questions and how each user/student likes/understands each latent characteristic. So, we can import techniques from collaborative filtering research to solve this problem, such as non-negative matrix factorization (NMF). A straightforward algorithm to solve the NMF optimization is a gradient descent approach to discover the latent matrices. However, as in a recommendation system setting, not all students may have answered the same questions in an authentic class. Winters *et al.* [104] propose using an alternative NMF optimization to estimate student knowledge and extract concepts from questions unsupervisedly. Following the item-response theory (IRT) guidelines, they propose two enhancements: 1) using IRT to estimate missing values and then use standard NMF, and 2) an NMF modified algorithm that applies a logistic regression on the factorized matrices multiplication since this could better match the mental model behind the student skills. They call this method GGLLM. They also propose an NMF extension called NMF+K, where they query an expert for some portion of the Q-matrix (called relevance matrix in the paper). Unlike the other papers, they are mainly interested in these algorithms' ability to extract a valid Q-matrix. They used experts to manually group each question into a concept cluster (they call topics in the study) and calculated precision and recall metrics after hard-assigning each question to a single concept. They also calculated the reconstruction error between their found Q-matrix and a Q-matrix provided by experts. When using matrix factorization, the order of the latent factors may vary in the latent dimension. It is not clear on the paper how they align the extracted latent concepts and the concepts from experts to assess the reconstruction error correctly. Winters *et al.* paper's key result is to show the feasibility of these methods, even though the results' precision was too low and inconclusive to build useful tools. This result indicates the need for more research and new proposals to address this issue, one of the motivations for this thesis.

Desmarais [105] replicated part of Winters's study to understand in which conditions the NMF technique is effective. To investigate these conditions, he proposed a simulator where: 1) each question is only associated with one concept, and 2) the students' performance depends either on the topic skill or a topic skill, item difficulty, and student general expertise combination. In the simulated scenario where the topic skill is the only factor that affects students' performance, NMF is highly effective. However, when applying it to a real dataset, it does not achieve good results, indicating that student performance is not based only on this assumption. In the second scenario, where students' outcome depends on topic skill, item difficulty, and student general expertise, NMF only performs well if the topic skills factor influences the outcome in the same proportion as the other two factors. His results indicate that topic skill has a much lower influence on student performance than generally thought. These results were also a motivation for this thesis. Can we complement student performance data to extract a better Q-matrix?

In Lan *et al.* [22], sparse factor analysis (SPARFA) was defined as a problem of factor analysis under three assumptions: 1) low dimensionality (the number of latent concepts is small relative to both the number of learners and the number of questions); 2) sparsity (each question is associated with just a few concepts, creating a sparse Q-matrix), and 3) non-negativity (the entries in the Q-matrix are non-negative, providing interpretable values for the student knowledge model).

Compared to the NMF assumptions, the SPARFA problem tries to estimate an extra parameter: the question difficulty, which is not present when solving the NMF problem, and that Desmarais suggested it as a factor that influences student performance. They propose two algorithms to retrieve the latent dimensions under these assumptions: SPARFA-M, based on maximum-likelihood, and SPARFA-B, using a Bayesian approach based on Markov chain Monte Carlos (MCMC). They experimented with synthetic and real education datasets, showing the SPARFA algorithms' efficacy in estimating student performance. Compared to the Q-matrix method, NMF and the SPARFA family retrieve question-concept associations in real-scaled values instead of binary values, which allows for different weights between questions and the underlying concepts. In these experiments, they did not directly validate the Q-matrix; they associated tags provided by experts to each question and created a bridge to navigate between the extracted abstract latent concepts and the tags, representing the experts' point of view. In this way, even though the extracted Q-matrix does not match the concepts imagined by experts, this methodology can still be used to provide the experts' insights about students' knowledge according to the experts' perspective.

The four methods described above (factor analysis, Q-matrix method, NMF, and SPARFA) do not consider student learning improvements while solving the exercises, relying on static student knowledge states. More recent methods, such as Topical HMM [106] and tensor factorizations [107, 108] consider a time dimension that can account for student knowledge improvement between student attempts in answering questions. Topical HMM creates a joint model of topic modeling and hidden Markov model. However, this approach has a trade-off between model interpretability and performance. The feedback-driven tensor factorization (FDTF) [108] method is a tensor factorization technique initially designed for student performance prediction tasks. It models the student's knowledge over attempts (as a time dimension) and assumes that it would be monotonically increasing after each learning material interaction. This method will be explained in Chapter 5.

This literature review included methods to estimate student knowledge based on student performance data without a predefined concept to question mapping. In summary, these methods provide interpretability, do not need previous association, and can be used to predict student performance. In all the methods, the interpretability is given by the Q-matrix's sparsity and non-negativity constraints, which is a proxy to the human intuition of additive parts to form a whole [26]. In this case, student performance is seen as a result of several factors that can only increase success probability.

On the other hand, we have little information on how the extracted latent question-concept associations relate to experts' perceptions. Barnes [101] visually compared different clustering schemes highlighting which elements are put in the same cluster by the Q-matrix method and experts. Winters *et al.* [104] validated it with experts but did not get conclusive results. Lan *et al.* [22] asked experts to manually label the concepts and propose a way to linearly transform the extracted unsupervised Q-matrix into a manually annotated one.

Our main goal is to extract unsupervisedly a Q-matrix that fulfills both requirements: it contains student data and experts' perceptions. Therefore, we use topic modeling to extract the concepts from the experts' point of view unsupervisedly and relate them to student performance. We compare one static method (NMF) with a dynamic method (FDTF) for student performance.

Chapter 3

Data Acquisition System and User Interaction

This chapter presents the teaching methodology used to design and implement the learning environment, named Machine Teaching. Then, we present the web system developed to acquire student data and interact with students and educators. The main features are presented as user stories, and then the system architecture is proposed as an improvement from a typical architecture from the personalized learning environment literature.

3.1 Teaching Methodology

The introductory programming course currently offered by the Computer Science Department of the Universidade Federal do Rio de Janeiro (DCC/UFRJ) for students of non-computing careers aims to develop the skills for building readable and modular Python programs. This methodology is based on the structured imperative approach, emphasizing problem-solving, procedural decomposition and basic skill mastery. This didactic proposal inverts the usual order of structured imperative teaching that usually starts with a complete program structure and user interaction such as *"print"*, *"input"*, and *"__main__"* statements. The proposal differential is that most of the emphasis consists of building concise code modules, leaving the user interaction mechanisms to the end of the course (when the student already mastered the basics). The syllabus is divided into 12 weeks [109] , as shown in Table 3.1. The adopted approach provides the student with an orientation towards the most abstract cognitive tasks of program design and construction since the beginning of the learning process. The detailed methodology description can be seen in Delgado *et al.* [110].

Table 3.1: Course Syllabus

1. Introduction and Functions
2. Functions
3. Data types, strings, and conditional structures
4. Variables and attribution, strings
5. String manipulation, tuples, and lists
6. Lists
7. Repetition structures - while
8. Repetition structures - for
9. Nested loops and arrays
10. Dictionary
11. User interaction and the main program
12. Modularization

3.2 User Stories

Requirements are a set of descriptions determining what a system should do and its operating restrictions. Sommerville [111] divides them into two types: system requirements and user requirements. System requirements are detailed descriptions with objective metrics that can be tested afterward, whereas user requirements are natural language statements expressing features desired by the users. In this thesis, we focus on the user requirements, understanding which functionalities our system should offer to acquire data for the proposed analyses.

A template can be used to express the desired features and create the user stories: *"As a <role>, I want <goal>, so that <benefit>"* [112, 113], answering the who?, what?, and why? questions respectively. For the Machine Teaching web-system, aim at two different types of users: the students and the educators, which can be teachers, professors, teaching assistants, pedagogues, parents and any other individual interested in understanding the dynamics of a class. Table 3.2 presents the defined user stories.

Table 3.2: User Requirements in User Stories Format

ID	**Who?**	**What?**	**Why?**
1	Student	Solve exercises	To study and learn and for the system to understand my strengths and difficulties.
2	Student	Visualize my and my peers' statistics	To better understand my strengths and difficulties and better guide my studies.
3	Educator	Provide exercises with automatic correction	To decrease my overload with repetitive manual tasks and increase my quality time with students.
4	Educator	Visualize students' submissions to a problem	To understand each student's gaps in knowledge and to diagnose misunderstandings and misconceptions.
5	Educator	Visualize learning analytics for a class	To have an overview of the class' general difficulties and grasp students' patterns.

3.3 System Architecture

A typical architecture for personalized learning environments is defined in the 2015 literature review on educational data mining [34]. Among the front-end features are an integrated development environment (IDE) for students to write, edit and execute code, a submission interface (can be embodied in the IDE or a separate part of the system), feedbacks for students' actions, and visualization schemes for teachers and researchers. Back-end usually supports saving data in some kind of storage, usually a relational database. However, part of this proposal is to integrate the proposed didactic approach described in Section 3.1. Therefore, none of the existing and revised systems in Section 2.1 could be used without modifications. To better control the desired features and captured data and avoid legacy code, it was decided to build a system from scratch using open-source Python and Javascript libraries. Fig. 3.1 illustrates an abstract view of the system, adapted from Ihantola *et al.* [34]. It shares most of the functionalities with the other systems. The main differences are the course weekly exercises defined by the professors and used by the students (Fig. 3.1(f)) and the assessment system (Fig. 3.1(b)), which takes into consideration the teaching methodology.

3.3.1 Collected Data

The collected log attributes (Fig. 3.1(c)) and their intended use within the system are:

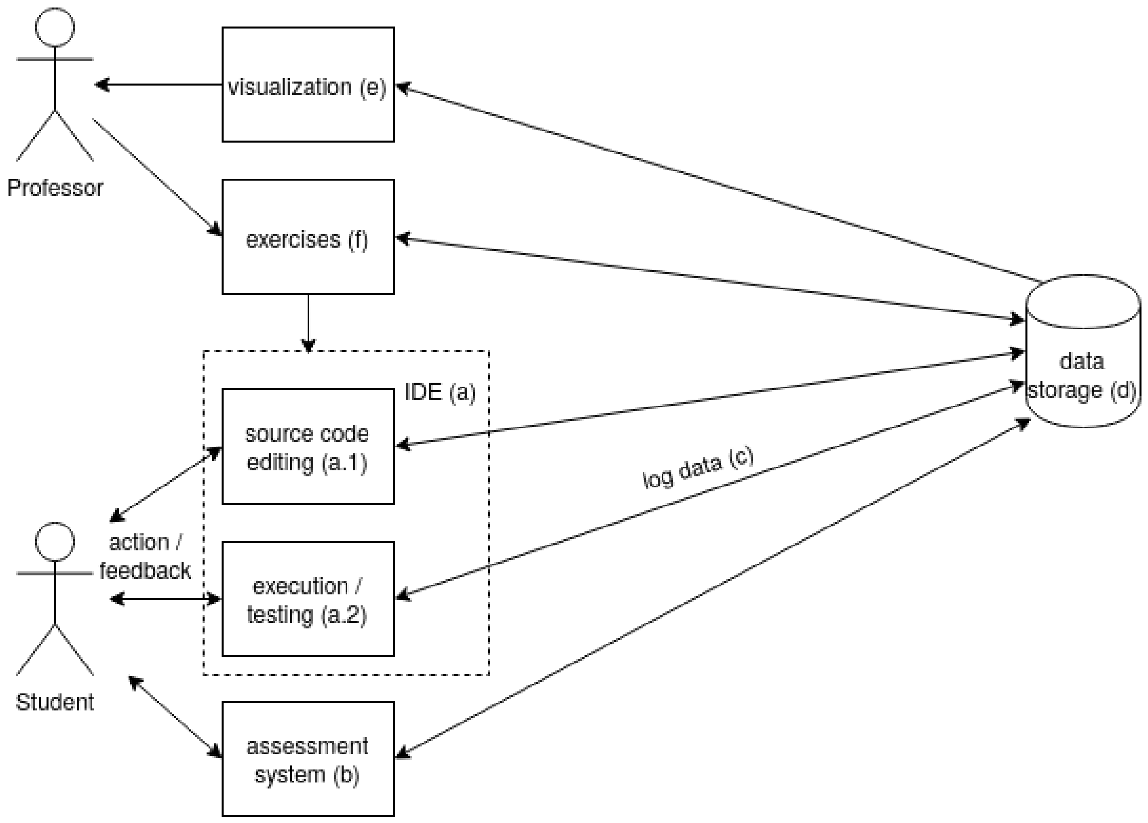


Figure 3.1: Abstract view of the system architecture. Adapted from [34].

Outcome features:

1. Percentage of test cases: The percentage of succeeded test cases. It is used to calculate the outcome and can be used to estimate student course completion (see Section 4.4).

2. Outcome: Can be either passed, failed, or skipped. Passed means that the student submitted code snippets passed all the test cases, failed means that it did not pass at least one, and skipped means that the student skipped that question and did not answer. This information can be used to understand students' difficulties.

Code features:

3. Student solution (code snippet): Run the test cases to correct the exercise, discover the concepts used by the student, discover students' code errors/exceptions.

4. Console output: Students' submission output. It may contain output values or errors. It is useful to understand if students' outputs present the expected types and format and understand in which test cases the solution pass or fails.

5. Number of lines: Derived from the code snippet feature, this feature is used to estimate code complexity and question difficulty.

6. Error: Discover students' code errors/exceptions, discover if student solution exceeded time limit constraint.

Temporal features:

7. Seconds in code, seconds in page, and seconds to begin: These three features combined are used to measure students' difficulties.

8. Timestamp: This feature can be used to analyze short-term and long-term learning.

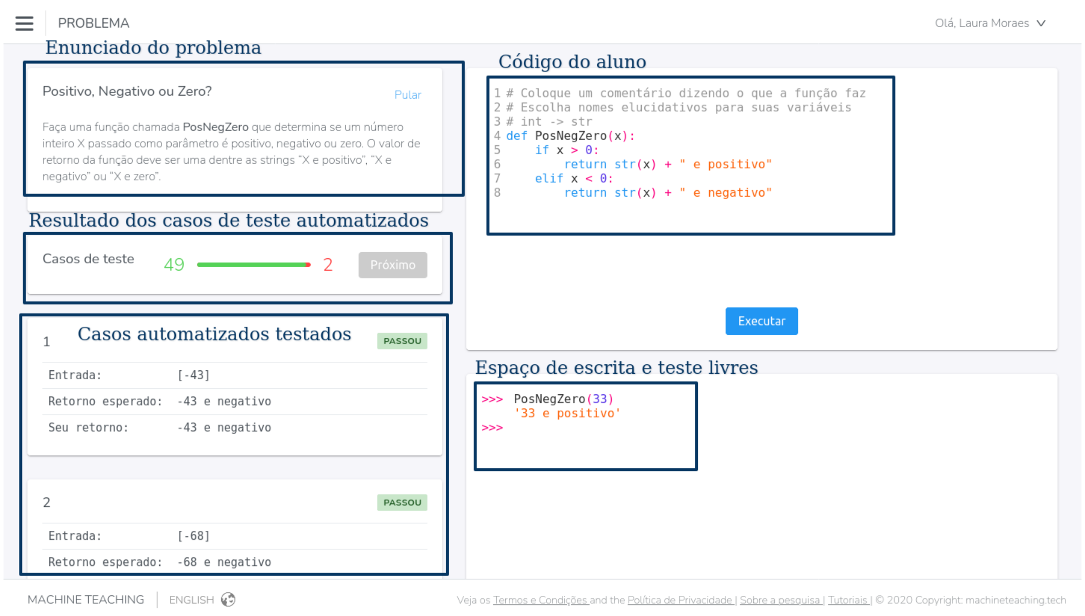
3.4 User Interaction

This section presents the main Machine Teaching user interfaces. We connect them to the user requirements and the system architecture defined in the previous sections. We used an agile methodology to develop the system [114]; therefore, not all interfaces were deployed and available at the same time. The interfaces presented in this section are already in their third version.

In the integrated development environment (Fig. 3.1(a)), the students are presented with a problem, and they should write the expected answer in a free-text coding format. For each exercise, a test case function generator was defined to correct the results (Fig. 3.1(a.2)). The students get feedback every time they submit an answer, and they can see whether they passed or failed a unit test case. If they get all of them correct, the task is considered done, and the student may move on to another problem. The system saves a state every time a student submits an answer. This IDE with the automated assessment fulfills requirements 1 and 3, and its interface can be seen in Fig. 3.2.

Students have access to a dashboard where they can visualize their completion statistics and compare themselves to their peers, as shown in Fig. 3.3. This dashboard shows how long the student took to finish the class, the number of errors, and each class' and total progress. This interface is related to requirement 2, and it is inserted in Fig. 3.1(e) module (visualization).

Requirements 4 and 5 are also included in Fig. 3.1(e) module (visualization). The Machine Teaching system provides two interfaces for the professors to understand students' submissions at class or individual levels. Fig. 3.4 presents the overall interface for a class, including students' outcome and their respective timestamps. Fig. 3.5 was directly requested by professors to be able to compare individual submissions from students. This interface shows every student submission, side-by-side,


PROBLEMA

Olá, Laura Moraes

Enunciado do problema

Positivo, Negativo ou Zero?

Pular

Faça uma função chamada **PosNegZero** que determina se um número inteiro X passado como parâmetro é positivo, negativo ou zero. O valor de retorno da função deve ser uma dentre as strings "X e positivo", "X e negativo" ou "X e zero".

Resultado dos casos de teste automatizados

Casos de teste 49 2 Próximo

Casos automatizados testados

1 PASSOU

```
Entrada:            [-43]
Retorno esperado:   -43 e negativo
Seu retorno:        -43 e negativo
```

2 PASSOU

```
Entrada:            [-68]
Retorno esperado:   -68 e negativo
```

Código do aluno

```
# Coloque um comentário dizendo o que a função faz
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    if x > 0:
        return str(x) + " e positivo"
    elif x < 0:
        return str(x) + " e negativo"
```

Executar

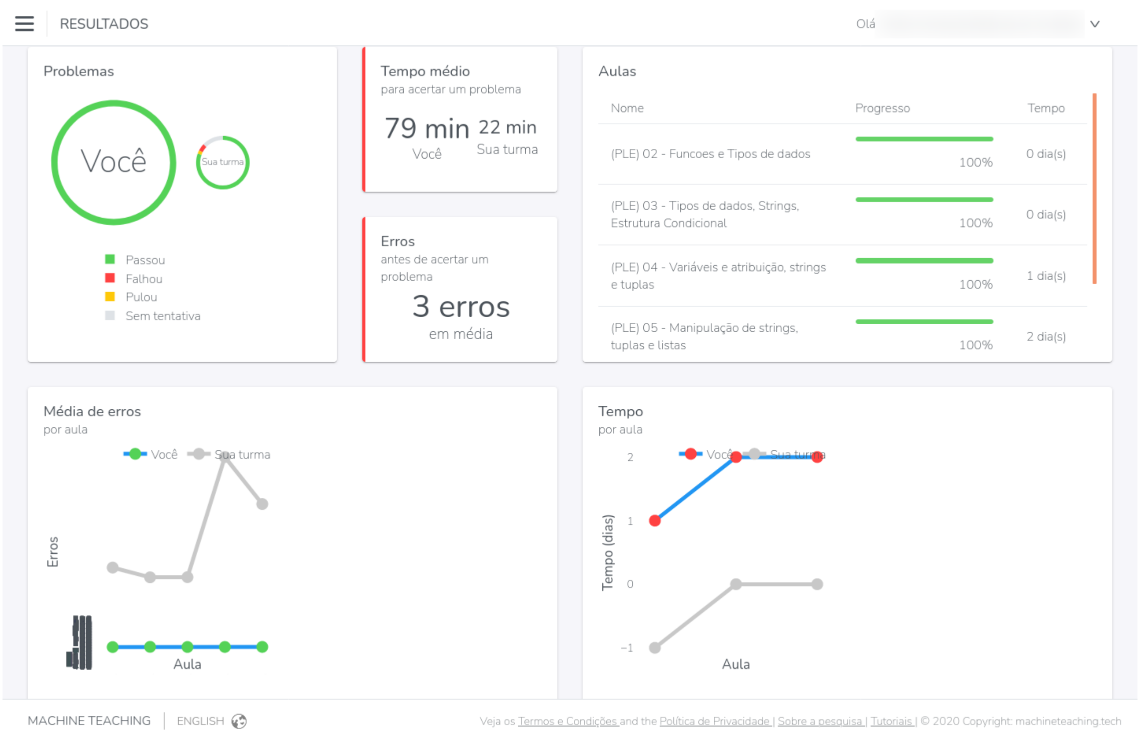
Espaço de escrita e teste livres

```
>>> PosNegZero(33)
    '33 e positivo'
>>>
```

MACHINE TEACHING | ENGLISH

Veja os Termos e Condições and the Política de Privacidade | Sobre a pesquisa | Tutoriais | © 2020 Copyright: machineteaching.tech

Figure 3.2: System IDE


RESULTADOS

Olá [redacted]

Problemas

🟩 Passou
🟥 Falhou
🟨 Pulou
⬜ Sem tentativa

Tempo médio
para acertar um problema

79 min Você 22 min Sua turma

Erros
antes de acertar um problema

3 erros
em média

Aulas

Nome	Progresso	Tempo
(PLE) 02 - Funcoes e Tipos de dados	100%	0 dia(s)
(PLE) 03 - Tipos de dados, Strings, Estrutura Condicional	100%	0 dia(s)
(PLE) 04 - Variáveis e atribuição, strings e tuplas	100%	1 dia(s)
(PLE) 05 - Manipulação de strings, tuplas e listas	100%	2 dia(s)

