

MASARYK UNIVERSITY  
FACULTY OF INFORMATICS



# **Techniques for measuring similarity of educational items**

BACHELOR'S THESIS

**Dominik Gmitterko**

Brno, Spring 2018



*This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.*



## **Declaration**

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Dominik Gmitterko

**Advisor:** doc. Mgr. Radek Pelánek, Ph.D.



## **Acknowledgements**

These are the acknowledgments for my thesis, which can span multiple paragraphs.

## **Abstract**

This work focuses on measuring of item similarity in tutoring systems using correctness of answers from users. We knew that some unexplained regularities might appear in a similarity of items. In observed tutoring system they caused separation of items into surprising clusters. This work focus on an exploitative analysis of possible causes. We found multiple high-level factors affecting similarity of items and therefore its clusters and projections. We show that structure of the system can affect item similarities. E. g., answer preferred by users causing separation of items by correct answer or user interface affecting the structure of missing data. We validated these using simulations. Findings are useful for further usage in the analyzed system or replicating similar experiments in other tutoring systems.



## **Keywords**

adaptive Learning, tutoring system, similarity, educational item, similarity measures, exploitative analysis, clustering, projections, machine learning, data analysis



# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Similarity</b>	<b>3</b>
1.1 <i>Tutoring systems</i>	3
1.2 <i>Why is similarity of items useful</i>	4
1.3 <i>Items</i>	5
1.4 <i>Measuring similarity</i>	6
1.4.1 <i>Standard pipeline</i>	7
1.4.2 <i>Similarity measures</i>	9
1.5 <i>Previous work</i>	10
<b>2 Used dataset</b>	<b>11</b>
2.1 <i>Simulated data</i>	11
2.2 <i>Umíme česky</i>	12
2.2.1 <i>Basic statistics</i>	14
2.3 <i>Visible properties of projections</i>	14
2.4 <i>Regularities</i>	16
2.4.1 <i>Level regularity</i>	16
2.4.2 <i>Answers regularity</i>	17
2.4.3 <i>Specific questions</i>	17
<b>3 Evaluation</b>	<b>19</b>
3.1 <i>Basic</i>	19
3.1.1 <i>First and last users answer</i>	19
3.1.2 <i>Level of regularity on different item-sets</i>	19
3.1.3 <i>Similarity measures</i>	21
3.2 <i>Answers regularity</i>	22
3.2.1 <i>Total similarity of items</i>	22
3.2.2 <i>Performance matrix</i>	24
3.2.3 <i>Default answer</i>	25
3.2.4 <i>Filter users</i>	25
3.3 <i>Levels regularity</i>	28
3.3.1 <i>Missing values</i>	28
3.3.2 <i>Item performance</i>	31
<b>4 Recommendations</b>	<b>35</b>

4.1	<i>Factors affecting different stages</i> . . . . .	36
4.2	<i>Practical recommendations</i> . . . . .	37
4.3	<i>General recommendations</i> . . . . .	38
<b>5</b>	<b>Conclusion</b>	<b>39</b>
<b>A</b>	<b>An appendix</b>	<b>41</b>
A.1	<i>Notebook</i> . . . . .	41
	<b>Bibliography</b>	<b>43</b>

# Introduction

Tutoring systems are computer-based systems designed to teach its users in various domains. They usually consist of a great number of task for users to solve which enables them to provide a personalized experience. To maintain a big pool of items efficiently, we need to be able to decide which items are useful and which are not.

One possible way how to achieve this is to use similarity of items. The similarity is a basic metric which can be then used in various use-cases, e.g., projections, clustering, outlier detection. This work discusses different possible similarity measures and differences between them.

The main part of this work consists of an explorative analysis of one such tutoring system. We were trying to find the different factors of tutoring system which can affect similarity of items and explain them.

We want to understand different factors mainly because some of them form unexpected regularities of items in projections and clusterings. We also want to know whether we can replicate this behavior on simulated data.

Besides Introduction and Conclusion chapters, this thesis is structured into four main chapters. The first chapter discusses the problem of measuring the similarity of educational items and advantages of using similarity. This chapter also describes previous work and different proposed measures for computing similarity. Which also requires an explanation of different types of data available in tutoring systems. The second chapter then in detail describes observed problems specific to data we are using. The third chapter gives an overview of many experiments that were concluded. And the last chapter summarizes results and points out practical recommendations.