Média de erros
por aula

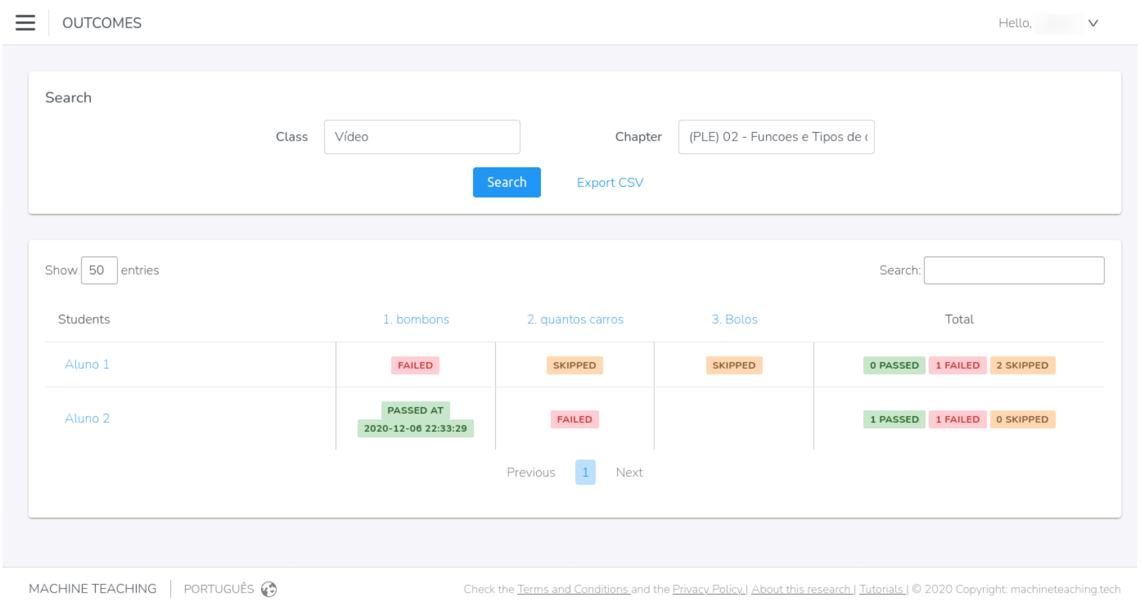
Tempo
por aula

MACHINE TEACHING | ENGLISH

Veja os Termos e Condições and the Política de Privacidade | Sobre a pesquisa | Tutoriais | © 2020 Copyright: machineteaching.tech

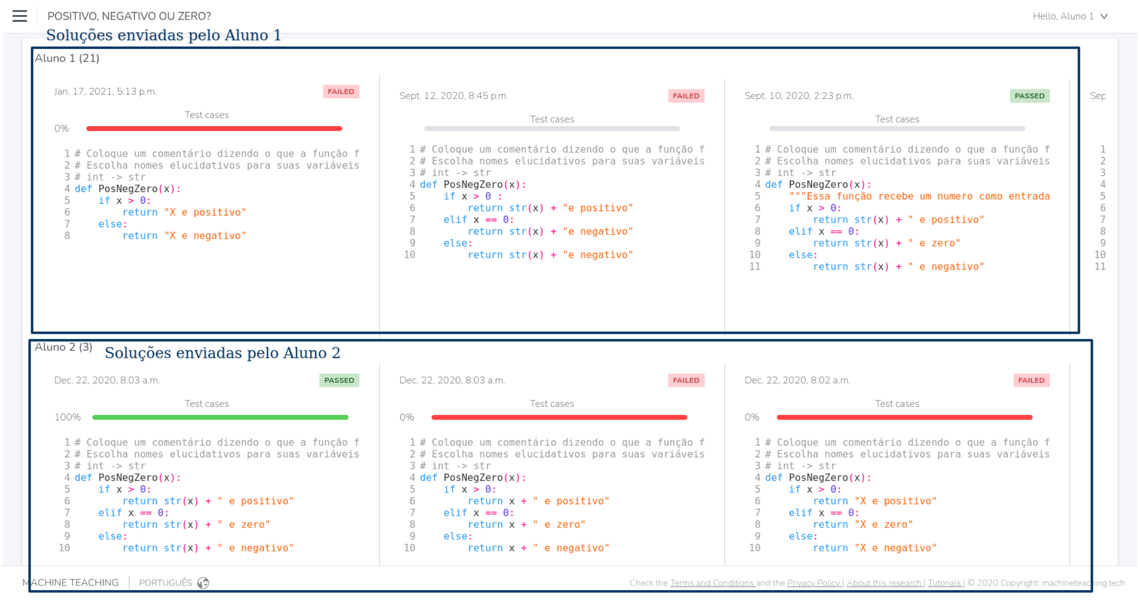
Figure 3.3: Student dashboard

with their respective timestamp and test case percentage for a single problem. In this way, the professor can spot common errors among students.


OUTCOMES
Hello,
Search
Class
Vídeo
Chapter
(PLE) 02 - Funcoes e Tipos de
Search
Export CSV
Show 50 entries
Search:
Previous
1
Next
MACHINE TEACHING
PORTUGUÊS
Check the Terms and Conditions and the Privacy Policy | About this research | Tutorials | © 2020 Copyright: machineteaching.tech

Students	1. bombons	2. quantos carros	3. Bolos	Total
Aluno 1	FAILED	SKIPPED	SKIPPED	0 PASSED 1 FAILED 2 SKIPPED
Aluno 2	PASSED AT 2020-12-06 22:33:29	FAILED		1 PASSED 1 FAILED 0 SKIPPED

Figure 3.4: Overview interface for a class. Detailed information on students' outcomes, with the respective timestamp and overall performance.


POSITIVO, NEGATIVO OU ZERO?
Hello, Aluno 1
Soluções enviadas pelo Aluno 1
Aluno 1 (21)
Soluções enviadas pelo Aluno 2
Aluno 2 (3)
Test cases
MACHINE TEACHING
PORTUGUÊS
Check the Terms and Conditions and the Privacy Policy | About this research | Tutorials | © 2020 Copyright: machineteaching.tech

Jan. 17, 2021, 5:13 p.m. FAILED 0%

```
# Coloque um comentário dizendo o que a função f
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    if x > 0:
        return "X e positivo"
    else:
        return "X e negativo"
```

Sept. 12, 2020, 8:45 p.m. FAILED

```
# Coloque um comentário dizendo o que a função f
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    if x > 0 :
        return str(x) + "e positivo"
    elif x == 0:
        return str(x) + "e negativo"
    else:
        return str(x) + "e negativo"
```

Sept. 10, 2020, 2:23 p.m. PASSED

```
# Coloque um comentário dizendo o que a função f
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    """Essa função recebe um numero como entrada
    if x > 0:
        return str(x) + " e positivo"
    elif x == 0:
        return str(x) + " e zero"
    else:
        return str(x) + " e negativo"
```

Dec. 22, 2020, 8:03 a.m. PASSED 100%

```
# Coloque um comentário dizendo o que a função f
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    if x > 0:
        return str(x) + " e positivo"
    elif x == 0:
        return str(x) + " e zero"
    else:
        return str(x) + " e negativo"
```

Dec. 22, 2020, 8:03 a.m. FAILED 0%

```
# Coloque um comentário dizendo o que a função f
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    if x > 0:
        return x + " e positivo"
    elif x == 0:
        return x + " e zero"
    else:
        return x + " e negativo"
```

Dec. 22, 2020, 8:02 a.m. FAILED 0%

```
# Coloque um comentário dizendo o que a função f
# Escolha nomes elucidativos para suas variáveis
# int -> str
def PosNegZero(x):
    if x > 0:
        return "X e positivo"
    elif x == 0:
        return "X e zero"
    else:
        return "X e negativo"
```

Figure 3.5: Single problem view. Students' submissions are presented side-by-side for easy viewing of common errors.

Chapter 4

Database and Exploratory Analysis

In this chapter, we present the data acquisition methodology used to generate the datasets [30]. We used two complementary approaches: the system would be used before finals or at the beginning of the next semester for revision purposes, called the *revision dataset*, and the system would be used throughout a whole course (one semester) with weekly exercises, accompanying students from the 3rd to the 10th week of the course, called *semester dataset*. We also perform an exploratory data analysis comparing the student behavior in both datasets. Finally, we reproduce an early dropout prediction model, revisiting exploratory research results from the literature.

4.1 Revision Dataset

The first approach uses 48 CS1 problems crawled from four Python web tutorials: Practice Python [115], Python School [116], Python Programming Exercises [117], and W3Resource [118] that provided both solutions and exercise statements. Since the sources do not have label topics or follow a course curriculum with structured syllabus topics, we work in an unsupervised environment. We crawled 54 exercises for the training set. The code snippets are functions with an average of 9 lines/code. Students' responses to these exercises were collected using the Machine Teaching web-system, presented in Chapter 3, in 10 different classes at the Introduction to Programming courses at the Universidade Federal do Rio de Janeiro (UFRJ) over two course periods. Students were assigned to two different strategies: either the system showed random problems or they would follow a predefined path. The system was introduced at the end of the semester (before their finals exams) or the beginning of the next semester.

In total, there are 3,632 records from 192 students with an average of 18.4 attempts in 4.4 problems in an imbalanced dataset: it is 764 (21%) successful attempts versus 2,868 (79%) failed attempts. It means that, on average, each student attempts a problem 4 times before succeeding in the fifth.

4.2 Semester Dataset

Our second approach accompanied the students throughout a whole semester in four different Introduction to Programming courses (they had the same syllabus but different professors). Every week, an exercise list concerning the subject given in class was available in the system. They had one week as a deadline to finish them, and their performance on these lists composed part of their final grade. In total, there are 65 different problems with their respective solutions. Like the revision dataset, the code snippets are functions with an average of 7 lines/code.

This dataset is 7.5 times larger in the number of attempts than the previous one, containing 27,491 attempt records. However, since it accompanied the same students for an entire semester, the number of students is smaller: 181 different students. This dataset has a slightly higher success rate than the previous one, although it is still very imbalanced. It contains 6,849 (24.91%) success attempts against 20,642 (75.09%) unsuccessful ones. This difference can be explained by the fact that each exercises' weekly set mostly covers what was seen in class in the same week, so students did not have much time to forget the subject [93, 96, 119–121], in contrast to the review exercises that were done before finals or at the beginning of the next semester.

4.3 Exploratory Analysis

Some simple statistics for both datasets are shown in Tables 4.1 and 4.2. Fig. 4.1 is a histogram showing the distribution of success and failures per problem on the revision dataset. Similar behavior is found in the second dataset. Both success attempt distributions have smaller variance and smaller mean than the corresponding fail attempt distributions. This distribution is the expected designed system behavior since the students are given several tries to submit a correct response before moving on to the next problem.

Another interesting difference between the datasets is the maximum number of successful attempts per student. Whereas in the revision dataset, it is lower than the total number of questions, indicating that not all the questions were answered, in the semester dataset, it is two times the total number of available questions, indicating that some students redid the exercises even though they had already succeeded at it. This information can be used to measure if the students are using the system and the exercises to study and review the content, for example. Our survey concerning how the students perceived and used the system will be presented in Section 6.4.

Table 4.1: Revision Dataset Statistics

	Revision			
	Avg	Median	Min	Max
attempts per question	75.67	55	10	304
attempts per student	18.44	11	2	266
successful attempts per student	4.34	3	1	36
different students per question	18.17	14	4	71
different questions per student	4.43	3	1	44

Table 4.2: Semester Dataset Statistics

	Semester			
	Avg	Median	Min	Max
attempts per question	422.94	349	85	1291
attempts per student	151.88	114	2	1002
successful attempts per student	38.26	32	1	159
different students per question	87.22	87	39	145
different questions per student	31.32	31	1	65

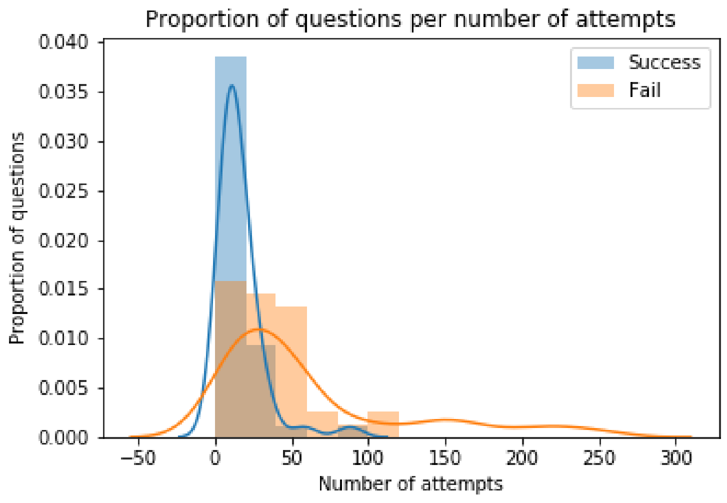


Figure 4.1: Distribution of questions' success and failed attempts

Fig. 4.2 shows the number of students per number of questions histogram on the semester dataset. We can notice that only 15 out of the 181 (8%) students attempted more than 59 out of the 65 (92%) different exercises. The exercise distribution is relatively flat (Pearson correlation = 0.28). If we divide it into thirds, approximately one-third of the class (64 students, 35%) attempted only one-third of the exercises, one third (60 students, 33%) attempted two-thirds of the exercises and the last third (57 students, 31%) attempted between 45 and 65 exercises. When provided in real-time, the professors can use this information to find students who are having difficulties finishing exercises and provide personalized assistance. Historically, courses that adopt the structured imperative approach have a high percentage of failure and dropouts [110]. In this course, from the 181 students that participated in the study, 108 (60%) did the exercises from the last two weeks and 125 (69%) when we consider the last three weeks of exercises within the system. Notice that this is not properly a dropout rate since the course already had two more weeks (with user input subject that is not covered by the Machine Teaching exercises) and the students could do the exams even though they did not attempt every list. However, Fig. 4.3 depicts a decreasing trend in the number of different students that submitted exercises per week (Pearson correlation $= -0.98$ and p-value < 0.001).

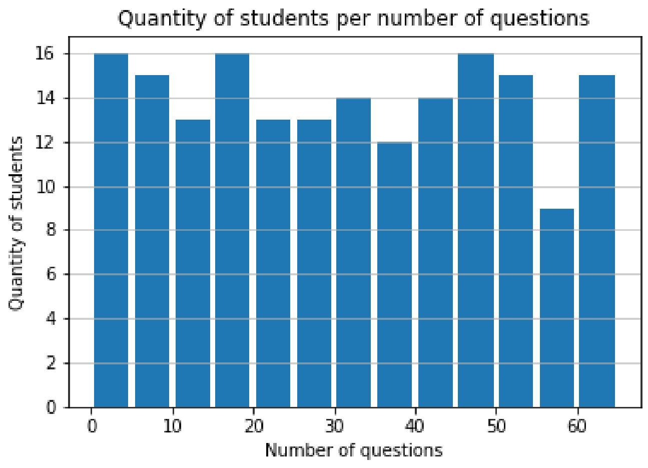


Figure 4.2: Distribution of students per number of questions

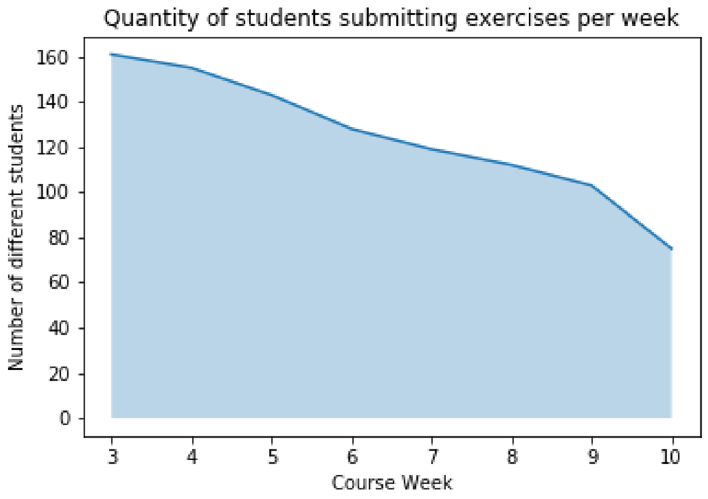


Figure 4.3: Active students per week

4.4 Early Dropout Prediction

Given the decreasing trend of active students during the semester, we decided to investigate if it was feasible to create a prediction model to alert professors at the beginning of the semesters. To produce this model, we used data from the first two weeks of exercise to predict if the students will submit at least half of the exercises from the last two weeks as it was done in previous studies [54]. If a student is likely to withdraw from the course, a professor may perform a personalized intervention.

Similar to Pereira *et al.* [54], our dataset is small to medium-sized and collected in a face-to-face university setting. We aim at reproducing and verifying these literature results in our dataset.

For this task, we engineered features based on literature findings as reported in Section 2.2. In literature, we found models that aggregated all the features independently of the week. We propose separating these features per week. We created models using both approaches and compared the results. Our final list of features is shown in Table 4.3.

Table 4.3: Previously Published Relevant Features to Predict Course Completion Included in this Experiment

Feature	**Source(s)**
Number of attempts (or submission clicks)	[54, 57]
Success rate (or assignment completion rate)	[21, 54]
Time doing the exercise, measured by three different variables: seconds in page, seconds to begin, and seconds in code	[56]
Number of unique days submitting exercises	[57]

The Machine Teaching system does not have a forum and does not store access frequency. Therefore, these two features could not be verified in this experiment. Previous research on dropout prediction has achieved good results using classification trees, providing interpretability [54]. We used the CART [122] algorithm with Gini Index as the splitting criterion and 10-fold cross-validation, with 80% of the observations in the train set and the remaining 20% in the test set. We would like to optimize the model for the F1 metric: balancing sensitivity and specificity for the *"not likely to submit exercises"* class. The dataset is slightly imbalanced: it has 60% of *"not likely to submit exercises"* against 40% of *"likely to submit exercises"*. We tuned the model for tree max depth, varying it from 2 to 4. The best results were achieved using max depth = 3. The resulting tree is shown in Fig. 4.4. Table 4.4 compares the CART algorithm using weekly features (CART+Weekly) and using the aggregated features (CART+Total). We also added as the baseline the average success rate from week 3, which was the most informative criterion, and the adjusted value of success rate from week 3, using it as 0.39 instead of the average as given by the decision tree. Using weekly features, the CART algorithm performed best on the desired metrics (F1 and Recall), which values the correct classification of *"not likely to submit exercises"* students.

Table 4.4: Dropout Prediction Results

	CART+Weekly	CART+Total	Avg. Week 3 Success Rate	Avg. Adj. Week 3 Success Rate
F1	**0.81**	0.7	0.58	0.67
Recall	**0.88**	0.7	0.58	0.58
Acc.	0.72	0.56	0.72	**0.81**

We can also extract from Fig. 4.4, where true stands for *"likely to submit exercises"* and false for *"not likely to submit exercises"*, the most important variables to predict student withdrawal. The weekly success rates are found to be the best predictors in this dataset, corroborating the findings from Pereira *et al.* [54] and Damasceno *et al.* [21]. They are followed by the number of attempts and the number of unique days submitting exercises. This result also validates Al-Shabandar *et al.*'s findings [57], which encountered a positive correlation between these features and course completion. The time features did not appear as relevant ones. It could be because we are measuring the time per session (it restarts counting if the student refreshes the page) and not considering a continuous calculation of time.

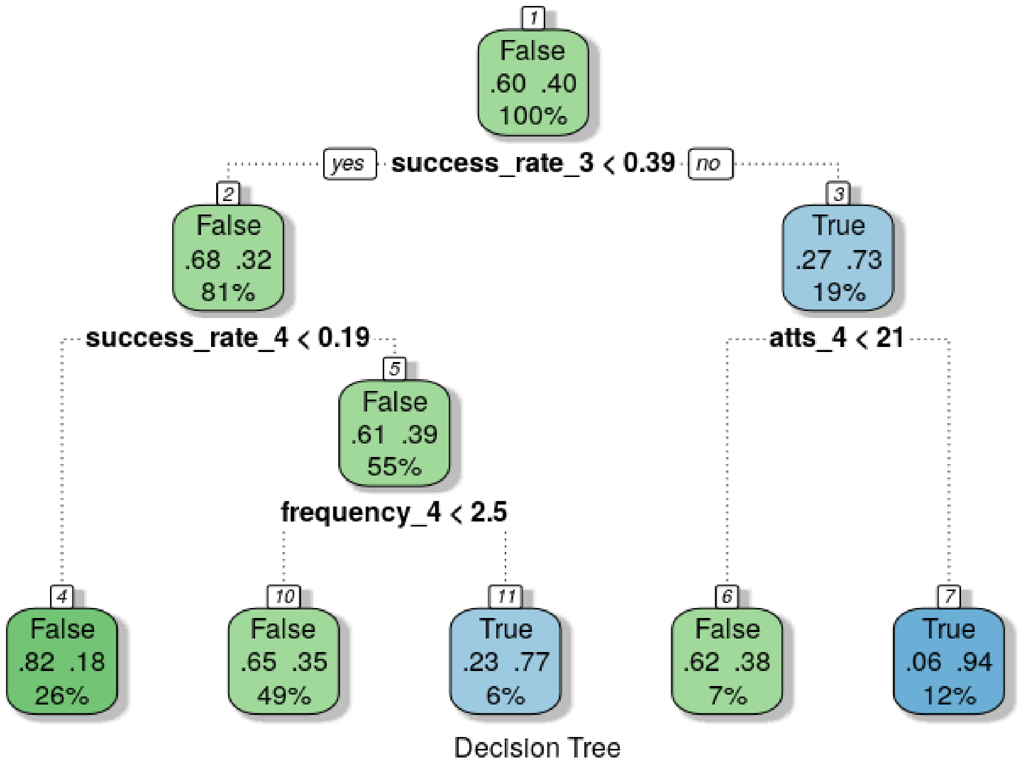


Figure 4.4: Decision tree for dropout prediction

Chapter 5

Mapping Methodology

In this chapter, we present our research methodology and the methodology used to map questions to concepts. Our goal is to provide automated tools to build a Q-matrix: a matrix inspired in the cognitive diagnostic test used in psychometrics where the rows are questions, and the columns are pieces of knowledge (skills or concepts). The cell values are closer to 1 where the corresponding knowledge is necessary to solve the respective question and closer to 0, otherwise [92, 101, 102]. We present two methods to extract in an unsupervised way the underlying concepts. Both methods are validated with an unseen test set and by experts. The first method uses topic modeling techniques to cluster code written solutions and obtain the terms that characterize each cluster [31]. The second method uses student performance data to extract concepts. Besides retrieving a matrix containing the questions and concept relationships, it also outputs a student knowledge model. Finally, we provide a way to compare both extracted concepts with the ones provided by experts.

5.1 Research Method

In this thesis, we adopted a mixed-methods research approach, which seeks to use both qualitative and quantitative methods to provide narrative and precision to the results [45]. One of its advantages is to allow the researcher to pick different methods for different research questions [45]. Malmi *et al.* [123] created a classification system for research in computing education, which we used to categorize this research. Table 5.1 provides an overview of the research classification used in this thesis, based on Sorva [45] and Malmi *et al.* [123].

Table 5.1: Overview of the research classification according to Malmi *et al.* [123].

Sec.	**Research question**	**Research purpose**	**Research framework**	**Data source**	**Form of analysis**
5.2 6.1	How can semantic relationships be extracted and structured from code?	formulative method	construc-tive	code snippets from CS1 problems	exploratory statistical analysis
5.2 6.1	How can humans read, interpret, and use the extracted relationships?	formulative method	construc-tive	code snippets from CS1 problems	qualitative analysis
5.4 6.3	How similar are the factorized latent variables from the concepts annotated by the professors?	quantitative evaluation	survey	annotated dataset and results from Sections 5.2 and 5.3	exploratory statistical analysis
6.4	How do students perceive the system?	qualitative evaluation	survey	online questionnaire	descriptive statistics, qualitative analysis

5.2 Q-Matrix Discovery from Code

In this methodology, we create a code-clustering pipeline to build a Q-matrix using solution code snippets from the revision dataset as input. We validate the results with the semester solutions acting as an external dataset. The code-clustering pipeline takes as input Python code snippets, which are semi-structured text documents. By using topic modeling techniques, the pipeline outputs an underlying structure within the semi-structured corpus. It contains the topics present in the code snippets and the most relevant words that characterize them. This method is based on the assumption that code snippets with similar CS1 concepts share identical terms. Therefore, based on this assumption, the extracted topic underlying structure can be interpreted as CS1 concepts or groups of CS1 concepts present in the code snippets.

The code-clustering pipeline starts by transforming the original data to the proper format expected by the topic modeling methods. We augmented the data and constructed a matrix D (the document-term matrix) where each element D_{ij} contains the weight of term w_j in document d_i. Then, using topic modeling, we calculated the relevance of each topic t_k for each document d_i and the relevance of each term w_j for each topic t_k. Finally, we applied a grid search and topic coherence to choose the best models and evaluate the external corpus results. In the topic filter and selection phase, we also processed the resulting topics by merging similar or removing topics with few documents. These results are presented in Section 6.1, while Fig. 5.1 illustrates the code-clustering pipeline. External evaluation is not depicted in this overview.

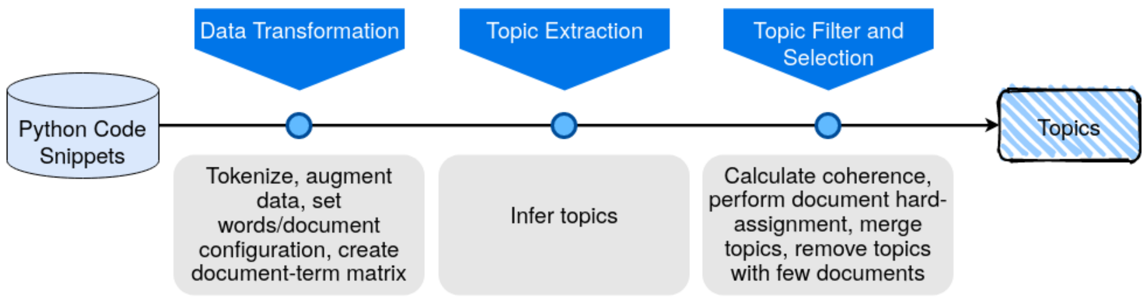

Data Transformation
Topic Extraction
Topic Filter and Selection
Python Code Snippets
Topics
Tokenize, augment data, set words/document configuration, create document-term matrix
Infer topics
Calculate coherence, perform document hard-assignment, merge topics, remove topics with few documents

Figure 5.1: Code-clustering pipeline overview. Adapted from Abolhassani *et al.* [80].

5.2.1 Data Transformation

In this application, the CS1 code solutions written in Python are considered documents. The document-term matrix creation process starts by splitting each code snippet into words. The first proposed tokenizer includes only split word tokens. Henceforth this tokenizer will be referred to as the *standard tokenizer*. As stated in the related work section: 1) the LDA usually does not perform well on short texts and 2) augmenting the corpus by adding the text's structure on semi-supervised documents demonstrated improved results [79, 80]. We propose a new tokenizer to augment the standard tokenizer with extra features and refer to it as the *augmented tokenizer*. The augmented tokenizer parses the code and makes special annotations by adding extra features if the token is a number, an array (or a list), a dictionary, a string, a logical (or arithmetic) operator, a class method, or indentation. The word itself is added to the document-term matrix if the token is a reserved word. Besides adding single tokens, this tokenizer also considers bigrams and trigrams. Although the document-term matrix does not consider the terms' order, this can be enforced by adding n-grams as a matrix feature. For example, the code snippet in Fig. 5.2(a) is first transformed to its augmented version (see Fig. 5.2(b)). Then, every single word, including the bigrams and the trigrams, are added as tokens to the document-term matrix. Table 5.2 presents some examples of the document-term matrix terms, and, in total, the document is tokenized into 75 terms.

After the document-term matrix creation, we applied some transformations to enhance document representation and decrease matrix sparsity. First, we removed tokens with document frequency below a fixed threshold to perform feature selection. This threshold was determined using a hyperparameter grid search ranging from 5% to 50% with a 5% step. Second, we decided how to count a token frequency: either the token is counted once per document (binary appearance) or every time it appears. Finally, some tokens may be more important than others. For example,

```
def light(switchA, switchB):
  if switchA == 1 and switchB == 1:
    return True
  else:
    return False
```

(a) Original code snippet

```
def light switchA switchB is_block
 is_indent if switchA is_op_logic is_number is_op_logic
             switchB is_op_logic is_number is_block
    is_indent return True
  is_dedent else is_block
    is_indent return False
  is_dedent
is_dedent
```

(b) Augmented code snippet

Figure 5.2: A code snippet and its augmented version. The augmented version will be processed in the augmented tokenizer, whereas the regular one will be processed in the standard tokenizer.

Table 5.2: Example of Document-Term Matrix. A Set of Terms from the Complete Document-Term Matrix after Augmenting and Tokenizing the Document from Fig. 5.2

Terms	Count
is_block	3
is_indent	3
is_number	2
is_op_logic	3
if	1
is_op_logic is_number	2
is_block is_indent	3
is_op_logic is_number is_op_logic	3

term frequency by inverse document frequency (TF-IDF) [124, 125] recalculates the tokens' weights by balancing two factors: 1) a term that occurs in many documents should not be as important as a more exclusive term, since it does not characterize documents well. Also, 2) a term that appears in a small number of documents may only be particular to those documents and not enough to distinguish a topic. Yan *et al.* [126] propose another way of recalculating the terms' weights. Their method (called NCut) comes from the normalized cut problem on term affinity graphs. This weighting scheme modifies terms' counts based on terms cooccurrence and not on document frequency. Their experiments show NMF's performance increase on short text clustering using the NCut weighting scheme.

5.2.2 Topic Extraction

As stated earlier, after document processing, a document-term matrix D is generated. The matrix rows represent points in an $\mathbb{R}^n$ feature space, where n is the total number of terms, and each term w_j corresponds to a dimension. It becomes a classical clustering problem where we expect similar documents to be in surrounding regions in space. So, clustering algorithms like K-means, hierarchical clustering, and nearest neighbors are applicable here. However, for topic modeling tasks, algorithms like the NMF [26, 127] and the LDA [27] are effective since they interpret terms' counts as a set of visible variables generated from a set of hidden variables (topics) [128, 129]. Accordingly, the documents can be modeled as a distribution of topics and topics as a distribution of terms. Fig. 5.3 presents a conceptual visualization of how the original document term-matrix can be factorized into two matrices. Fig. 5.4 shows an illustrative example of the expected matrices. We used two topic modeling techniques:

1. NMF: a matrix factorization technique with a particular property of only allowing non-negative values in its entries, which is well-suited for human-interpretability [26].

2. LDA: a generative probabilistic model that describes how to create documents in a collection. Once you have a dataset, a group of already written documents, we find the distributions that created these documents. The LDA algorithm tries to backtrack this probabilistic model to find a set of topics that are likely to have generated the dataset [28].

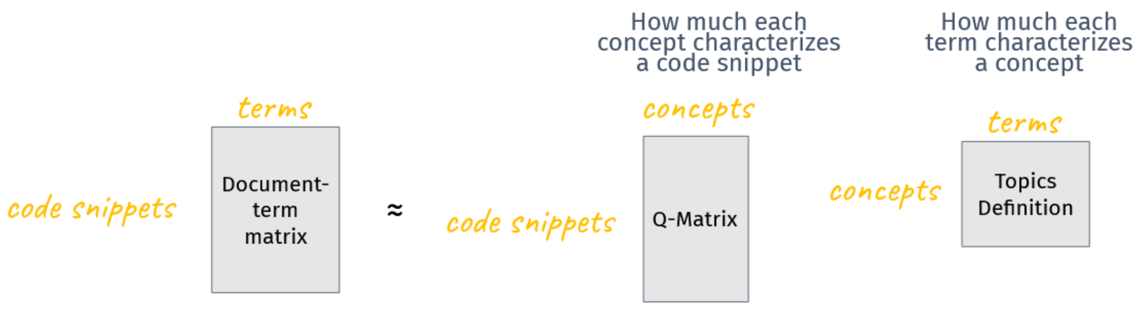


Figure 5.3: Matrix factorization conceptual visualization.

$$\begin{array}{l|c|c|c} & for & if & append \\ \hline Code\ Snippet\ 1 & 2 & 0 & 1 \\ Code\ Snippet\ 2 & 1 & 1 & 0 \\ Code\ Snippet\ 3 & 1 & 2 & 0 \end{array} \approx \begin{array}{l|c|c} & loop & conditional \\ \hline Code\ Snippet\ 1 & 1 & 0 \\ Code\ Snippet\ 2 & 0.5 & 0.5 \\ Code\ Snippet\ 3 & 0.25 & 0.75 \end{array} \begin{array}{l|c|c|c} & for & if & append \\ \hline loop & 0.75 & 0 & 0.25 \\ conditional & 0 & 1 & 0 \end{array}$$

Figure 5.4: Matrix factorization illustrative example.

5.2.3 Topic Filter and External Evaluation

Given several document-term matrix creation options and two different topic modeling methods, we need to find the best set of hyperparameters. There are strategies in the literature to find a near-optimal set of models' hyperparameters, such as manual search, grid search, and random search [130]. Although random search demonstrates promising results in general machine learning tasks [130], Chuang *et al.* [131] and Wang and Blei [132] results were competitive using grid search in topic modeling tasks. We chose to use a grid search approach. There was a prior manual stage to define the regions in which grid search would act. Since the dataset is not large and the number of hyperparameters to try is not extensive, it is efficient to run an exhaustive search combining hyperparameters. In total, there are 1680 possible combinations: 10 minimum document frequencies (ranging from 5% to 50% with 5% step increment), 2 binary appearance options, 3 token weights (counts) transformation possibilities (none, TF-IDF, and NCut), 2 clustering methods (LDA and NMF), and 14 number of clusters (i.e., $10 \times 2 \times 3 \times 2 \times 14 = 1680$). The grid search was set to search between 2 and 15 clusters (the upper bound is based on the number of concepts from Table 5.3 in Section 5.2.4).

To determine whether topics are well-defined, we can use topic coherence and pointwise mutual information (PMI) metrics, which correlated well with human-interpretability [133–135]. As explained in Section 5.2.2 (topic extraction), when using NMF or LDA, each topic is mapped to a list of top-N words that best define the topic. Topic coherence calculates the ratio between the cooccurrence of these top-N words and their total occurrence. The assumption is that the words that best characterize a topic often appear together if a topic is well-defined. We applied two types of topic coherence metrics: UCI [136] and UMass [133]. The UCI metric based on PMI is calculated using an external validation source. The PMI can be substituted using normalized PMI (NPMI) to better correlate with humans' ratings [134]. The UMass metric uses the conditional probability of one word occurring given that one other high-ranked word occurred and can be measured using the modeled corpus, without depending on an external reference corpus. We used the UMass coherence to choose the best models since it is an internal validation metric (it only evaluates the clustered data). To assess the models, we used an external dataset with the UCI NPMI metric.

Defining $P(w_i)$ as the probability of the term w_i occurring and $P(w_i, w_j)$ as the probability of terms w_i and w_j cooccurring, we calculated the coherence for a single topic t_k using (5.1), (5.2), and (5.3). The topic coherence for a single topic was calculated using top-5 and top-10 terms. After calculating each topic's coherence in a single hyperparameter combination, this combination's coherence was reported as the median of all topic coherence.

$$C_{UMass}(W) = \sum_{i=1}^{N}\sum_{j=1}^{N} log\frac{P(w_i, w_j) + \epsilon}{P(w_i)} \tag{5.1}$$

$$C_{UCI}(W) = \sum_{i=1}^{N}\sum_{j=1}^{N} NPMI(w_i, w_j) \tag{5.2}$$

$$NPMI(W) = \frac{log\frac{P(w_i,w_j)+\epsilon}{P(w_i)P(w_j)}}{-log(P(w_i, w_j) + \epsilon)} \tag{5.3}$$

where $W = (w_1, w_2, ..., w_N)$are the top-N terms for calculating the coherence. An ϵ value of 0.01 was used to avoid taking a zero logarithm.

We performed hard-assignment to cluster documents by topic by assigning each document to the topic with the most relevance (weight) in the document-topic matrix. The hard-assignment was achieved with minimal loss of information when a topic strongly characterized a document. In addition to assigning documents to topic clusters, the set of features/terms that best characterize each cluster/topic were extracted for further analysis.

As a final step, we also merged semantically similar topics and removed those not containing relevant information.

5.2.4 Topics Contextualization

To relate concepts and topics, we first defined the most commonly seen concepts in CS1 exercises. The following four references were used to create a list of concepts commonly used in CS1 courses:

1. Computer Science Curricula 2013 [137]: a document jointly built by the Association of Computer Machinery and the IEEE Computer Society. The document recommends curricular guidelines for computer science education, which we used as the main concept list. We used the papers in items 2, 3, and 4 to improve it.

2. Exploring programming assessment instruments: A classification scheme for

examination questions [58]: creates a classification scheme characterizing exam questions by their concept areas, question style, and skills a student needs to solve them. We used the list of the proposed concepts as a second source to enhance the main list.

3. Reviewing CS1 exam question content [59]: this paper used nine experienced CS1 instructors to review the concepts required in examinations from different North American institutions. It created a list of concepts using the intersection of three other experiments [138–140].

4. Identifying challenging CS1 concepts in a large problem dataset [141]: this paper identifies the most challenging CS1 concepts for students based on 266,852 web-based code-writing student responses. Their concept list is based on the course structure from the CS1 class where they experimented.

Table 5.3 shows the final list of consolidated concepts.

Table 5.3: List of CS1 Concepts

1. Syntax	6. Logic	11. Conditional
2. Assignment	7. Data type: string	12. Loop
3. Data type: number	8. Data type: array	13. Nested loop
4. Data type: boolean	9. Data type: tuple	14. Function
5. Math	10. Data type: dict	15. Recursion

Then, to interpret the meaning of the topics, we asked 14 professors to perform three tasks. The professors (with 2 to 20 years of teaching experience) teach CS1 or other programming-related subjects.

- **Theme identification:** we present some code snippets belonging to the topic and found essential tokens for each topic. The professors were asked to label each topic with free-text descriptions. We tokenized the descriptions and counted the terms. We also created the topic titles based on the terms that appeared more frequently in the descriptions.

- **Concept identification:** each professor was asked to associate up to three concepts (from the 15 available in Table 5.3) to 15 randomly assigned code snippets. Then, we aggregated the concepts related to each code to the associated topics. In this way, it is possible to understand the most relevant concepts in each topic and calculate how well the concepts inside a cluster agree to them. A similar approach was used by Lan *et al.* [22] to contextualize concept clusters.

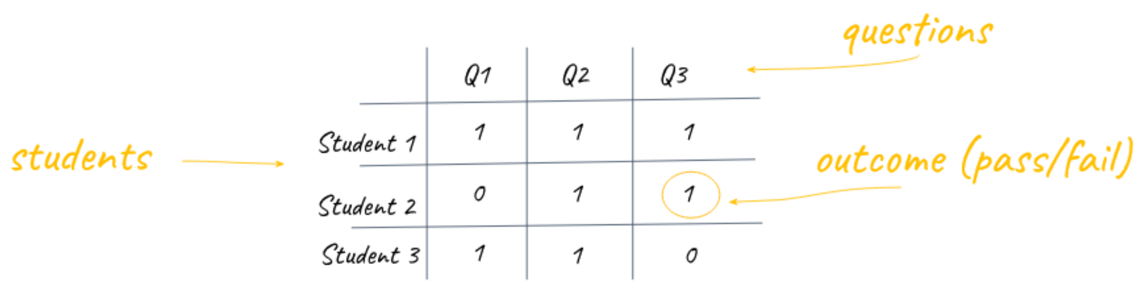
- **Intruder identification:** Chang *et al.* [142] proposed quantitative methods to analyze the interpretability of the latent space created using topic modeling techniques. Their paper proposed a method to identify an intruder topic given a document. We adapted this method because we hard-assigned each document to a single topic. For our analysis, the professors were asked to identify the intruder document given a topic. This approach has been used by Mahmoud and Bradshaw [143] to assess the quality of their topic modeling approach on Java-based systems.

5.3 Q-Matrix Discovery from Student Performance

Discovering the underlying concepts of questions by grouping their solutions using topic modeling techniques may provide insightful interpretable concepts. However, they do not consider each student; they provide a specialists' expected concepts view. An approach to include the student perspective is to examine students' performance and outcomes (if the student did or did not get the question correctly) instead of looking at the written solutions. The presented methods are widely used to predict student performance. In this thesis, we are using these methods to verify if the obtained Q-matrices (a matrix that related latent variables and questions) agree with specialists' annotations.

5.3.1 Factorization methods to predict student performance

In this section, we describe two state-of-the-art methods to extract Q-matrices based on student performance (if the student did or did not get the question correctly): 1) Non-negative matrix factorization (NMF) [25, 105], 2) Feedback-driven tensor factorization (FDTF) [108]. NMF was already introduced in Section 5.2.2 as a topic modeling method. As it is a matrix factorization method, it can be used for different tasks if modeling the input differently. Fig. 5.5 presents an example of how to model the input based on student performance data, whereas Fig. 5.6 shows how the factorization occurs when using this modeling approach. To predict student performance, we model its input as the outcome of the first student response at each question or an average of all students' responses. In this way, when the factorization is performed, two matrices are obtained: the Q-matrix and a latent students skill matrix (also called student knowledge model). When modeling the student performance data using tensors instead of a matrix, we gain an extra dimension: each response (or attempt). There is no need to choose between the first response or an average: the extra dimension corresponds to each student response. In this thesis, we use the FDTF algorithm for tensor factorization. Fig. 5.7 depicts how the tensor factorization decomposes the student performance tensor into a student knowledge tensor and a Q-matrix.



	Q1	Q2	Q3
Student 1	1	1	1
Student 2	0	1	1
Student 3	1	1	0

questions
students
outcome (pass/fail)

Figure 5.5: Input based on performance data.

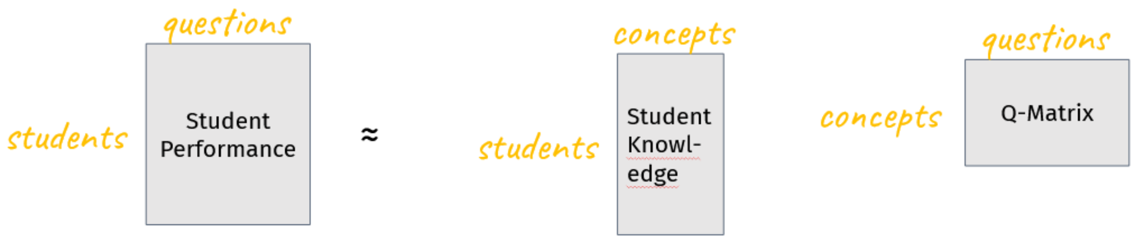


Figure 5.6: Student matrix factorization conceptual visualization.

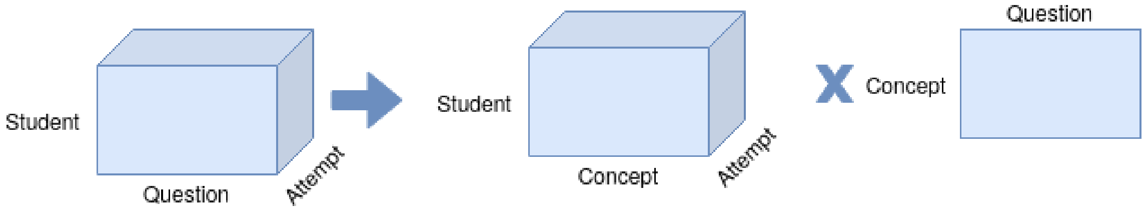


Figure 5.7: Tensor factorization. Both matrix and tensor factorizations produce a matrix called Q-matrix (Concepts x Questions). FDTF has one more dimension than NMF: attempt.

Non-negative Matrix Factorization

In the NMF model, we formulate the problem as:

$$\min_{T \in \mathbb{R}^{p \times r}, Q \in \mathbb{R}^{r \times n}} \|X - TQ\|_F^2 + \lambda\left(\sum_{i=1}^{t} \|T\|^2 + \|Q\|^2\right) \quad \text{s.t.} \quad T \geq 0 \text{ and } Q \geq 0 \tag{5.4}$$

where X represents the students' performance matrix, T represents the students' knowledge matrix, Q represents the Q-matrix and λ controls the amount of regularization applied to the model.

The student performance matrix containing students' outcomes may be filled with a summarized result (average outcome, for example) or a single result (outcome at the nth attempt, for example). This is set as one of the model's hyperparameters.