# 1 Similarity

In this chapter, we start with describing specific problem this work is trying to solve. Then we explain what are items (questions) in tutoring systems and computing their similarity. At the beginning of the chapter, we focus on explaining basic terms like tutoring system, item, and similarity which will be used all through the work. This sections also explain why is similarity useful and lists its possible usage. The core of the chapter focuses on a description of different kinds of data available about items and possible measures for calculating similarity. Then we discuss previous work in this area.

This work focuses on tutoring system Umíme česky which has thousands of items. For management of the items, we want to display them in some way that can be naturally interpreted. Figure 1.1 shows such image. Each dot in the image represents one item, and its proximity to others represents how similar they are.

However, there some unexplained regularities that can appear. They are causing separation of items into clusters which we want to get rid of or at least understand why they are formed.

## 1.1 Tutoring systems

Tutoring systems are computer systems which purpose is to teach its users (students) some knowledge or skill. These systems do this autonomously to some degree. The degree of automation may vary for each tutoring system.

Many different aspects of learning may be implemented into the system, or they may also be left out. The most common task which tutoring systems solve is choosing a most beneficial item from the pool of items available for users to solve. The system may adapt to each of the users choosing more challenging and interesting items to users who are more skilled while presenting users who struggle with more basic problems. It is beneficial for maximizing learning aspect of the system [1]. Another aspect which can tutoring systems solve is providing hints and feedback to students. When teachers are interacting with the system, we can also provide feedback about their students progress and so on.

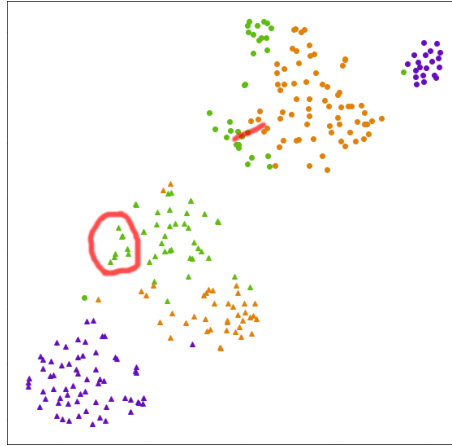


Figure 1.1: Sample projection

### 1.2 Why is similarity of items useful

As we mentioned previously, the key part of learning is solving educational items. In general, there are multiple approaches where we can use machine learning techniques with tutoring systems [2]. However, we are focusing only on different usages of similarity of items, and this section lists them.

First, most direct, usage is a recommendation of items for a student to solve. We do not want the system to recommend items which are very similar to those that user solved without any problem. However, when user struggled, the system should consider recommending more of the similar problems to strengthen users knowledge.

Another possible usage is generating hints by selecting examples which are similar to the item which is currently solved. Examples are selected from a database of examples. This usage of similarity was utilized by Hosseini; Brusilovsky[3].

Two previous use cases were using problem similarity to make some choices inside tutoring system automatically. Another approach is to bring humans into the decision-making loop [4]. This approach provides authors of tutoring system with visualizations which should inform them what changes may be useful. E.g., detecting redundant problems and pointing out where there are not enough similar problems.



One possible way of achieving this is plotting problems to plane and displaying it to the author. This approach still cannot be used for a very large amount of items. But we choose this approach for our specific data as they contain a large number of items but can be divided into item-sets that are solved independently in the system.

Another use-case for similarity is outlier detection. These are items which behave differently from others. This can be directly visible in projections as such items should lie far away from others or detected from similarity directly.

There is one more usage of similarity which we will not be discussing further. The similarity of items can be utilized for automatic construction of hierarchical categorization. Even when an author already has items categorized, he can compare it to computed categorization to verify that groups are formed correctly and refine them if needed.

Mentioned use cases of similarity are still quite abstract. We have to use some standard methods to achieve them, namely projections, clustering and outlier detection which all use similarity. E.g., we can use projections for visualizations, and clustering for construction of categorization.

### 1.3 Items

In this work, we use the term “items” which may refer to problems, questions, assignments in different systems. Item is a single entry in an educational system which users can answer. Since many aspects of this work are generally applicable, we decided to use this broad term. The complexity of single item in tutoring system can differ greatly. In some tutoring systems, this may refer to simple choice from two options while in other one item is complex tasks which user solves in a matter of minutes.

To further specify the context of our research, we will describe characteristics of items. We can compute similarity by many different measures, but they all have to use data which are available for each item. Therefore we first describe sources of data that can be utilized for measuring the similarity of items.