Feedback-Driven Tensor Factorization

FDTF is a tensor factorization method that assumes that student knowledge would be monotonically increasing while answering questions. The objective function of FDTF is formulated as:

$$\min_{T_t \in \mathbb{R}^{p \times r}, Q \in \mathbb{R}^{r \times n}} \sum_{i=1}^{t} \|X_t - T_t Q\|^2 + \lambda(\sum_{i=1}^{t} \|T_i\|^2 + \|Q\|^2) \tag{5.5}$$

$$T_t = 2T_{t-1} + \frac{2(1 - T_{t-1})}{1 + exp(-\mu X_t Q')} - 1 \tag{5.6}$$

where X_t is the students' performance matrix at time t, T_t is students' knowledge matrix at time t, Q is the Q-matrix, μ is a hyper-parameter that captures how much a student knowledge increases in each attempt and λ controls the amount of regularization applied to the model.

The maximum number of attempts to use in the training phase is set as a model hyperparameter.

5.3.2 Evaluation

The described methods factorize the student performance matrix or tensor in two components: student knowledge and Q-matrix. Usually, we cannot assess these two components directly since student knowledge is difficult to measure by only observing the students' outcomes, and many times a Q-matrix is not provided. The best model is the one that better predicts students' performance in an test dataset using root means square error (RMSE).

5.4 Analysis of Similarities

We want to evaluate how the discovered Q-matrices relate to professors' manually annotated Q-matrix in this scenario, as represented in Fig. 5.8. How similar to the manually annotated Q-matrix are the factorized ones? Our proposed approach to compare the factorized Q-matrices with the one annotated by the professors is to use analysis of similarities (ANOSIM) with hierarchical clustering, described below. We then relate student performance with the factorized Q-matrix by plotting the clusters, their respective concepts and providing insights to the teachers by aggregating the students' attempts and success rate information into the plot.

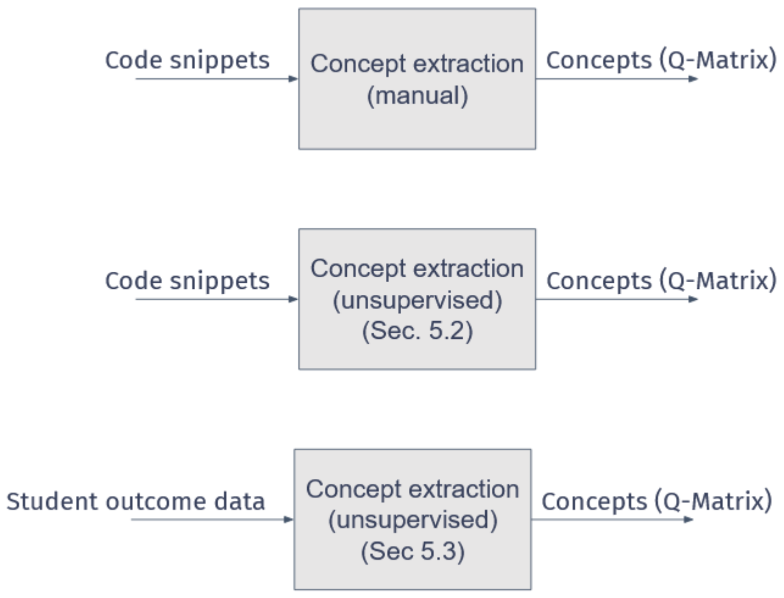

Code snippets
Concept extraction (manual)
Concepts (Q-Matrix)
Code snippets
Concept extraction (unsupervised) (Sec. 5.2)
Concepts (Q-Matrix)
Student outcome data
Concept extraction (unsupervised) (Sec 5.3)
Concepts (Q-Matrix)

Figure 5.8: Q-matrix comparison.

In order to assess how well the factorization methods are retrieving the Q-matrix, the non-parametric ANOSIM statistical test [144–146] was used. The ANOSIM test is a non-parametric ANOVA-like test statistic widely used in ecology to verify if similarities within animal groups with smaller spatial distances are statistically more significant than similarities between animal groups with larger spatial distances [145, 146]. Translating it to a machine learning clustering problem: it measures if the distances within clusters (the intracluster distance) is statistically smaller than the distances between clusters (the intercluster distance). The test statistic is constrained between -1 and 1, where positive numbers suggest that within-group similarities are higher than between groups, 0 indicates random grouping (null hypothesis), and negative values indicate that the similarities between groups are higher than within groups [145]. The dissimilarity matrix is transformed to its rank version, and the value of ANOSIM statistic, called R, is given by [146, 147]:

$$R = \frac{r_b - r_w}{n(n-1)/2} \tag{5.7}$$

where r_b represents the average of all rank similarities among observations in different clusters (intercluster distances), r_w represents the average of all rank similarities among observations in the same clusters (intracluster distance), and n is the number of observations. Permuted R statistics are also calculated to test whether the R statistic in the proposed grouping scheme is consistent. The ANOSIM test permutes the groups to which the observations belong and recalculates the R statistic. The p-value is given by the proportion of permuted R statistics that are greater than the original R. Therefore if the result of the original clustering is unlikely to be found at random, we reject the null hypothesis and accept that there is some degree of discrimination between groups [147].

In our case, we need to compare the groups created from the factorized Q-matrices with the actual distances between questions provided by the manually annotated Q-matrix. A hierarchical clustering algorithm was used to group the questions based on their factorized Q-matrices' distances. The central intuition is to measure if both Q-matrices contain the same underlying structure, meaning that if the factorization is assigning to questions the same concepts from the reference Q-matrix, these questions will belong to the same group after clustering them and their distance in the reference Q-matrix will also be small.

Chapter 6

Results and Discussion

In this chapter, we present each method's results and compare them to a reference Q-matrix. We also present the results of a survey conducted with students to understand their perception of the system.

6.1 Q-Matrix Discovery from Code

We run each hyperparameter combination from the 1680 possibilities 10 times and calculated their average coherence and standard deviation. Next, the two best-ranked results are analyzed. They were calculated using Fagin's algorithm [148] for top-5 and top-10 terms UMass coherence. Table 6.1 shows the set of hyperparameters for each experiment.

Table 6.1: Set of Hyperparameters for Each Experiment

	Min DF	Binary	Vectorizer	Method	# of clusters
Exp. 1	0.35	True	NCut	NMF	7
Exp. 2	0.05	True	Count	LDA	12

6.1.1 Experiment 1

After the document-term matrix factorization, we hard-assigned each document to the topic with the highest relevance (highest weight in the document-topic distribution). Table 6.2 shows the number of documents assigned per topic. After assigning each document to its related topic in this experiment, the documents are only assigned to four of the seven topics. Fig. 6.1 shows the documents projected to two dimensions using principal component analysis (PCA) [149].

Using a minimum document frequency of 35% kept only 23 valid terms. Fig. 6.2 shows the essential terms per topic where the terms that are exclusively important for a single topic (a term is vital if it is above the 75th percentile of all weights) are

Table 6.2: Number of Documents per Topic in Experiment 1

Topics	1	2	3	4	5	6	7
# of documents	**5**	**4**	**26**	**19**	0	0	0

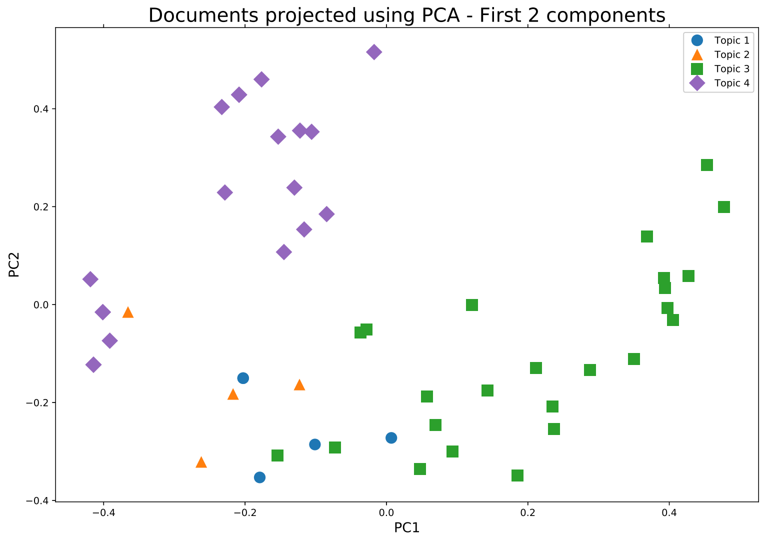


Figure 6.1: Documents projected into the first two dimensions using principal component analysis. Points with the same color and marker belong to the same cluster.

denoted in green. In this plot, topics 3 and 4 share almost all terms. By adjusting the document-term matrix values using the NCut vectorizer, the factorization split topics 3 and 4 using the conditional *if* term. Topic 4 is exclusive for code snippets that are solved using conditional statements, whereas topic 3 comprises the opposite.

Fig. 6.3 shows the topic distribution per document. As explained in Section 5.2.2 (topic extraction), distribution over topics describes a document. Darker cells imply that the topic characterizes a document better. As stated before, if a topic strongly characterizes a document, then we can hard-assign it to a single topic. However, Fig. 6.3 shows that most documents assigned to topics 1 and 2 (top part of the plot) spread throughout the topics. It suggests we have to combine the most important terms for each topic to interpret these code snippets.

Analyzing the code snippets from topic 1, they combine for-loops with conditional statements. Topic 2 is a mixture containing the code snippets that do not belong to any other topic.

Fig. 6.4, using the LDAVis tool [150], calculates the topics' distance and projects them to 2D using principal coordinate analysis. Topics 3 and 4 are located close to each other, and they correspond to 45% of the terms and 83% of the documents. Fig. 6.4 also validates that these topics are not that different when their crucial terms are analyzed. Still, the conditional statements that characterize topic 4 are

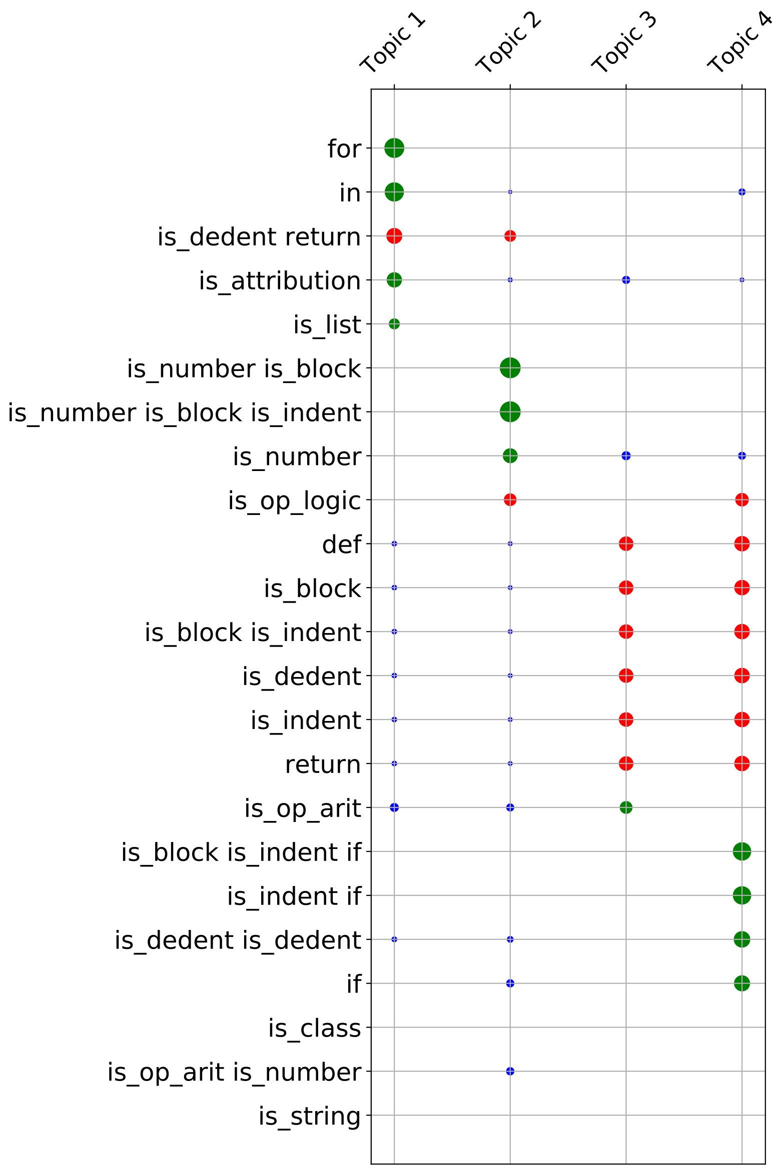


Figure 6.2: Term importance for the four most populated topics. Each row represents a term, the size of the point corresponds to the term's weight for the topic, and red points are the points above the 75th percentile of all weights. The green points denote words above the 75th percentile limit on only one of the four topics.

enough to produce a linearly separable 2D data projection, except for a few outliers, as shown in Fig. 6.1.

6.1.2 Experiment 2

Table 6.3 shows the number of assigned documents per topic with hyperparameters combination producing a more uniform grouping scheme than the previous one. Although we initially set 12 clusters, two of them (topics 9 and 11) are empty after assigning each document to the topic with the highest relevance (weight). Topics 6, 8, 10, and 12 have the largest number of documents. Fig. 6.5 shows the topic per

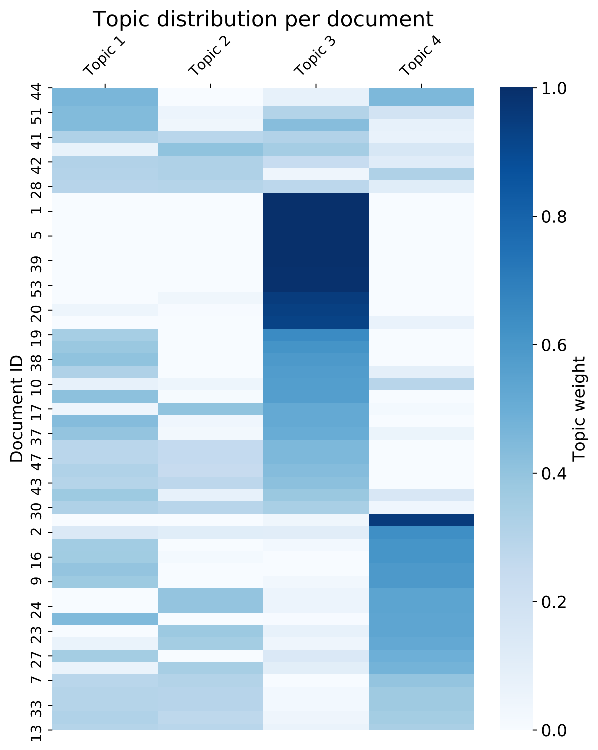


Figure 6.3: Topic distribution per document. Darker cells indicate a better description of the document by the topic.

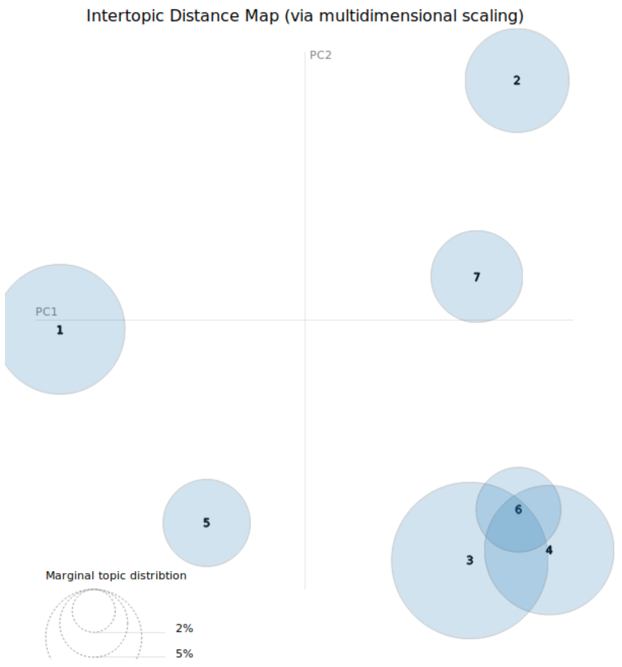


Figure 6.4: Topics are represented as circles proportional to the number of terms whose weights are most associated with the topic and the distance between the circles is the intertopic distance.

document distribution where the topics better characterize each document.

Table 6.3: Number of Documents per Topic in Experiment 2

Topics	1	2	3	4	5	6	7	8	9	10	11	12
# of documents	2	1	1	2	1	**13**	1	**14**	0	**7**	0	**12**

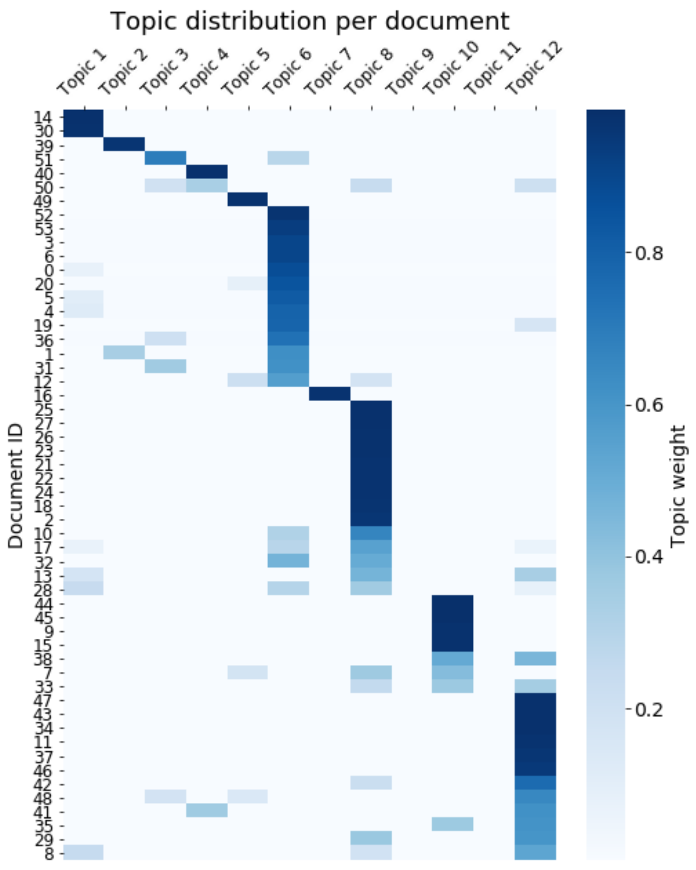


Figure 6.5: Topic distribution per document. Darker cells indicate a better description of the document by the topic.

The complete term per topic plot for this experiment is omitted because it is long and challenging to read. Using a minimum document frequency of 5% increased the number of terms to 236.

Fig. 6.6 shows the intertopic map. The main topics 6, 8, 10, and 12 correspond to 85% of the documents and 77.4% of the terms and we do not observe any main topic overlap in this plot. The next subsections analyze these main topics in detail. Topics 2 and 4 will also be analyzed since they occupy a different space on the map. This step belongs to the topic filter and selection phase from the code-clustering pipeline depicted in Fig. 5.1. After hard-assigning the documents to the clusters (removing topics 9 and 11), merging topics 2 and 4, and removing topics with a few documents (less than three documents per topic: topics 1, 3, 5, and 7), it resulted in five conceptual clusters (six from the original topics in total) to be analyzed in detail.

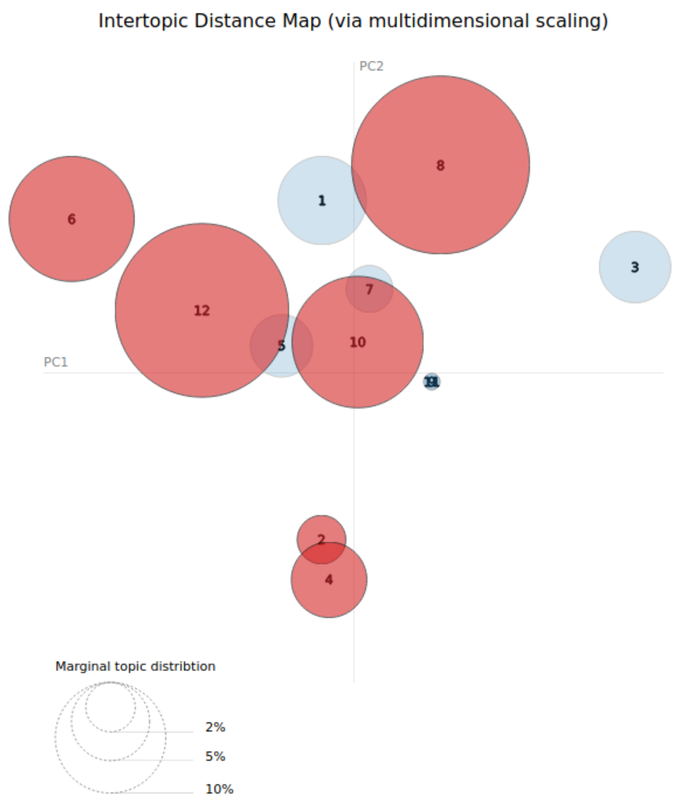


Figure 6.6: Intertopic distance map for experiment 2. Topics are represented as circles proportional to the number of terms whose weights are most associated with the topic and the distance between the circles is the intertopic distance. Topics in red are further discussed in detail.

Using the Sievert and Sheley relevance metric implemented in the LDAVis tool [150], the top-30 most relevant terms per topic were extracted. Table 6.4 shows the five most relevant terms. We wrote a description for each topic after analyzing the essential terms and code snippets from each class. Pages 49 and 50 show examples of code snippets for each topic and highlights the terms used to define them.

- Topic 8 is strongly characterized by conditional statements, logical operators, and Boolean values.
- Topic 6 does not seem to have a clear definition by just inspecting its terms. Indentation terms appear among the most relevant ones. By analyzing the code snippets, topic 6 comprises code with one indentation structure (simple coding structures), sometimes without assigning variables to solve the exercise.

- Topic 12 is characterized by for-loops, especially range loops combined with arithmetic operations.
- Topics 2 and 4 present for-loops, conditionals, and lists, and these tools are used to perform string operations. A strong mark is the presence of the auxiliary functions split and join.
- Topic 10 is about lists and their usual operations: for-loops, conditionals, and appending elements.

Table 6.4: Five Most Relevant Terms for Each of the Analyzed Topics. The Terms Starting with 'Is' Are the Special Annotated Terms Explained in Section 5.2.1 (Data Transformation)

Topics 2 & 4	**Topic 6**	**Topic 8**	**Topic 10**	**Topic 12**
split	is_op_arit	is_op_logic	append	range
is_string	is_indent	if	is_list	is_op_arit
join	def	else	for	for
for	is_number	True	is_attribution	is_number
if	len	False	if	is_attribution

Topic 2

```
def sort_csv(csv):
    items = csv.split(", ")
    items.sort()
    return ", ".join(items)
```

Topic 4

```
def sort_dedupe(words):
    items = words.split(" ")
    items_dedupe = []
    for word in items:
        if word not in items_dedupe:
            items_dedupe.append(word)
    items_dedupe.sort()
    return " ".join(items_dedupe)
```

Topic 6

```
import math
def formula(D):
    C = 50
    H = 30
    Q = round(math.sqrt(2*C*D/float(H)))
    return Q
```

Topic 8

```
def is_prime(number):
    """Returns True for prime numbers, False otherwise"""
    # Edge cases
    if number == 1:
        prime = False
    elif number == 2:
        prime = True
    # All other primes
    else:
        prime = True
        for check_number in range(2, int(number//2)+1):
            if number % check_number == 0:
                prime = False
                break
    return prime
```

Topic 10

```
def dedupe(dup_list):
    nodup_list = []
    for i in dup_list:
        if i not in nodup_list:
            nodup_list.append(i)
    return nodup_list
```

Topic 12

```
def fatorial(number):
    total = 1
    for i in range(number, 1, -1):
        total = total * i
    return total
```

To better understand the topics' differences, we performed a topic dissimilarity analysis shown in Fig. 6.7, representing the real distances of the topics, not reduced to two dimensions as in Fig. 6.6. In this analysis, topic 8 (conditional) is the furthest from all others. Topic 10 is central, being slightly closer to topic 4 than to topic 12. This result is reasonable as it contains elements between these two topics (numeric and range loops vs. loops with strings). Table 6.4 shows their differences as being the data types, which can be observed in the most important terms ("range" and "is_number" for topic 12, "append" and "is_list" for topic 10 and "split", and "is_string" for topics 2 and 4).

We also analyzed how the test dataset fits into the six (including topics 2 and 4) valid topics to understand if the topics are representative of the possible concepts present in an unseen code. We assigned each code to the topic with the highest weight as we did for the training set. Table 6.5 shows the number of assigned documents per topic. Except for two documents, all the others belong to one of the six valid topics. It confirms that the different topics (the ones considered invalid) detect specific code traits and not their general concepts. It is important to notice that topic modeling is a soft clustering technique: a document has a probability of belonging to each topic and can be associated with more than one. So, a document can be related to the main topic with its specificity related to minor ones.

Table 6.5: Number of Test Set Documents per Topic for Experiment 2

Topics	1	2	3	4	5	6	7	8	9	10	11	12
# of documents	2	**4**	0	**1**	0	**23**	0	**17**	0	**3**	0	**15**

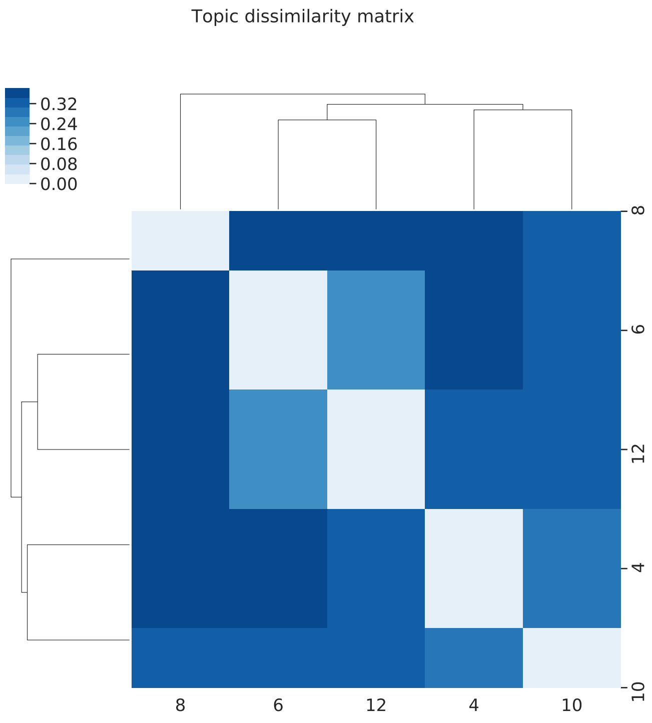


Figure 6.7: Topic dissimilarity matrix. The darker the cell more distant the topics are from each other.

6.1.3 Coherence Evaluation

Both experiments were analyzed using the UCI coherence metric with NPMI [134], as described in Section 5.2.3 (topic filter and external evaluation), to validate how well the proposed methodology performs in an external dataset. Although AST trees have been used to cluster code, they do not provide an intuitive way to analyze the important features besides reading it. We compare our results with a K-means clustering method using the proposed augmented tokenizer and logistic regression to extract the important features per cluster as a baseline. We also compared our best results using the standard tokenizer instead of our proposed tokenizer. We used $k = 5$ for K-means since there were five main conceptual clusters found in the LDA. We ran each method 100 times and averaged their UCI coherence metric. Statistical difference was measured using the Mann–Whitney U test [151], and all the results were statistically significant with $p < 0.001$. Table 6.6 reports the mean and standard deviation for each experiment. In the UCI coherence with NPMI metric, the values are bounded between 1 and -1, where 1 means that the top words only occur together, zero means that they are distributed as expected under independence, and -1 means that they only occur separately. The UCI coherence for the standard tokenizer using the top-10 terms could not be measured because there were no important top-10 term pairwise combinations in this setting that appeared in at least one document. The NMF experiment with the augmented tokenizer considering the top-10 terms demonstrated the best UCI occurrence metric, followed by the LDA experiment, which had the best performance considering the top-5 terms.

Table 6.6: The Mean and Standard Deviation of UCI Coherence Using NPMI

	C_{UCI} (NPMI)	
	Top-5 terms	Top-10 terms
NMF (experiment 1 with augmented tokenizer)	0.73 (0.03)	**0.83 (0.02)**
LDA (experiment 2 with augmented tokenizer)	**0.75 (0.06)**	0.76 (0.04)
K-Means (with augmented tokenizer)	0.55 (0.09)	0.53 (0.03)
LDA (best result for standard tokenizer)	0.53 (0.03)	—

6.1.4 Discussion about Experiments 1 and 2

Experiments 1 and 2 are the two best-ranked results on the top-5 and top-10 UMass coherence metric. Both experiments have different hyperparameters. Experiment 1 uses a minimum document frequency of 35%, NCut to weight the terms, and then perform NMF to extract the topics. Experiment 2 uses a minimum document frequency of 5%, regular count of words, and LDA to extract the topics.

We found both experiments to have their main concepts in a few clusters (two main clusters in Experiment 1 and six main clusters in Experiment 2). The remaining clusters are associated with code specificity. In the case of Experiment 1, using NCut and a high document frequency threshold, the topic modeling from Experiment 1 focused on finding structures with high volume and cooccurrence rates, resulting in separation of the *if/else* structure from the rest. The conditional structure was first separated from a hierarchical perspective, and the remaining structures were all grouped in a cluster. In Experiment 2, the conditional topic (topic 8) is also the furthest from the other topics. As shown in the hierarchical clustering of Fig. 9, this topic is the last one to be aggregated (or the first one to be separated). The common code snippets between the conditional clusters in each experiment also validate this result. From the 14 code snippets associated with the conditional topic (topic 8) in Experiment 2, 11 of them (79%) belong to the conditional topic in Experiment 1 (topic 4). Therefore, Experiment 2 demonstrates more granularity than Experiment 1.

6.1.5 Topics Contextualization

Experiment 2 shows a better distribution of exercises and less overlap among topics than Experiment 1; we followed up the analysis with this experiment, which was the second-best result considering the C_{UCI} with NPMI metric.

- **Theme identification:** after tokenizing the labels given by the professors, we separated the most frequent tokens and manually created the topic titles, as follows. All the tokens in the presented titles appeared in at least 50% of the labels.

  1. Topics 2 & 4: string manipulation,
  2. Topic 6: math functions,
  3. Topic 8: conditional structure,
  4. Topic 10: list loops,
  5. Topic 12: math loops.

- **Concept identification:** each professor was asked to associate up to three concepts (from the 15 available in Table 5.3) to each presented code. Four professors analyzed each code. In 37 of the 54 code snippets, there was at least one concept in common between all four professors. In 53 of the 54 code snippets, at least one concept was common between three out of the four professors (75%). Therefore, we decided to use the 75% threshold of the agreement to relate the exercises' concepts. The concepts in each topic were aggregated to provide an overview of the main concepts needed to solve the cluster's problems, as summarized in Table 6.7. These results validate the topic themes defined in the previous task: the professors also elected each topic's main concept as a word in free-text labels. Notice that we performed the two tasks independently.

Table 6.7: The relation between the Found Topics and the Main CS1 Concept Related to the Topic

Topic	Main Concept	Agreement inside cluster
2 & 4 - String manipulation	7. Data type: string	66.7%
6 - Math functions	14. Function	50%
8 - Conditional structure	11. Conditional	86.7%
10 - List loops	12. Loop	71.4%
12 - Math loops	12. Loop	75%

- **Intruder identification:** four code snippets were presented in each of the five groups, three belonging to the same topic (randomly chosen from the topic pool) and an intruder (also randomly chosen from another topic). Fig. 6.8 shows a confusion matrix, presenting how well the professors could distinguish the intruder in each topic. In this figure, each row sums to 1 and represents how often the professors correctly guess the intruder (the diagonal values) and how often the professors confuse the intruder (the intruder cluster is depicted in the columns). We can draw some insights from this analysis:

– The "conditional structure" topic performed well, with the intruder code being identified 79% of the time, meaning that identifying a code snippet from a different cluster can be done 4 out of 5 times.

– The intruder code inside the "math loops" topic was identified 2 out of 3 times (64%), being confused with "list loops" the last third of the time. These topics work on the same main concept, as seen in the concept identification task.

– The same behavior is not seen for the "list loops" topic. This topic and the "string manipulation" topic present very similar behavior. They can be distinguished frequently (2x better than the 25% random baseline), but we do not see a confusion pattern.

– On the other hand, the "math functions" topic seemed to confuse the professors, being switched half of the time with the "math loops" topic. By backtracking the previous two tasks and analyzing them with this point of view, this topic's main concept was function, present in all exercises. As stated in Section 6.1.2, this topic does not have well-defined terms; indentation terms appeared among the most relevant. Although the indentations when programming in Python can indicate the difficulty of an exercise, they are not a natural human way of splitting them. Therefore, it is hard for humans to interpret this kind of clustering scheme.

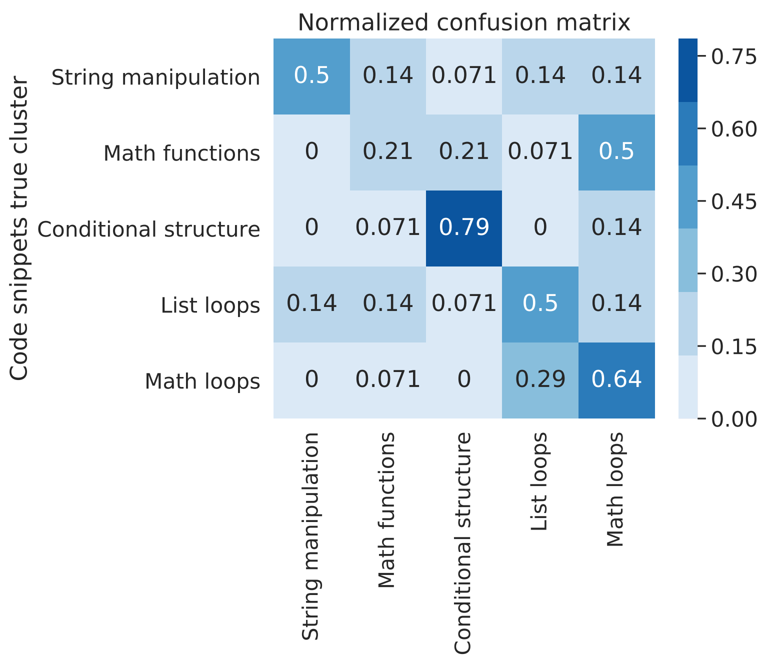


Figure 6.8: Normalized confusion matrix for the intruder-identification task.

6.2 Q-Matrix Discovery from Student Performance

We first evaluate the fit of the models using the student performance prediction task. We choose each setting's best model to conduct the ANOSIM experiment and compare the extracted Q-matrix with the validated one.

6.2.1 Train and test split

We used 10-fold cross-validation to choose the best set of parameters. For each fold, we randomly selected 70% of the users (with all their records) as train and the remaining 30% as test. Notice that we are splitting users (not individual students' attempts) to avoid leaking information from the training dataset into the test dataset. This prediction task faces the cold start problem: there is no data to begin the test students' predictions. Therefore, we moved 20% of the test students' first attempts into the training dataset and predicted only the remaining 80% of the test students' attempts.

For FDTF, since some student sequences can be way longer than others, we cut off some sequences. We vary this maximum cut-off value and report it as the number of attempts.

6.2.2 Prediction results

Student performance prediction is a binary problem: either the student solves the exercise correctly (pass) or incorrectly (fail). We use Root Mean Squared Error (RMSE) to evaluate this task, as it has been traditionally used in the literature. We examine the prediction performance of both NMF and FDTF methods according to their RMSE.

We select the best training results for each configuration and compare their test results. Table 6.8 summarizes the best results for each method and task. FDTF is used to predict each student's attempt and NMF is used to predict the first student attempt or the average grade considering all attempts. For the FDTF method, similar train RMSE was obtained when using 20, 50, and 150 attempts and mu = 0.1. Notice that FTDF improves its performance prediction on test data as the number of attempts increases. NMF's best average attempt configuration is when using L2 regularization with λ in the 0 (no regularization) to 0.2 range. Its first attempt configuration performs better when using seven concepts and L2 regularization penalty with λ in the range of 0.2 to 0.4. Although NMF has a low training error, it does not perform well for predicting the average outcome in the test set, even with a high λ (0.2), to avoid overfitting. After finding the best configuration for each method, we retrained the models using all available data (not in folds) and factorized the matrix or tensor into a student knowledge matrix or tensor and a Q-matrix for the Q-matrix evaluation.

Table 6.8: Prediction Results

Method	Concepts	Train RMSE	Test RMSE
FDTF (training with 20 attempts)	3	0.437 (0.003)	0.517 (0.057)
FDTF (training with 50 attempts)	6	0.437 (0.003)	0.469 (0.038)
FDTF (training with 150 attempts)	12	0.438 (0.002)	0.43 (0.021)
NMF (averaging attempts)	3	0.465 (0.003)	0.617 (0.049)
NMF (first attempt)	7	0.378 (0.002)	0.598 (0.043)

6.3 Analysis of Similarities

We conducted the ANOSIM experiment to verify if there is a positive correlation between the groups that are found in the factorized Q-matrices and the distances between questions on the human-annotated Q-matrix. We clustered the factorized Q-matrices for each experiment using a hierarchical clustering algorithm with ward linkage [152] and euclidean distance. We used the clustering results as the grouping scheme to be analyzed and provided the Jaccard dissimilarity metric from the reference Q-matrix as intra-cluster and inter-cluster distances. Since the ANOSIM test transforms distances to ranking and does not compare them directly, there is no problem using different similarity metrics. We opted for the Jaccard similarity in the reference Q-matrix because this matrix, differently from the extracted one (that each row sums up to one), contains several ones or zeros per row, indicating either the presence or absence of a concept. The Jaccard similarity calculates the proportion of the shared concepts between the two questions. According to Provost and Fawcet [153] this similarity metric is useful when a common absence of a characteristic is not as important as a shared one.

We varied the number of clusters from 2 to 15 (maximum number of concepts as defined in Table 5.3) using 100,000 permutations for each. Table 6.9 reports our findings. None of the experiments using student performance data can produce a Q-matrix significantly correlated with the human-annotated Q-matrix distances, as seen inconclusively in Winters *et al.* [104]. On the other hand, statistical significance is achieved using the topic modeling techniques described in Chapter 5. The best overall results are using the raw data from the topic modeling technique, grouping them into 14 clusters. These results validates what was seen by Barnes [101] and Winters *et al.* [104] in their research: the needed skills for each question (formalized through the Q-matrix) based solely on student performance data does not correspond to professors' expected skills for each question.

Table 6.9: ANOSIM statistics

Method	Statistical Significance (p <0.01)	Concepts	Best R Statistic
FDTF (20 attempts)	No	–	–
FDTF (50 attempts)	No	–	–
FDTF (150 attempts)	No	–	–
NMF (averaging attempts)	No	–	–
NMF (first attempt)	No	–	–
Topic Model (raw result)	Yes	12	0.21 (14 clusters)
Topic Model (result after refinement)	Yes	5	0.16 (14 clusters)

Figure 6.9 depicts the resulting dendrogram with the respective clusters. We used the Ward method [152] with the euclidean distance as a dissimilarity metric. From this structure, we can notice three major concept clusters divided into several small subclusters.

To analyze the relationship between the 14 created clusters from the factorized Q-matrix and the reference Q-matrix, Figs. 6.10 and 6.11 shows the average student attempt versus the percentage of successful attempts per cluster. The point size is proportional to the number of students that have attempted questions in that

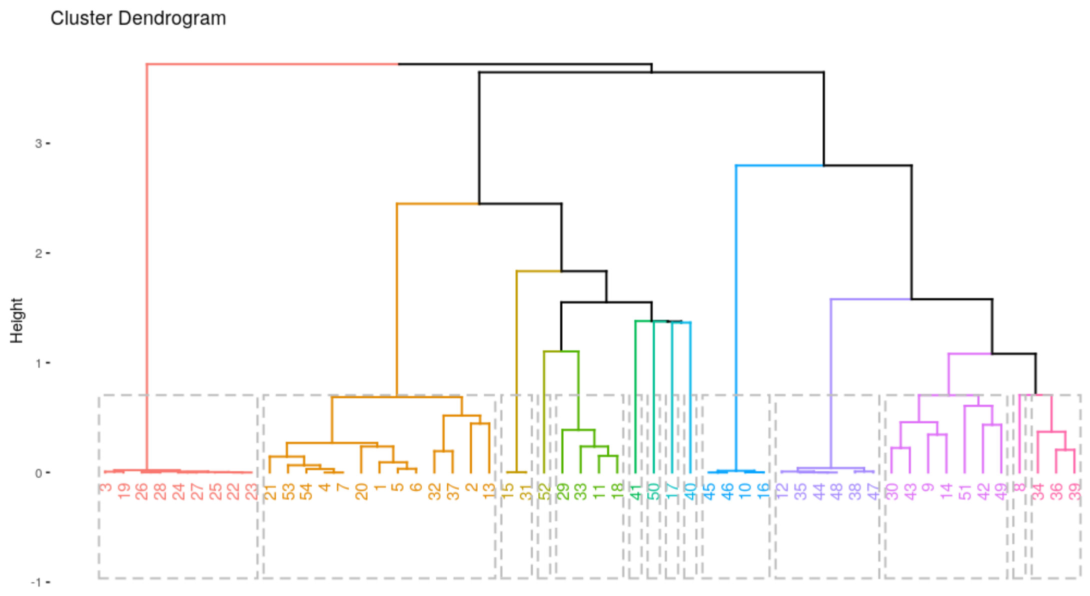


Figure 6.9: Question dendrogram using Euclidean distance. Each color (each dotted rectangle) represents a cluster.

cluster, which is also indicated by the number inside the point. Each color represents the concepts in which 60% of the questions belonging to the cluster agree on. For example, most of the questions inside the orange cluster work on the "Conditional" concept.