## 1. SIMILARITY

---

- **Item statement** is some specification of the item for a learner to solve, e.g., a natural language description of the task ( $3 + 5 = ?$ ). Another commonly used format is a grid. Many systems focusing on logic puzzles and programming problems use it.
- **Item solutions** may provide us with additional information about an item. There will usually be some sample solution (8) provided by the author, and we can also utilize all the solutions from users (8, 15, 8, 7...).
- **User's performance** consists of information provided indirectly by users. Performance of items may represent user solving times (2.8s, 4.0s, 5.1s, 1.0s...), the correctness of the answers (correct, incorrect, correct, incorrect...), the number of attempts required.

Structure of both item statement and solutions differ considerably based on the context of a tutoring system. However, in general, it is still possible to convert data into some standard form and use one of the standard measures despite original format of items.

This description of an item is broad enough to cover most of the tutoring systems. Next chapter discusses more closely tutoring system used for our experiments.

### 1.4 Measuring similarity

This section explains possible approaches to measuring the similarity of educational items. Following subsections then focuses on specific pipeline we use and comparison of used similarity measures. Similarity can also be viewed as a distance between two items, and it can be converted easily as  $1 - \text{item distance}$  or similar.

Items can be converted into vectors of numeric values. This will be useful for measuring similarity. For performance data, this is a vector containing correctness or time from all users to given item.

Other properties of items like item statement may also be represented as a vector, e.g., by using a standard technique in natural language processing – bag-of-words. It is counting occurrences of each possible word that can appear in item statement.

When we have items represented as vectors, we can compare them pairwise to get similarity using some standard similarity measures

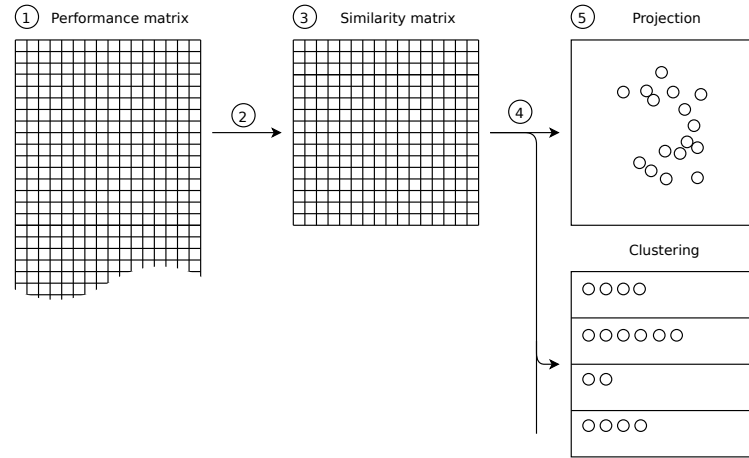


Figure 1.2: Diagram showing stages of standard pipeline

like Pearson correlation coefficient, Cosine similarity, Sokal measure or Euclidean distance.

Another possible way to measuring similarity is counting edits which would convert one item to another. This approach is called edit distance, and there are standard ways of computing it for both strings and trees. Edit distance can be then converted into similarity using something like  $1/(1 + \text{edit distance})$ . Therefore we can easily cover another common group of data structures we encounter.

### 1.4.1 Standard pipeline

To better understand how is similarity of items determined and then used we have to understand steps of its computation. We call this process a standard pipeline, and it is used in most of our experiments. When some of the further analysis modify it slightly, it is pointed out in the corresponding section. Whenever we do not specify otherwise, all projections were produced using this default pipeline. There are a few data structures and calculations involved in this process:

1. **Feature matrix** is matrix (items  $\times$  features) containing source data. As we said earlier, this could represent any property of items. This matrix consists of one vector for each item, and it may represent item statement, solution or solutions of the item and naturally even users performance.

## 1. SIMILARITY

---

The first step we take is converting raw logged information into this matrix. We used correctness of first users answer to a specific item. It will contain value 1 in case of the correct answer and 0 for incorrect. It is worth mentioning that performance matrix is relatively sparse as it is not common for users to solve all the items in the system.

In our particular case, we call this matrix a **performance matrix** because it contains information about user performance.

2. **Measuring similarity** may involve some similarity measure or edit distance. This step is used to compute the similarity between all pairs of items. In other words, we transform feature matrix into similarity matrix. For this stage, we use Pearson correlation coefficient which was selected based on previous work. Values produced by correlation are between  $-1$  and  $+1$ .
3. **Similarity matrix** is filled with a pairwise similarity of all items from the feature (performance) matrix. This gives matrix some properties. It is symmetric, and all values at main diagonal are 1.0 (because each item is identical with itself).
4. **Dimensionality reduction** is used to transform similarity matrix into a projection.