Figs. 6.10 and 6.11 depict an overview picture of the exercises, their concepts, and the students' interactions with them. Fig. 6.10 depicts all the clusters and Fig. 6.11 zooms in the dense part between 0 and 6 average attempts per student per problem. This image can be seen as four quadrants: on the top left are the easy questions (high successfully attempts percentage and attempts on average); on the bottom left are the medium questions (questions with low success rate but also a low attempts average); on the top right there are not any questions (it would be questions with high success rate but also a high number of tries, which is unlikely) and finally, on the bottom right are the questions where students have difficulties. Excluding the outlier and analyzing by the quadrants, the red cluster (1st quadrant) requires the use of the "Function" concept, which are introductory CS1 exercises where the student is supposed to write a function with simple math operations directly in the return statement. The exercises belonging to this cluster are solved in a smaller number of attempts and have a higher success rate than the other clusters. There are some surprising clusters with a high successful rate involving conditional and loops in lists, which are considered complex concepts for CS1. However, each of these clusters has only one exercise, becoming outliers of the concepts they represent. The clusters around 1 to 3 attempts on average involve more advanced concepts such as strings, loops, and conditionals. Finally, some clusters have more than four

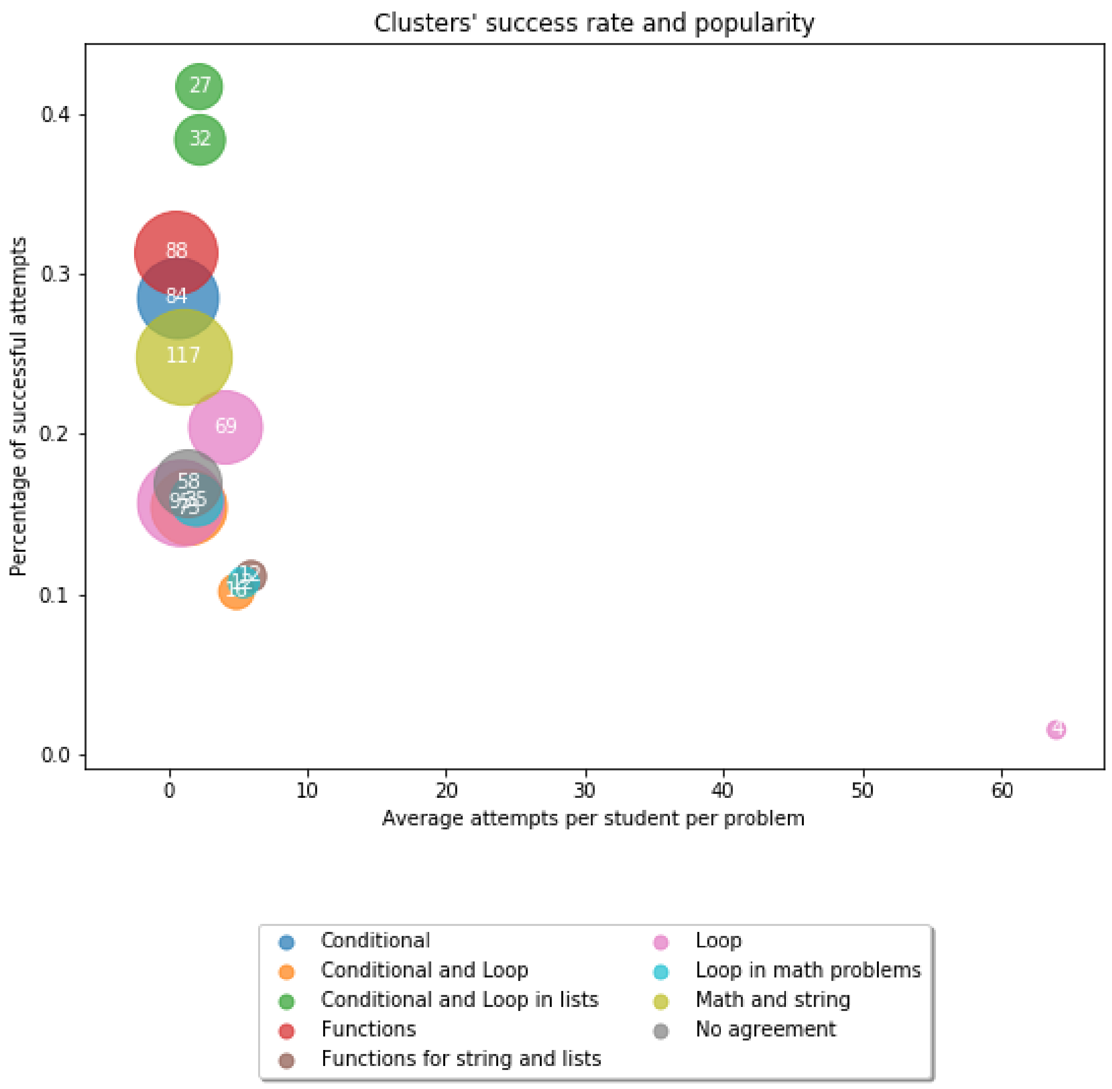


Figure 6.10: Each color represents the main concept in the cluster and the size, its popularity among students.

attempts on average per student per problem and a low success rate. They mostly involve loops combined with some other concept.

Another interesting thing from this overview picture is to understand the students beyond the averages. In section 4.1, we reported the average successful attempt of the dataset as approximately 21%. From this figure, we can easily spot the "outliers". The clusters between 1 to 2.5 attempts bring the average to this value, while the easier clusters or the harder ones are increasing or decreasing the success rate. With this information in real-time, educators can provide better assistance where students have more difficulties (4th quadrant, challenging exercises) or decreasing the workload where students have already mastered the concept (1st quadrant, easy exercises).

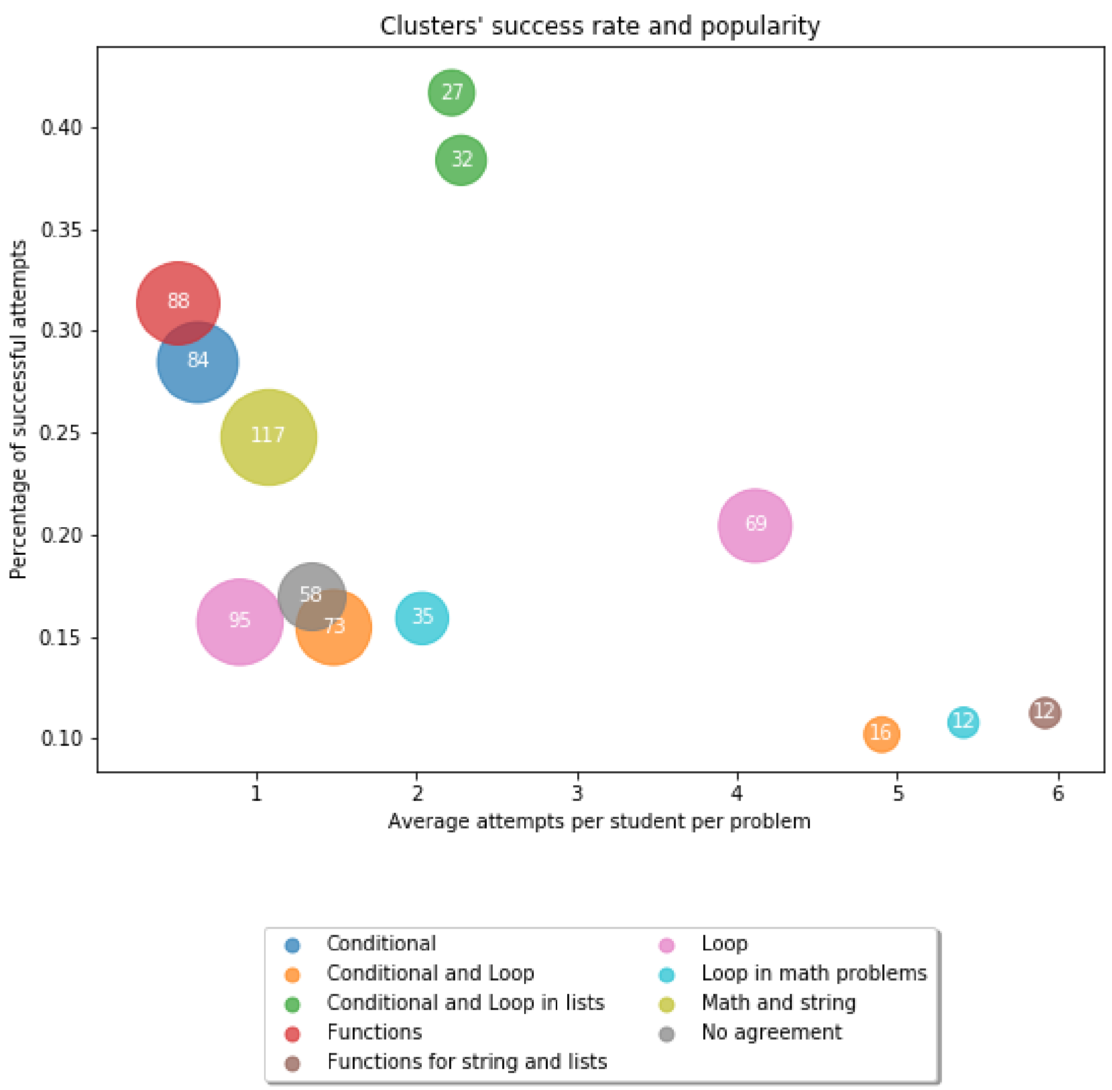


Figure 6.11: Each color represents the main concept in the cluster and the size, its popularity among students. This figure zooms in the dense left size of plot 6.10.

6.4 Student Survey

The Machine Teaching web system was developed not only to acquire student data and perform code and concept analyses but also to improve professors' and students' experience while solving exercises, studying, or teaching a class. In this section, we present the results of a survey conducted with the students from the four classes that used the system regularly for one entire semester, as described in section 4.2.2. Since we are using an agile methodology to develop the system, the survey did not include the student dashboard presented in Section 3.4, available only on the most recent version of the system. Participation was anonymous and voluntary, and the survey was conducted on Google Forms using a Likert scale on nine items with five values ranging from "Disagree completely" to "Agree completely". There was an extra item asking which software the students were using to write and test their code and an open-ended suggestion part. We were investigating user perception in three areas:

1. **Interface satisfaction:** subjects found the Machine Teaching system easy to use; they could find the exercises that had to be done, exercises that they had already done and where they preferred to write their code (inside or outside the system).

    I. The system has a friendly interface.

    II. It was easy for me to find past exercises to study.

    III. It was easy for me to find this week's exercises.

    IV. Which software did you use to prepare the answers for the questions?

2. **Feedback coverage:** subjects were inquired about the debugging features and their feedback relevance.

    V. When the answer was wrong, using the Console output helped me solve the exercise.

    VI. When the answer was wrong, checking the wrong inputs helped me solve the exercise.

3. **Learning perception:** subjects responded if they perceived the system as a tool to help them study and learn.

    VII. Using the system to solve the exercises has helped me learn.

    VIII. I used the system to study for the exams.

    IX. I would recommend the system for a student in a CS1 course.

    X. I would like to have used the system when studying the user input subject.

In total, 41 students replied. Figs. 6.12 and 6.13 show the students' responses concerning Interface satisfaction. In general, they were able to navigate through the system and find past and current exercises. Current exercises had a higher median than past exercises since the system was projected to present the most recent exercises on top. Friendly interface perception had a higher dispersion than the other two questions, but it still received a median of 4 out of 5 on the Likert scale. Fig. 6.13 presents the proportion of students who used only the system compared to the students who used other software to write their code. Almost half of the students (48.8%) that replied used only the system to write and test the code. The other half was divided between using other software directly and using the system just to submit the answer when they thought it was correct (29.3%), and trying it on the system first but changing to another software if they could not get the answer correctly in the first tries (22%).

When analyzing feedback coverage responses in Fig. 6.14, the overall user grades for this category are lower, especially when inquired about the “Console” feature. Complementing it with the responses for item IV (Which software did you use to prepare the answers for the questions?) and with some suggestions done in free-text format, we can determine which enhancements need priority in the students’ point of view. Of the 41 students, 21 wrote improvement suggestions. From these 21, five mentioned features related to the feedback coverage topic. The most mentioned enhancement was to point errors more clearly, followed by an ad-hoc testing scheme (not just autocorrect), where users could test arbitrary argument values.

Students also responded to four questions relating their learning achievements and the web system, as shown in Fig. 6.15. In general, they thought that the way the system was structured has helped them learn, and they would recommend it to other students. However, it was divided whether they used it for studying for the exams or not. It is important to highlight that the exams are done on paper, not on the computer. Even though we need more information to understand why they did not use it for studying for the exams, this difference could justify why they did not see the system as a primary resource while studying. The system was not available for the last two course subjects (user interaction, main program, and modularization). Most of the students responded that they would like to have used the system to make the exercises from this part as well.

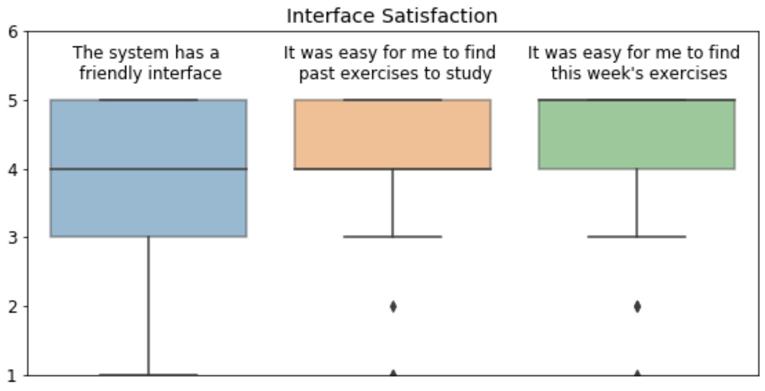


Figure 6.12: Boxplots for user interface satisfaction questions

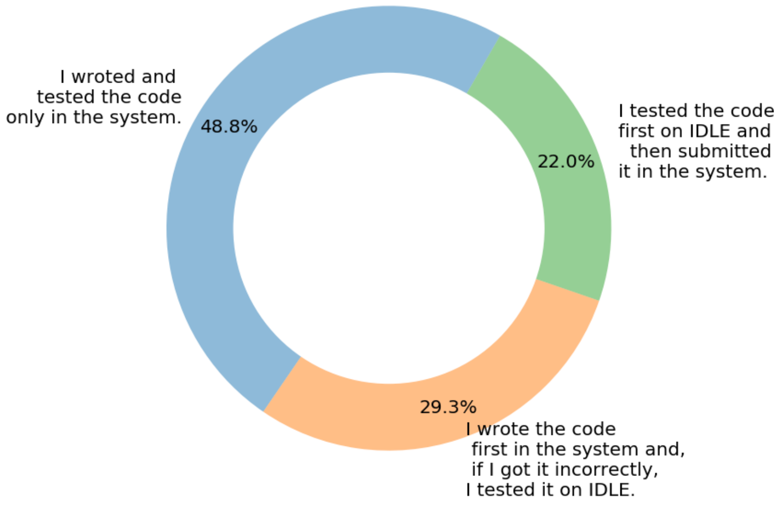


Figure 6.13: Software used by students to write code

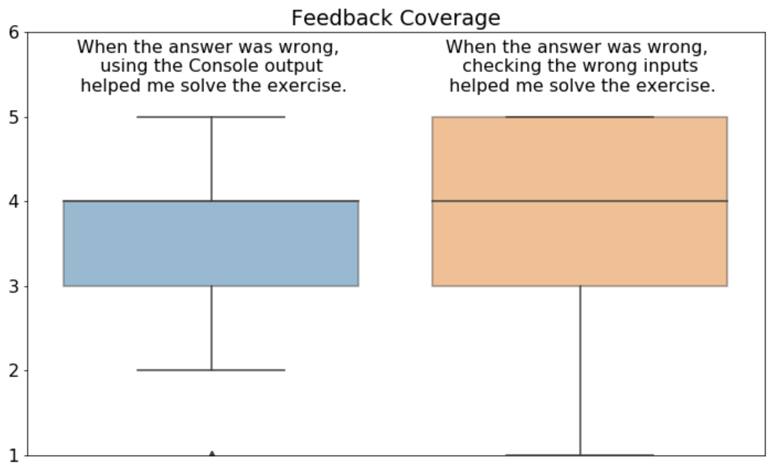


Figure 6.14: Boxplots for feedback coverage questions

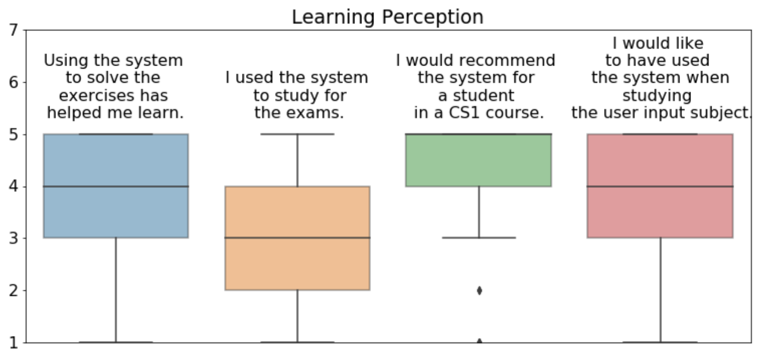


Figure 6.15: Boxplots for learning perception questions

Chapter 7

Conclusion

This thesis presented improvements to assist in closing the learning feedback loop applied to computer science education. Besides contributing by proposing new methods to extract concepts from code and student performance in an unsupervised way, our research was able to approximate research and practice by designing and deploying a web-system currently in use at the Universidade Federal do Rio de Janeiro. During the year 2020, UFRJ has become completely remote due to the COVID-19 pandemic. Having an already tested and deployed system tailored for the teaching methodology used in the introduction to programming classes helped in the transition from face-to-face classes to remote classes. At the present moment, the built web-system is being used by 15 classes and 500 students. In total, more than 1200 students have used the system in the past three years.

We designed and implemented a learning environment based on a teaching methodology, inserting the professors and students into the designing loop. User stories were created to gather the desired features, and system architecture was proposed as an enhanced version from the literature. The Machine Teaching system was developed to assist students and professors in modularized function-based Python classes. Since we are using a formal didactic approach to construct the learning environment, the findings from our work will be used to adapt the teaching methodology, which will reflect again on the system, running a loop improvement process. We surveyed the students from the previous two semesters about their perceptions of the system. The student survey showed that, even though students perceived the system as helpful for learning, they did not use it to optimize their studying strategies. Students' experience using the system was positive overall, and we could identify improvement points, especially on the feedback coverage topic.

The exploratory analysis found a decreasing trend in the number of active students during the semester and reproduced early dropout prediction models, corroborating the literature findings. We gathered relevant findings from the literature on this task and attested their validity when working with this dataset. We proposed an enhanced model dividing the features by week instead of aggregating them. The weekly features improved the model results and provided finer-grained relevant features for professors. Using explainable models such as decision trees and hierarchical clustering to perform the analyses can integrate the professors into the decision-making process.

7.1 Final Remarks

This thesis's main contribution is to propose a methodology to extract concepts from code unsupervisedly by clustering code optimizing metrics that are positively correlated with human-interpretability. With the associated evaluation metric, the proposed method was able to find semantically related code-clustering schemes suited for human interpretability with minimal supervision. The method is expected to provide semantics for large amounts of unannotated code. Although we did not investigate the methodology's applicability to other domains, we believe that it could potentially be generalized, in future work, by modifying the tokenizer step.

Although code clustering in the CS1 context has been widely applied using the AST trees, the advantages of working with topic modeling are the terms per topic results that may help experts better assess each cluster's contents. The methodology has also been shown to overcome the small-sized code snippets challenge by extending the tokenizer to augment the corpus with the code structure. The standard tokenizer could not create semantically related topics, but adding structural information, as features: indents and data types, and enforcing the order using n-grams enriched the code representation and found topics suitable for human-interpretability. For example, augmenting the corpus with structural information as indents/blocks (in Python, indents indicate how deep a block of code is; other languages like C++ and Java could count the number of "{" and "}") helps to separate single loops from nested loops. Combining trigrams (to enforce order) with structural information can distinguish subtle differences in precondition and postcondition loops. Notice that postcondition loops do not exist in Python, so we could not verify this specific assumption. In our dataset, we expect trigram tokens to be enough to capture these varieties because a typical CS1 solution does not have more than three or four nested structures. Still, it may limit our model in identifying large nested structures on more complex code. Also, even though there is a recursion concept in the concepts list, there was no exercise using this technique in our dataset to verify how it would be clustered.

In the second experiment, we set the best number of topics to 12, but only six (considering the merge of topics 2 and 4) were valid. Standard topic modeling techniques with the augmented tokenizer yielded the best results. The LDA-based clustering demonstrates better interpretable results, as shown by our detailed topic analysis. When tested with an unseen dataset, the six topics comprised the main concepts present in the test set. Except for a few exceptions, most of them relate to the six main topics, and a few of them account for code specificity.

To understand how humans could read and interpret the extracted relationships, we asked 14 professors to contextualize CS1 concepts. Our results showed that professors could relate topics and concepts with 54 observations in the training set. Each cluster's main topic was explicit in the theme label task, and the clusters contained different concepts per topic. The professors can use this information to understand how often concepts cooccur while solving exercises. A future research direction is necessary to investigate if more specific clusters could be produced by increasing the number of samples or using more complex models.

We also investigated how an extracted Q-matrix from student performance differs from a Q-matrix made with specialists' input. We compared NMF and FDTF, two methods usually employed in student performance prediction tasks, in their ability to discover the latent skills only based on students' history (success or failure in solving the problems). To evaluate the item to skill mapping task and calculate the similarity between the extracted Q-matrices and the human-annotated matrix, the ANOSIM statistic showed to be a useful tool by providing a correlation coefficient and the number of clusters. As a result, none of the Q-matrices extracted from student performance data presented a statistical correlation with the human-annotated Q-matrix. The lack of relation between unsupervised and annotated Q-matrices was a result that was already seen in previous studies. In this thesis, we overcame this issue by using topic modeling techniques to create an unsupervised Q-matrix that is positively correlated with the original Q-matrix.

For interpreting the results and relating the extracted Q-matrix with student performance, a simple visualization was used, where it was possible to get a general overview of concept popularity and to pinpoint students' difficulties and achievements. This tool has the potential to be used to assist teachers in discovering domain models (in the absence of a validated Q-matrix) and edit or enhance learning materials based on students' achievement and struggles.

7.2 Future Work

The work presented in this thesis has opened several research directions concerning the use of these experiments in a real-world environment. We plan to keep improving the system based on feedback given by professors and students. We want to investigate how to assist students, professors, educators, and faculty in making sense of their own (in the case of students) or their classes' learning (for educators). This research's challenge is to process all collected data and present it in meaningful ways to improve learning outcomes. A first step was already done by creating a student dashboard. The next steps consist of understanding how the students can effectively use the dashboard to improve their studying strategies. The same idea can be applied to professors and faculty. They can use the collected data to understand if the tasks are compatible with the expected time to solve them and how the class responds to the content, if there are outlier students, among other questions. One feature requested by students is the inclusion of extra exercises for them to practice and study. We plan on creating a recommendation engine to provide personalized exercise recommendations.

We also foresee exploring the relation between different latent concepts as future work. Using student data to extract the Q-matrix does not correlate to the Q-matrix extracted from the professors or specialists' perspective. A future step would be to extract a common Q-matrix even when working with different sources. The shared Q-matrix could provide insights into understanding the main differences between the actual students' knowledge and the knowledge expected by the professors. To improve this understanding, an index to calculate the degree of agreement between students and teachers could be created as well as visualizations to highlight the differences between students' perceptions of difficulties and those perceived by teachers.

Finally, one of the outstanding contributions of this work is to collect data over several semesters. An observable target indicator should be set to measure the actual effects of using learning environments on learning. Especially in education, some achievements may take a long time to be observed. However, the proposal of this research is also to lay the ground for longitudinal experiments. A control experiment can be done to evaluate the effects on students. However, for professors and faculty staff, this kind of experiment is more complicated because it is harder to isolate confounding variables. Further research should be done on this matter.

References

[1] JAFFEE, D. “Asynchronous Learning: Technology and Pedagogical Strategy in a Distance Learning Course”, *Teaching Sociology*, v. 25, n. 4, pp. 262–277, out. 1997. ISSN: 0092-055X. doi: 10.2307/1319295.

[2] PERREAULT, H., WALDMAN, L., ALEXANDER, M., et al. “Overcoming Barriers to Successful Delivery of Distance-Learning Courses”, *J. Educ. Business*, v. 77, n. 6, pp. 313–318, jul. 2002. ISSN: 0883-2323. doi: 10.1080/08832320209599681.

[3] BOUHNIK, D., MARCUS, T. “Interaction in distance-learning courses”, *J. American Society Inf. Sci. Technol.*, v. 57, n. 3, pp. 299–305, nov. 2005. ISSN: 1532-2890. doi: 10.1002/asi.20277.

[4] SOUZA, S. D., FRANCO, V. S., COSTA, M. L. F., et al. “Educação a distância na ótica discente”, *Educação e Pesquisa*, v. 42, n. 1, pp. 99–114, mar. 2016. ISSN: 1517-9702. doi: 10.1590/s1517-9702201603133875.

[5] ROCHA, S. S. D., JOYE, C. R., MOREIRA, M. M. “A Educação a Distância na era digital: tipologia, variações, uso e possibilidades da educação online”, *Research, Society and Development*, v. 9, n. 6, abr 2020. ISSN: 2525-3409. doi: 10.33448/rsd-v9i6.3390.

[6] GOOGLE. “Google Trends”. 2020. Disponível em: <`https://trends.google.com.br/trends/explore?date=2016-07-02%202021-02-02&q=%2Fm%2F02h32`>. Online; accessed 02-February-2021.

[7] GEWIN, V. “Five tips for moving teaching online as COVID-19 takes hold”, *Nature*, v. 580, n. 7802, pp. 295–296, mar. 2020. doi: 10.1038/d41586-020-00896-7.

[8] COOK, D. A., DUPRAS, D. M. “A practical guide to developing effective web-based learning”, *J. General Internal Medicine*, v. 19, n. 6, pp. 698–707, jun. 2004. doi: 10.1111/j.1525-1497.2004.30029.x.

[9] INTERNATIONAL EDUCATIONAL DATA MINING SOCIETY. "educationaldatamining.org". `www.educationaldatamining.org`, 2021. [Online; accessed 13-Jan-2021].

[10] WITTEN, I. H., FRANK, E., HALL, M. A., et al. *Data Mining Practical Machine Learning Tools and Techniques*. 4 ed. Cambridge, MA, USA, Morgan Kaufmann (Publishers, Inc.), 2017. ISBN: 978-0-12-804291-5. doi: 10.1016/C2015-0-02071-8.

[11] BAKER, R. S. J. D., YACEF, K. "The State of Educational Data Mining in 2009 : A Review and Future Visions", *J. Educational Data Mining*, v. 1, n. 1, pp. 3–17, out. 2009. doi: 10.5281/zenodo.3554657.

[12] BAKER, R. S., D'MELLO, S. K., RODRIGO, M. M. T., et al. "Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive-affective states during interactions with three different computer-based learning environments", *Int. J. Human Comput. Stud.*, v. 68, n. 4, pp. 223–241, abr. 2010. ISSN: 1071-5819. doi: 10.1016/j.ijhcs.2009.12.003.

[13] PARDO, A., BARTIMOTE, K., SHUM, S. B., et al. "OnTask: Delivering Data-Informed, Personalized Learning Support Actions", *Learning Analytics*, v. 5, n. 3, pp. 235–249, dez. 2018. doi: 10.18608/jla.2018.53.15.

[14] ROSÉ, C. P., MCLAUGHLIN, E. A., LIU, R., et al. "Explanatory learner models: Why machine learning (alone) is not the answer", *British Journal of Educational Technology*, v. 50, n. 6, pp. 2943–2958, nov. 2019. doi: 10.1111/bjet.12858.

[15] JOVANOVIĆ, J., DAWSON, S., JOKSIMOVIĆ, S., et al. "Supporting actionable intelligence: reframing the analysis of observed study strategies". In: *Proceedings of the 10th International Conference on Learning Analytics & Knowledge*, LAK '20, pp. 161–170, New York, NY, USA, mar. 2020. Association for Computing Machinery. doi: 10.1145/3375462.3375474.

[16] BULL, S., KAY, J. "SMILI: a Framework for Interfaces to Learning Data in Open Learner Models, Learning Analytics and Related Fields", *Int J Artif Intell Educ*, v. 26, n. 1, pp. 293–331, mar. 2016. ISSN: 1560-4306. doi: 10.1007/s40593-015-0090-8.

[17] BAKER, R. S. "Stupid tutoring systems, intelligent humans", *International Journal of Artificial Intelligence in Education*, v. 26, n. 2, pp. 600–614, 2016. doi: 10.1007/s40593-016-0105-0.

[18] NIKULA, U., GOTEL, O., KASURINEN, J. “A Motivation Guided Holistic Rehabilitation of the First Programming Course”, *ACM Trans. Comput. Educ.*, v. 11, n. 4, nov. 2011. doi: 10.1145/2048931.2048935.

[19] ROGALSKI, J., SAMURCAY, R. “Task Analysis and Cognitive Model as a Framework to Analyse Environments for Learning Programming”. In: Lemut, E., du Boulay, B., Dettori, G. (Eds.), *Cognitive models and intelligent environments for learning programming*, v. 111, pp. 6–19, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. doi: 10.1007/978-3-662-11334-9_2.

[20] ABDUL-RAHMAN, S.-S., DU BOULAY, B. “Learning programming via worked-examples: Relation of learning styles to cognitive load”, *Comput. Human Behav.*, v. 30, pp. 286 – 298, 2014. ISSN: 0747-5632. doi: 10.1016/j.chb.2013.09.007.

[21] DAMASCENO, A., ALMEIDA, C., FERNANDES, W., et al. “What Can Be Found from Student Interaction Logs of Online Courses Offered in Brazil”, *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, v. 30, n. 1, pp. 1641, nov. 2019. doi: 10.5753/cbie.sbie.2019.1641.

[22] LAN, A. S., WATERS, A. E., STUDER, C., et al. “Sparse Factor Analysis for Learning and Content Analytics”, *J. Mach. Learn. Res.*, v. 15, n. 1, pp. 1959–2008, 2014. ISSN: 1532-4435.

[23] HUNDHAUSEN, C. D., OLIVARES, D. M., CARTER, A. S. “IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda”, *ACM Trans. Comput. Educ.*, v. 17, n. 3, pp. 11:1–11:26, ago. 2017. doi: 10.1145/3105759.

[24] SCHWENDIMANN, B. A., RODRIGUEZ-TRIANA, M. J., VOZNIUK, A., et al. “Perceiving learning at a glance: A systematic literature review of learning dashboard research”, *IEEE Trans. Learn. Technol.*, v. 10, n. 1, pp. 30–41, jan. 2017. doi: 10.1109/TLT.2016.2599522.

[25] DESMARAIS, M. C. “Mapping Question Items to Skills with Non-Negative Matrix Factorization”, *SIGKDD Explorations Newslett.*, v. 13, n. 2, pp. 30–36, maio 2012. doi: 10.1145/2207243.2207248.

[26] LEE, D. D., SEUNG, H. S. “Learning the parts of objects by non-negative matrix factorization”, *Nature*, v. 401, pp. 788–791, out. 1999. doi: 10.1038/44565.

[27] BLEI, D. M., NG, A. Y., JORDAN, M. I. “Latent Dirichlet Allocation”, *J. Mach. Learn. Res.*, v. 3, pp. 993–1022, jan. 2003.

[28] STEYVERS, M., GRIFFITHS, T. “Probabilistic Topic Models”. In: Landauer, T., McNamara, S. D., Kintsch, W. (Eds.), *Handbook of Latent Semantic Analysis*, 1 ed., Psychology Press, cap. 21, pp. 427–448, New York, NY, USA, 2007. doi: 10.4324/9780203936399.

[29] HOFMANN, T. “Probabilistic Latent Semantic Analysis”. In: Laskey, K. B., Prade, H. M. (Eds.), *Proc. 15th Conf. Uncertainty Artificial Intelligence*, pp. 289–296, Stockholm, Sweden, 30 jul.–1 ago. 1999.

[30] MORAES, L. O., PEDREIRA, C. E. “Designing an Intelligent Tutoring System Across Multiple Classes”. In: *4th Educational Data Mining in Computer Science Education Workshop*, 10 jul. 2020.

[31] MORAES, L. O., PEDREIRA, C. E. “Clustering Introductory Computer Science Exercises Using Topic Modeling Methods”, *IEEE Trans. Learn. Technol.*, 2021. doi: 10.1109/TLT.2021.3056907.

[32] VANLEHN, K. “The Behavior of Tutoring Systems”, *Int. J. Artif. Intell. Ed.*, v. 16, n. 3, pp. 227–265, ago. 2006. ISSN: 1560-4292.

[33] SELF, J. “Theoretical Foundations for Intelligent Tutoring Systems”, *J. Artif. Intell. Educ.*, v. 1, n. 4, pp. 3–14, set. 1990. ISSN: 1043-1020.

[34] IHANTOLA, P., VIHAVAINEN, A., AHADI, A., et al. “Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies”. In: *Proc. 2015 ITiCSE Working Group Report*, pp. 41–63, Vilnius, Lithuania, jul. 2015. doi: 10.1145/2858796.2858798.

[35] LUXTON-REILLY, A., SIMON, ALBLUWI, I., et al. “Introductory Programming: A Systematic Literature Review”. In: *Proc. Companion 23rd Annu. ACM Conf. Innovation and Technology in Computer Science Education*, ITiCSE 2018 Companion, p. 55–106, Larnaca, Cyprus, jul. 2018. doi: 10.1145/3293881.3295779.

[36] ROBINSON, D. “The Incredible Growth of Python”. `https://stackoverflow.blog/2017/09/06/incredible-growth-python/`, 2017. Online; accessed 01-November-2020.

[37] STACK OVERFLOW. “Developer Survey Results 2018”. `https://insights.stackoverflow.com/survey/2018/`, 2019. Online; accessed 12-June-2020.

[38] MASON, R., SIMON. "Introductory Programming Courses in Australasia in 2016". In: *Proc. 19th Australasian Computing Educational Conference*, p. 81–89, Geelong, VIC, Australia, jan. 2017. doi: 10.1145/3013499.3013512.

[39] HOVEMEYER, D., SPACCO, J. "CloudCoder: A web-based programming exercise system", *J. Comput. Sci. in Colleges*, 2013. ISSN: 1937-4771.

[40] PAPANCEA, A., SPACCO, J., HOVEMEYER, D. "An open platform for managing short programming exercises". In: *Proc. 2013 ACM Conf. Int. Computing Education Research*, pp. 47–52, San Diego, CA, USA, 12–14 ago. 2013. doi: 10.1145/2493394.2493401.

[41] PANAMALAI MURALI, K. *CodeWorkout: Design and implementation of an online drill-and-practice system for introductory programming.* Thesis, Virginia Tech, jun. 2016. Disponível em: <`https://vtechworks.lib.vt.edu/handle/10919/81072`>.

[42] ZINGARO, D., CHERENKOVA, Y., KARPOVA, O., et al. "Facilitating code-writing in PI classes". In: *Proc. 44th ACM Technical Symp. Computer Science Education*, pp. 585–590, Denver, CO, USA, 6–9 mar. 2013. doi: 10.1145/2445196.2445369.

[43] DAGOSTINI, J., LIMA, M. V. D. M., BEZ, J. L., et al. "URI Online Judge Blocks: Construindo soluções em uma plataforma online de programação", *Brazilian Symp. Computers Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, v. 29, n. 1, pp. 168, out. 2018. doi: 10.5753/cbie.sbie.2018.168. 1.

[44] BEZ, J. L., TONIN, N., SELIVON, M. "URI Online Judge Academic: Integração e consolidação da ferramenta no processo de ensino/aprendizagem". In: *Anais do XXIII Workshop sobre Educação em Computação*, pp. 188–195, 20–23 jul. 2015. doi: 10.5753/wei.2015.10235.

[45] SORVA, J., SIRKIÄ, T. "UUhistle: a software tool for visual program simulation". In: *Proc. 10th Koli Calling Int. Conf. Computing Education Research*, pp. 49–54, Koli, Finland, out. 2010. doi: 10.1145/1930464.1930471.

[46] EDWARDS, S. H., TILDEN, D. S., ALLEVATO, A. "Pythy: improving the introductory python programming experience". In: *Proc. 45th ACM technical symposium on Computer science education*, pp. 641–646, Atlanta, GA, USA, 5–8 mar. 2014. doi: 10.1145/2538862.2538977.

[47] EDWARDS, S. H., PEREZ-QUINONES, M. A. "Web-CAT: automatically grading programming assignments". In: *Proc. 13th Annu. Conf. on Innovation and Technology Computer Science Education*, p. 328, Madrid, Spain, 30 jun.–2jul. 2008. doi: 10.1145/1384271.1384371.

[48] HU, Y.-H., LO, C.-L., SHIH, S.-P. "Developing early warning systems to predict students' online learning performance", *Comput. in Human Behavior*, v. 36, pp. 469–478, jul. 2014. ISSN: 07475632. doi: 10.1016/j.chb.2014.04.002. Disponível em: <`https://linkinghub.elsevier.com/retrieve/pii/S0747563214002118`>.

[49] MARBOUTI, F., DIEFES-DUX, H. A., MADHAVAN, K. "Models for early prediction of at-risk students in a course using standards-based grading", *Computers and Education*, v. 103, pp. 1–15, dez. 2016. doi: 10.1016/j.compedu.2016.09.005.

[50] COSTA, E. B., FONSECA, B., SANTANA, M. A., et al. "Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses", *Comput. Human Behavior*, v. 73, pp. 247–256, ago. 2017. doi: 10.1016/j.chb.2017.01.047.

[51] ALAMRI, A., ALSHEHRI, M., CRISTEA, A., et al. "Predicting MOOCs Dropout Using Only Two Easily Obtainable Features from the First Week's Activities". In: Coy, A., Hayashi, Y., Chang, M. (Eds.), *Intelligent Tutoring Systems. Lecture Notes in Computer Science*, v. 11528, *Lecture Notes in Computer Science*, pp. 163–173, Cham, 2019. ISBN: 978-3-030-22244-4. doi: 10.1007/978-3-030-22244-4_20.

[52] CHEN, W., BRINTON, C. G., CAO, D., et al. "Early Detection Prediction of Learning Outcomes in Online Short-Courses via Learning Behaviors", *IEEE Trans. on Learn. Technol.*, v. 12, n. 1, pp. 44–58, jan. 2019. ISSN: 1939-1382. doi: 10.1109/TLT.2018.2793193.

[53] GRAY, C. C., PERKINS, D. "Utilizing early engagement and machine learning to predict student outcomes", *Computers and Education*, v. 131, pp. 22–32, abr. 2019. doi: 10.1016/j.compedu.2018.12.006.

[54] PEREIRA, F. D., OLIVEIRA, E., CRISTEA, A., et al. "Early Dropout Prediction for Programming Courses Supported by Online Judges". In: Isotani, S., Millán, E., Ogan, A., et al. (Eds.), *Artificial Intelligence in Education*, Lecture Notes in Computer Science, pp. 67–72, Cham, 2019. Springer International Publishing. doi: 10.1007/978-3-030-23207-8_13.

[55] PEREIRA, F. D., OLIVEIRA, E. H. T., FERNANDES, D., et al. "Early performance prediction for CS1 course students using a combination of machine learning and an evolutionary algorithm". In: *2019 IEEE 19th Int. Conf. on Advanced Learning Technologies*, pp. 183–184, Maceió, Brazil, 15–18 jul. 2019. doi: 10.1109/ICALT.2019.00066.

[56] ANDRES, J. M. L., BAKER, R. S., GAŠEVIĆ, D., et al. "Studying MOOC completion at scale using the MOOC replication framework". In: *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*, pp. 71–78, Sydney New South Wales Australia, mar. 2018. ACM. doi: 10.1145/3170358.3170369.

[57] AL-SHABANDAR, R., HUSSAIN, A. J., LIATSIS, P., et al. "Analyzing Learners Behavior in MOOCs: An Examination of Performance and Motivation Using a Data-Driven Approach", *IEEE Access*, v. 6, pp. 73669–73685, 2018. doi: 10.1109/ACCESS.2018.2876755.

[58] J. SHEARD *ET AL.* "Exploring Programming Assessment Instruments: A Classification Scheme for Examination Questions". In: *Proc. 7th Int. Computing Education Research Workshop*, pp. 33–38, Providence, RI, USA, 8–9 ago. 2011. doi: 10.1145/2016911.2016920.

[59] PETERSEN, A., CRAIG, M., ZINGARO, D. "Reviewing CS1 Exam Question Content". In: Cortina, T. J., Walker, E. L., King, L. A. S., et al. (Eds.), *Proc. 42nd ACM Technical Symp. Computer Science Education*, pp. 631–636, Dallas, TX, USA, 9–12 mar. 2011. doi: 10.1145/1953163.1953340.

[60] FEI, T., HENG, W. J., TOH, K. C., et al. "Question classification for e-learning by artificial neural network". In: *Proc. 2003 Joint Conf. 4th Int. Conf. Information, Communications and Signal Processing and 4th Pacific Rim Conf. Multimedia*, pp. 1757–1761, Singapore, 15–18 dez. 2003. doi: 10.1109/ICICS.2003.1292768.

[61] SUPRAJA, S., HARTMAN, K., KHONG, A. W. H. "Toward the Automatic Labeling of Course Questions for Ensuring their Alignment with Learning Outcomes". In: Hu, X., Barnes, T., Hershkovitz, A., et al. (Eds.), *Proc. 10th Int. Conf. Educational Data Mining*, pp. 56–63, Wuhan, China, 25–28 jun. 2017.

[62] GODEA, A., TULLEY-PATTON, D., BARBEE, S., et al. "Classifying Educational Questions Based on the Expected Characteristics of Answers". In: Penstein Rosé, C., Martínez-Maldonado, R., Hoppe, H. U., et al. (Eds.),

Int. Conf. Artificial Intelligence Education, pp. 104–108, London, United Kingdom, 20 jun. 2018. doi: 10.1007/978-3-319-93846-2_20.

[63] GONZÁLEZ, P., GIBAJA, E., ZAPATA, A., et al. "Towards Automatic Classification of Learning Objects: Reducing the Number of Used Features". In: Hu, X., Barnes, T., Hershkovitz, A., et al. (Eds.), *Proc. 10th Int. Conf. Educational Data Mining*, pp. 394–395, Wuhan, China, 25–28 jun. 2017.

[64] NUNN, S., AVELLA, J. T., KANAI, T., et al. "Learning Analytics Methods, Benefits, and Challenges in Higher Education: A Systematic Literature Review", *Online Learning*, v. 20, n. 2, jun. 2016. doi: 10.24059/olj.v20i2.790.

[65] DUTT, A., ISMAIL, M. A., HERAWAN, T. "A Systematic Review on Educational Data Mining", *IEEE Access*, v. 5, pp. 15991–16005, jan. 2017. doi: 10.1109/ACCESS.2017.2654247.

[66] RODRIGUES, M. W., ISOTANI, S., ZÁRATE, L. E. "Educational Data Mining: A review of evaluation process in the e-learning", *Telematics and Inform.*, v. 35, n. 6, pp. 1701–1717, set. 2018. doi: 10.1016/j.tele.2018.04.015.

[67] HERNÁNDEZ-BLANCO, A., HERRERA-FLORES, B., TOMÁS, D., et al. "A Systematic Review of Deep Learning Approaches to Educational Data Mining", *Complexity*, v. 2019, maio 2019. doi: 10.1155/2019/1306039.

[68] ALDOWAH, H., AL-SAMARRAIE, H., FAUZY, W. M. "Educational data mining and learning analytics for 21st century higher education: A review and synthesis", *Telematics and Inform.*, v. 37, pp. 13–49, abr. 2019. doi: 10.1016/j.tele.2019.01.007.

[69] TRIVEDI, S., PARDOS, Z., SÁRKÖZY, G., et al. "Spectral Clustering in Educational Data Mining". In: Pechenizkiy, M., Calders, T., Conati, C., et al. (Eds.), *Proc. 4th Int. Conf. Educational Data Mining*, pp. 129–138, Eindhoven, the Netherlands, 6–8 jul. 2011.

[70] FIGUEIRA, A. P. D. B. B. R. "A repository with semantic organization for educational content". In: *2008 8th IEEE Int. Conf. Advanced Learning Technologies*, pp. 114–116, Santander, Spain, 1–5 jul. 2008. doi: 10.1109/ICALT.2008.60.

[71] HUANG, J., PIECH, C., NGUYEN, A., et al. "Syntactic and functional variability of a million code submissions in a machine learning MOOC". In:

Walker, E., Looi, C.-K. (Eds.), *Proc. Workshops 16th Int. Conf. Artificial Intelligence Education 2013*, v. 1, pp. 25–32, Memphis, TN, USA, 9–13 jul. 2013. doi: 10.1007/978-3-642-39112-5.

[72] NGUYEN, A., PIECH, C., HUANG, J., et al. “Codewebs: Scalable Homework Search for Massive Open Online Programming Courses”. In: *Proc. 23rd Int. Conf. World Wide Web*, pp. 491–502, Seoul, Korea, 7–11 abr. 2014. doi: 10.1145/2566486.2568023.

[73] PAASSEN, B., MOKBEL, B., HAMMER, B. “Adaptive structure metrics for automated feedback provision in intelligent tutoring systems”, *Neurocomputing*, v. 192, pp. 3–13, jun. 2016. doi: 10.1016/j.neucom.2015.12.108.

[74] PRICE, T., ZHI, R., BARNES, T. “Evaluation of a Data-driven Feedback Algorithm for Open-ended Programming”. In: Hu, X., Barnes, T., Hershkovitz, A., et al. (Eds.), *Proc. 10th Int. Conf. Educational Data Mining*, pp. 192–197, Wuhan, China, 25–28 jun. 2017.

[75] MOKBEL, B., GROSS, S., PAASSEN, B., et al. “Domain-Independent Proximity Measures in Intelligent Tutoring Systems”. In: D’Mello, S. K., Calvo, R. A., Olney, A. (Eds.), *Proc. 6th Int. Conf. Educational Data Mining*, pp. 334–335, Memphis, TN, USA, 6–9 jul. 2013.