The similarity matrix is useful for a computer to make a decision based on it directly. However, this matrix still contains way too many values for a human to interpret. As some of our goals are explaining data to authors of the system we continue with a dimensional reduction to produce a more compact representation of data.

We decided to use Principal component analysis (PCA), and first two principal components are then used for 2D visualizations. We choose PCA over other commonly used technique t-SNE (which can produce better-looking results) for one important reason. Results of PCA are deterministic. It produces the same result for same input each time it is run. This is not true for t-SNE which is technique using machine learning and gradient descent for finding some local extreme. Stable results are more suitable for understanding data as there is no variation in results caused

by the algorithm. Besides, it is much easier to compare results when altering measure used for computing item similarities.

5. **Projection** is more compact representation ( $\text{item} \times 2$ ) of similarity matrix used for visualizations for end users. There are other possible use-cases of similarity like clustering, outlier detection but we are focusing on projections.

We choose this specific pipeline as it can directly utilize data about items performance. In our specific tutoring system performance holds a reasonable amount of information which can be used to calculate item similarity. Other possible choices are using item statement and solutions provided by students. Item statements in our observed exercise consist only of few Czech words with one missing spot. Student solution consists only from a selection between two provided options. Item statement and solutions can be used more effectively in other contexts like programming, mathematics, or physics where item statements are much more complex.

Also, performance is easier to transfer into different tutoring systems because performance is usually available and contrary to item statement performance can be expressed using a standard format (performance matrix).

A similar pipeline was previously used in multiple articles [5, 6]. This choice of a pipeline is also really simple. Therefore it is easy to understand and explain. As it consists only of few steps which can be studied separately and interchanged. It is possible to extend this pipeline for different types of features by adding some pre-processing and post-processing functions [6].

### 1.4.2 Similarity measures

There are many possibilities how to calculate the similarity of two vectors describing the performance of users. As we focus on using performance matrix which contains only two (boolean) values, we can summarize the performance of all users on two items by using just four values using count of encountered answer pairs [7]. See table 1.1.

Some of the measures are not using information about correct and incorrect answers symmetrically. For example, Sokal measure uses both  $a$  and  $d$  interchangeably, but similar measure Jaccard does not.

## 1. SIMILARITY

---

Table 1.1: Four parameters of used boolean similarity measures

	incorrect	correct
incorrect	$a$	$b$
correct	$c$	$d$

Pearson	$(ad - bc) / \sqrt{(a + b)(a + c)(b + d)(c + d)}$
Yule	$(ad - bc) / (ad + bc)$
Jaccard	$a / (a + b + c)$
Sokal	$(a + d) / (a + b + c + d)$
Cosine	$a / \sqrt{(a + b)(a + c)}$

Especially, some metrics ignore value  $d$ . Therefore we choose to use  $a$  as a count of incorrect answers for both items because they are rarer than correct answers.

### 1.5 Previous work

Most of the work relevant to our us comes from single article [8].

They compared few chosen similarity measures to each other. The result was that some of the similarity measures correlate greatly and some do not. It was also determined, based on simulated data, which similarity measures it is best to use for the stability of results. Especially that using Pearson correlation coefficient is a good default choice.

We are also using it for several reasons. The described article showed that it has good enough results in producing distinct clusters on performance data. Another reason is that Pearson correlation is easy to use as many computational environments already contain fast implementation.

The article also showed that using the “second level of similarity” improves the stability of results further. However, we choose not to use it as using more straightforward measure is beneficial when explaining results and this factor is important to our work.

## 2 Used dataset

The first half of this chapter focuses on used sources of data. It describes both simulated data and dataset from the actual system. We outline structure of tutoring system and focus on exercises and its objects that are useful for our work. Another section includes basic statistics about data from Umíme česky. End of the chapter provides an example of the simple projection and description what properties should useful projection have. The last section describes problems we are trying to solve and specific questions this work tries to answer.

In our analysis, we use both real data from the educational system and simulated data. Reason for this is that real-world data are useful for concluding any practical results. However, evaluation of this data is often complicated as we do not know the truth about many of its aspects. That is why we also use simulated data for validating some of our conclusions.

### 2.1 Simulated data

When simulating data, we are trying to generate results which are similar to real tutoring system. However, we use much simpler model. The similar model was previously used in multiple studies [9, 8].

The result of the simulation consists mainly of performance matrix with items as columns and users as its rows. First, we have to generate items. For each item, we choose its difficulty and skill required to solve this item. The difficulty of item is value drawn from normal distribution  $\mathcal{N}(0, 1)$ .