[76] LU, Y., HSIAO, I.-H. “Modeling Semantics between Programming Codes and Annotations”. In: Lee, D., Sastry, N., Weber, I. (Eds.), *Proc. 29th Hypertext and Social Media*, pp. 101–105, Baltimore, MD, USA, 9–12 jul. 2018. doi: 10.1145/3209542.3209578.

[77] CHEN, T. H., THOMAS, S. W., HASSAN, A. E. “A survey on the use of topic models when mining software repositories”, *Empirical Softw. Eng.*, v. 21, n. 5, pp. 1843–1919, out. 2016. doi: 10.1007/s10664-015-9402-8.

[78] AZCONA, D., ARORA, P., HSIAO, I.-H., et al. “User2code2vec: Embeddings for Profiling Students Based on Distributional Representations of Source Code”. In: *Proc. 9th Int. Learning Analytics and Knowledge Conf.*, pp. 86–95, Tempe, AZ, USA, 4–8 mar. 2019. doi: 10.1145/3303772.3303813.

[79] CHEN, Q., YAO, L., YANG, J. “Short text classification based on LDA topic model”. In: *2016 Int. Conf. Audio, Language and Image Processing*, pp. 749–753, Shanghai, China, 11-12 jul. 2016. doi: 10.1109/ICALIP.2016.7846525.

[80] ABOLHASSANI, N., RAMASWAMY, L. "Extracting Topics from Semi-structured Data for Enhancing Enterprise Knowledge Graphs". In: Wang, X., Gao, H., Iqbal, M., et al. (Eds.), *Int. Conf. Collaborative Computing: Networking, Applications and Worksharing*, v. 292, pp. 101–117, London, United Kingdom, 19–22 ago. 2019. doi: 10.1007/978-3-030-30146-0_8.

[81] HSIAO, I.-H., AWASTHI, P. "Topic Facet Modeling: Semantic Visual Analytics for Online Discussion Forums". In: Baron, J., Lynch, G., Maziarz, N., et al. (Eds.), *Proc. 5th Int. Conf. Learning Analytics Knowledge*, pp. 231–235, Poughkeepsie, NY, USA, 16–20 mar. 2015. doi: 10.1145/2723576.2723613.

[82] HSIAO, I.-H., LIN, Y.-L. "Enriching programming content semantics: An evaluation of visual analytics approach", *Comput. Human Behav.*, v. 72, pp. 771–782, jul. 2017. doi: 10.1016/j.chb.2016.10.012.

[83] ZHAO *ET AL.* "Comparing Twitter and Traditional Media Using Topic Models". In: Clough, P., Foley, C., Gurrin, C., et al. (Eds.), *Proc. 33rd European Conf. Advances Information Retrieval*, v. 6611, pp. 338–349, Dublin, Ireland, 18–21 abr. 2011. doi: 10.1007/978-3-642-20161-5_34.

[84] STEYVERS, M., SMYTH, P., ROSEN-ZVI, M., et al. "Probabilistic Author-Topic Models for Information Discovery". In: Kohavi, R., Gehrkeand, J., DuMouchel, W., et al. (Eds.), *Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 306–315, Seattle, WA, USA, 22-25 ago. 2004. doi: 10.1145/1014052.1014087.

[85] ROSEN-ZVI, M., CHEMUDUGUNTA, C., GRIFFITHS, T., et al. "Learning Author-Topic Models from Text Corpora", *ACM Trans. Inf. Syst.*, v. 28, n. 1, pp. 1–38, jan. 2010. doi: 10.1145/1658377.1658381. Art. no. 4, doi: 10.1145/1658377.1658381.

[86] LI, S., LI, J., PAN, R. "Tag-Weighted Topic Model for Mining Semi-Structured Documents". In: Rossi, F. (Ed.), *Proc. 23rd Int. Joint Conf. Artificial Intelligence*, pp. 2855–2861, Beijing, China, 3–9 ago. 2013.

[87] WENG, J., LIM, E.-P., JIANG, J., et al. "TwitterRank: Finding Topic-Sensitive Influential Twitterers". In: Davison, B. D., Suel, T., Craswell, N., et al. (Eds.), *Proc. 3rd ACM Int. Conf. Web Search and Data Mining*, pp. 261–270, New York, NY, USA, 3–6 fev. 2010. doi: 10.1145/1718487.1718520.

[88] SYED, S., SPRUIT, M. “Full-Text or Abstract? Examining Topic Coherence Scores Using Latent Dirichlet Allocation”. In: *2017 IEEE Int. Conf. Data Science and Advanced Analytics*, pp. 165–174, Tokyo, Japan, 19–21 out. 2017. doi: 10.1109/DSAA.2017.61.

[89] CORBETT, A. T., ANDERSON, J. R. “Knowledge tracing: Modeling the acquisition of procedural knowledge”, *User Model. User-Adapted Interact.*, v. 4, pp. 253–278, dez. 1994. doi: 10.1007/BF01099821.

[90] YUDELSON, M. V., KOEDINGER, K. R., GORDON, G. J. “Individualized bayesian knowledge tracing models”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 171–180, 2013. doi: 10.1007/978-3-642-39112-5_18.

[91] WILSON, K. H., KARKLIN, Y., HAN, B., et al. “Back to the basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation”. In: *Proc. 9th International Conference on Educational Data Mining*, p. 6, 2016.

[92] VANLEHN, K., NIU, Z., SILER, S., et al. “Student Modeling from Conventional Test Data: A Bayesian Approach without Priors”. In: Goettl, B. P., Halff, H. M., Redfield, C. L., et al. (Eds.), *Intelligent Tutoring Systems*, pp. 434–443, Berlin, Heidelberg, 1998. doi: 10.1007/3-540-68716-5_49.

[93] QIU, Y., QI, Y., LU, H., et al. “Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing.” In: Pechenizkiy, M., Calders, T., Conati, C., et al. (Eds.), *Proc. 4th International Conference on Educational Data Mining*, pp. 139–148, Eindhoven, the Netherlands, jul. 2011.

[94] PELÁNEK, R. “Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques”, *User Model. User-Adapted Interact.*, v. 27, pp. 313–350, 2017. ISSN: 15731391. doi: 10.1007/s11257-017-9193-2.

[95] PARDOS, Z. A., HEFFERNAN, N. T. “Modeling individualization in a Bayesian networks implementation of knowledge tracing”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 6075, pp. 255–266, 2010. doi: 10.1007/978-3-642-13470-8_24.

[96] NEDUNGADI, P., REMYA, M. S. “Incorporating forgetting in the Personalized, Clustered, Bayesian Knowledge Tracing (PC-BKT) model”.

In: *Proc. 2015 Int. Conf. cognitive computing and information processing (CCIP)*, Noida, India, maio 2015. ISBN: 978-1-4799-7171-8. doi: 10.1109/CCIP.2015.7100688.

[97] LIN, C., CHI, M. “Intervention-BKT: Incorporating instructional interventions into Bayesian knowledge tracing”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9684, pp. 208–218, 2016. ISBN: 978-3-319-39582-1. doi: 10.1007/978-3-319-39583-8_20. ISSN: 16113349.

[98] VIE, J.-J., KASHIMA, H. “Knowledge Tracing Machines: Factorization Machines for Knowledge Tracing”. In: *Proc. AAAI Conf. Artificial Intelligence*, v. 33, pp. 750–757, nov. 2018. doi: 10.1609/aaai.v33i01.3301750.

[99] PAVLIK, P. I., CEN, H., KOEDINGER, K. R. “Performance factors analysis - A new alternative to knowledge tracing”. In: *Frontiers in Artificial Intelligence and Applications*, v. 200, pp. 531 – 538, 2009. doi: 10.3233/978-1-60750-028-5-531.

[100] GALYARDT, A., GOLDIN, I. “Convergent Validity of a Student Model: Recent-Performance Factors Analysis”. In: *Proc. 8th Int. Conf. Educational Data Mining*, pp. 548–551, 2015.

[101] BARNES, T. M. *The Q-matrix Method of Fault-Tolerant Teaching in Knowledge Assessment and Data Mining.* Tese de Doutorado, North Carolina State University, 2003.

[102] TATSUOKA, K. K. “Toward an integration of item-response theory and cognitive error diagnosis.” In: *Diagnostic monitoring of skill and knowledge acquisition.*, Lawrence Erlbaum Associates, Inc, pp. 453–488, Hillsdale, NJ, US, 1990. ISBN: 0-89859-992-X (Hardcover).

[103] CEN, H., KOEDINGER, K., JUNKER, B. “Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement”. In: Hutchison, D., Kanade, T., Kittler, J., et al. (Eds.), *Intelligent Tutoring Systems*, v. 4053, Springer Berlin Heidelberg, pp. 164–175, Berlin, Heidelberg, 2006. doi: 10.1007/11774303_17.

[104] WINTERS, T., SHELTON, C., PAYNE, T., et al. “Topic extraction from item-level grades”. In: *American Association for Artificial Intelligence 2005 Workshop on Educational Datamining*, 2005.

[105] DESMARAIS, M. “Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization”. In: Pechenizkiy, M., Calders, T., Conati, C., et al. (Eds.), *Proc. 4th Int. Conf. Educational Data Mining*, pp. 41–50, Eindhoven, the Netherlands, 6–8 jul. 2011.

[106] GONZÁLEZ-BRENES, J. “Modeling Skill Acquisition Over Time with Sequence and Topic Modeling”. In: Lebanon, G., Vishwanathan, S. V. N. (Eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, v. 38, *Proceedings of Machine Learning Research*, pp. 296–305, San Diego, California, USA, 09–12 May 2015. PMLR.

[107] DOAN, T.-N., SAHEBI, S. “Rank-Based Tensor Factorization for Student Performance Prediction”. In: *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*, v. 288, p. 293. ERIC, 2019.

[108] SAHEBI, S., LIN, Y.-R., BRUSILOVSKY, P. “Tensor Factorization for Student Modeling and Performance Prediction in Unstructured Domain.” *International Educational Data Mining Society*, 2016.

[109] DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO - UFRJ. “Python UFRJ”. `https://dcc.ufrj.br/~pythonufrj/python1_37.html`, 2019. Online; accessed 01-November-2020.

[110] DELGADO, C., DA SILVA, J., MASCARENHAS, F., et al. “The teaching of functions as the first step to learn imperative programming”. In: *Anais do Workshop sobre Educação em Computação (WEI)*, pp. 388–397. Sociedade Brasileira de Computação - SBC, jan. 2016. doi: 10.5753/wei.2016.9683.

[111] SOMMERVILLE, I. *Engenharia de software*. 9 ed. São Paulo, Brasil, Pearson Prentice Hall, 2011. ISBN: 9788579361081.

[112] COHN, M. *User Stories Applied: For Agile Software Development*. 1 ed. IL, USA, Addison-Wesley Professional, mar. 2004.

[113] LUCASSEN, G., DALPIAZ, F., WERF, J. M. E. M. V. D., et al. “The Use and Effectiveness of User Stories in Practice”. In: *International working conference on requirements engineering: Foundation for software quality*, v. 9619, *LNCS*, pp. 205–222, 2016. ISBN: 978-3-319-30282-9. doi: 10.1007/978-3-319-30282-9_14.

[114] PRESSMAN, R. S., MAXIM, B. *Software Engineering: A Practitioner's Approach.* 8ª edição ed. New York, NY, McGraw-Hill Science/Engineering/Math, jan. 2014. ISBN: 978-0-07-802212-8.

[115] PRATUSEVICH, M. "Practice Python". `www.practicepython.org`, 2017. [Online; accessed 07-Jan-2021].

[116] SENTANCE, S., MCNICOL, A. "Python School". `https://pythonschool.net/`, 2016. [Online; accessed 07-Jan-2021].

[117] HU, J. "Python Programming Exercises". `https://github.com/zhiwehu/Python-programming-exercises`, 2018. [Online; accessed 07-Jan-2021].

[118] W3RESOURCE. "W3Resource". `https://www.w3resource.com/python/python-tutorial.php`, 2018. [Online; accessed 07-Jan-2021].

[119] CHOFFIN, B., POPINEAU, F., BOURDA, Y., et al. "DAS3H: Modeling Student Learning and Forgetting for Optimally Scheduling Distributed Practice of Skills". In: *Proceedings of 12th International Conference on Educational Data Mining*, pp. 29–38, Montreal, Canada, jul. 2019.

[120] LALWANI, A., AGRAWAL, S. "What Does Time Tell? Tracing the Forgetting Curve Using Deep Knowledge Tracing". In: Isotani, S., Millán, E., Ogan, A., et al. (Eds.), *International Conference on Artificial Intelligence in Education*, LNCS, pp. 158–162, jun. 2019. doi: 10.1007/978-3-030-23207-8_30.

[121] NAGATANI, K., CHEN, Y. Y., ZHANG, Q., et al. "Augmenting knowledge tracing by considering forgetting behavior". In: *Proceedings of The World Wide Web Conference*, WWW '19, San Francisco, CA, USA, maio 2019. doi: 10.1145/3308558.3313565.

[122] BREIMAN, L., FRIEDMAN, J., OLSHEN, R., et al. *Classification and regression trees.* 1st ed. Boca Raton, FL, USA, CRC Press, 1984.

[123] MALMI, L., SHEARD, J., SIMON, et al. "Characterizing Research in Computing Education: A Preliminary Analysis of the Literature". In: *Proc. 6th Int. Workshop on Computing Education Research*, p. 3–12, Aarhus, Denmark, 9–10 ago. 2010. doi: 10.1145/1839594.1839597.

[124] SALTON, G., MCGILL, M. J. *Introduction to Modern Information Retrieval.* New York, NY, USA, McGraw-Hill, 1986. ISBN: 0070544840.

[125] ZHANG, W., YOSHIDA, T., TANG, X. "A comparative study of TF*IDF, LSI and multi-words for text classification", *Expert Syst. Appl.*, v. 38, n. 3, pp. 2758–2765, mar. 2011. doi: 10.1016/j.eswa.2010.08.066.

[126] YAN, X., GUO, J., LIU, S., et al. "Clustering Short Text Using Ncut-Weighted Non-Negative Matrix Factorization". In: *Proc. 21st ACM Int. Conf. Information and Knowledge Management*, pp. 2259–2262, Maui, HI, USA, 29 out.–2 nov. 2012. doi: 10.1145/2396761.2398615.

[127] CICHOCKI, A., PHAN, A. H. "Fast Local Algorithms for Large Scale Nonnegative Matrix and Tensor Factorizations", *IEICE Trans. Fundam. Electron., Commun. and Comput. Sci.*, v. E92.A, n. 3, pp. 708–721, mar. 2009. doi: 10.1587/transfun.E92.A.708.

[128] O'CALLAGHAN, D., GREENE, D., CARTHY, J., et al. "An analysis of the coherence of descriptors in topic modeling", *Expert Syst. Appl.*, v. 42, n. 13, pp. 5645–5657, ago. 2015. doi: 10.1016/j.eswa.2015.02.055.

[129] CHEN, Y., ZHANG, H., LIU, R., et al. "Experimental explorations on short text topic mining between LDA and NMF based Schemes", *Knowledge-Based Syst.*, v. 163, pp. 1–13, jan. 2019. doi: 10.1016/j.knosys.2018.08.011.

[130] BERGSTRA, J., BENGIO, Y. "Random Search for HyperParameter Optimization", *J. Mach. Learn. Res.*, v. 13, pp. 281–305, fev. 2012.

[131] CHUANG, J., GUPTA, S., MANNING, C., et al. "Topic model diagnostics: Assessing domain relevance via topical alignment". In: Dasgupta, S., McAllester, D. (Eds.), *Proc. 30th Int. Conf. Machine Learning*, pp. 612–620, 16–21 jun. 2013.

[132] WANG, C., BLEI, D. M. "Collaborative Topic Modeling for Recommending Scientific Articles". In: *Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 448–456, San Diego, CA, USA, 21–24 ago. 2011. doi: 10.1145/2020408.2020480.

[133] MIMNO, D., WALLACH, H. M., TALLEY, E., et al. "Optimizing Semantic Coherence in Topic Models". In: Barzilay, R., Johnson, M. (Eds.), *Proc. Conf. Empirical Methods Natural Language Processing*, pp. 262–272, Edinburgh, Scotland, United Kingdom, 27–29 jul. 2011.

[134] ALETRAS, N., STEVENSON, M. "Evaluating Topic Coherence Using Distributional Semantics". In: Koller, A., Erk, K. (Eds.), *Proc. 10th Int. Conf. Computational Semantics*, pp. 13–22, Potsdam, Germany, 19–22 mar. 2013.

[135] LAU, J. H., NEWMAN, D., BALDWIN, T. "Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality". In: Wintner, S., Goldwater, S., Riezler, S. (Eds.), *Proc. 14th Conf. European Chapter Association Computational Linguistics*, pp. 530–539, Gothenburg, Sweden, 26–30 abr. 2014. doi: 10.3115/v1/E14-1056.

[136] NEWMAN, D., LAU, J. H., GRIESER, K., et al. "Automatic Evaluation of Topic Coherence". In: *Human Language Technologies: 2010 Annu. Conf. North American Chapter Association Computational Linguistics*, pp. 100–108, Los Angeles, CA, USA, 1–6 jun. 2010.

[137] JOINT TASK FORCE ON COMPUTING CURRICULA, ASSOCIATION FOR COMPUTING MACHINERY (ACM) AND IEEE COMPUTER SOCIETY. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* New York, NY, USA, Association for Computing Machinery, dez. 2013. ISBN: 9781450323093. doi: 10.1145/2534860.

[138] SCHULTE, C., BENNEDSEN, J. "What Do Teachers Teach in Introductory Programming?" In: Anderson, R., Fincher, S. A., Guzdial, M. (Eds.), *Proc. 2nd Int. Workshop Computing Education Research*, pp. 17–28, 9–10 set. 2006. doi: 10.1145/1151588.1151593.

[139] ROBINS, A., HADEN, P., GARNER, S. "Problem Distributions in a CS1 Course". In: Tolhurst, D., Mann, S. (Eds.), *Proc. 8th Australasian Conf. Computing Education*, v. 52, pp. 165–173, Hobart, Australia, jan. 2006.

[140] GOLDMAN *ET AL.* "Setting the Scope of Concept Inventories for Introductory Computing Subjects", *ACM Trans. Comput. Educ.*, v. 10, n. 2, pp. 1–29, jun. 2010. Art. no. 5, doi: 10.1145/1789934.1789935.

[141] CHERENKOVA, Y., ZINGARO, D., PETERSEN, A. "Identifying Challenging CS1 Concepts in a Large Problem Dataset". In: Dougherty, J., Nagel, K., Decker, A., et al. (Eds.), *Proc. 45th ACM Technical Symp. Computer Science Education*, pp. 695–700, Atlanta, GA, USA, 5–8 mar. 2014. doi: 10.1145/2538862.2538966.

[142] CHANG, J., GERRISH, S., WANG, C., et al. "Reading Tea Leaves: How Humans Interpret Topic Models". In: Bengio, Y., Schuurmans, D., Lafferty, J. D., et al. (Eds.), *Advances in Neural Information Processing Systems 22, Proc. 23rd Annu. Conf. Neural Information Processing Systems*, pp. 288–296, Vancouver, British Columbia, Canada, 7–10 dez. 2009.

[143] MAHMOUD, A., BRADSHAW, G. “Semantic topic models for source code analysis”, *Empirical Softw. Eng.*, v. 22, n. 4, pp. 1965–2000, ago. 2017. doi: 10.1007/s10664-016-9473-1.

[144] ANDERSON, M. J., WALSH, D. C. I. “PERMANOVA, ANOSIM, and the Mantel test in the face of heterogeneous dispersions: What null hypothesis are you testing?” *Ecological Monographs*, v. 83, n. 4, pp. 557–574, 2013. ISSN: 1557-7015. doi: 10.1890/12-2010.1.

[145] CHAPMAN, M., UNDERWOOD, A. “Ecological patterns in multivariate assemblages:information and interpretation of negative values in ANOSIM tests”, *Mar. Ecol. Prog. Ser.*, v. 180, pp. 257–265, 1999. ISSN: 0171-8630, 1616-1599. doi: 10.3354/meps180257.

[146] CLARKE, K. R. “Non-parametric multivariate analyses of changes in community structure”, *Australian J. Ecology*, v. 18, n. 1, pp. 117–143, 1993. ISSN: 1442-9993. doi: 10.1111/j.1442-9993.1993.tb00438.x.

[147] BÜNDCHEN, C. *Avaliação da distribuição da estatística R e nível descritivo amostral na Análise de Similaridade – ANOSIM: um estudo de caso do Projeto MAPEM.* Monografia, Universidade Federal do Rio Grande do Sul, 2010.

[148] FAGIN, R. “Combining Fuzzy Information from Multiple Systems”. In: Hull, R. (Ed.), *Proc. 15th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Systems*, pp. 216–226, Montreal, Canada, 3–5 jun. 1996. doi: 10.1145/237661.237715.

[149] ABDI, H., WILLIAMS, L. J. “Principal component analysis”, *WIREs Computational Statistics*, v. 2, n. 4, pp. 433–459, jul. 2010. doi: 10.1002/wics.101.

[150] SIEVERT, C., SHIRLEY, K. “LDAvis: A method for visualizing and interpreting topics”. In: *Proc. Workshop Interactive Language Learning, Visualization, and Interfaces*, pp. 63–70, Baltimore, MD, USA, 27 jun. 2014. doi: 10.3115/v1/W14-3110.

[151] MANN, H. B., WHITNEY, D. R. “On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other”, *Ann. Math. Statist.*, v. 18, n. 1, pp. 50–60, mar. 1947. doi: 10.1214/aoms/1177730491.

[152] MURTAGH, F., LEGENDRE, P. “Ward’s hierarchical agglomerative clustering method: which algorithms implement Ward’s criterion?” *J. Classification*, v. 31, n. 3, pp. 274–295, out. 2014.

[153] PROVOST, F., FAWCETT, T. *Data Science para Negócios: O que Você Precisa Saver Sobre Mineração de Dados e Pensamento Analítico de Dados*. 1st ed. Rio de Janeiro/RJ, Brazil, Alta Books, 2016.