After that, we continued with the construction of users. We generate skills for all the users. They are random values from normal distribution  $\mathcal{N}(0, 1)$  and are stored in a  $\text{user} \times \text{skills}$  matrix.

Next, we simulate each user answering to each item. The user answers correctly with probability given by a logistic function of the difference of item difficulty  $d$  and user skill  $s$ . User skill is one of the skills which corresponds to the skill required for given item.

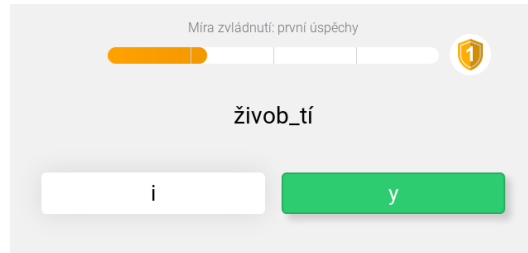


Figure 2.1: Sample question from “fill-in-the-blank” exercise in Umíme česky

$$\sigma(x) = \frac{1}{1 + e^{-1x}}$$

then

$$P(\text{correct}|s, d) = \sigma(s - d)$$

We will talk more in depth about how we generated simulated data in next chapter when describing how we used them specifically. In all experiments using simulated data, we altered this basic simulation in some way to achieve results corresponding to some attribute of real data.

## 2.2 Umíme česky

**Umíme česky** is a system for practice of Czech grammar. This section describes the tutoring system, exercise used in our analysis and lists basic statistics about datasets size.

The system contains multiple exercise types, but in our analysis, we use only one exercise type. It is simple “fill-in-the-blank” with two possible answers. A user is asked to choose one of them. This is showed in figure 2.1. After answering user receives feedback whether the answer was correct or not.

Although we focused only on “fill-in-the-blank” exercise, it can still be used to train various concepts of Czech grammar. The exercise contains many questions, and they are divided into item-sets. Each item-set consists of items practicing a different aspect of language, e.g., “Vyjmenovaná slova po B” or “Velká písmena: státy, oblasti”. This



Table 2.1: Basic statistics of dataset

	Items	Users	Answers
Whole exercise	6 037	46 128	10 421 521
“Vyjmenovaná slova po B”	273	14 207	1 216 403

item-sets are arranged into hierarchical categorization as seen in the user interface. Since our analysis works only with item-sets separately this categorization into different aspects is not important to us. After the user chooses some item-set the questions are selected randomly, so we consider the items in each item-set to be uniform and can compare them.

Items in item-sets are divided into levels. Where higher levels are intended for solving by more experienced users as they contain more difficult questions. In general, there are three difficulty levels, but not all item-sets have all three of them. Some easy item-sets include only first level. On another hand, there are item-sets which have all three levels or only higher levels.

The used dataset contains multiple sources of information about items. We have item statement which is a question with one missing spot (“živob\_tí”) and two possible answers to fill in there (“i” and “y”). There are also available all the answers from users. Another source of information is correctness of user answers and their response time.

However, it is not possible to use response time directly. We need to normalize it in some clever way as raw response time is greatly affected by both lengths of questions and users reading speed. Also, it is a good idea to use a logarithm of time instead of time itself as shown by Niznan; Pelánek; Řihák[10]. Only then it would be useful to use data about response times.

We choose this tutoring system because, as we explained before, we are focusing on problems which are most important for systems with a large number of items. Also provided dataset has a large number of users and answers which is great for the stability of results.

### 2.2.1 Basic statistics

This section provides some basic statistics about the size of dataset and users of the system. Table 2.1 shows number of items, users and answers. Statistics are both global and for selected item-set “Vyjmenovaná slova po B” which is most commonly used for our analysis. This item-set has most answers from users. Therefore, it is ideal for analysis because results are more stable. We used this item-set by default, but we also confirmed observed behavior on other item-sets.

A primary group of users using Umíme česky are children studying at primary and secondary school. Users can have individual accounts, but the system also allows teachers to create a virtual class and assign students to it. A teacher is then able to select some exercises and give them to students to solve until some deadline. Each week about 200 classes visit the system.

### 2.3 Visible properties of projections

Projections come in handy when it is hard to understand data directly because there is way too much of them. And this is our case as we focus on systems which consist of thousands of solvable items and even more users. Projections are results of dimensionality reduction techniques. We project many-dimensional data into two thoughtfully chosen dimensions to simplify them.

In general, we want projection which puts similar items together. This property can be achieved in many different ways. The following section explains choices we made when selecting a source of data, a method of processing them before applying dimensionality reduction, and dimensionality reduction technique itself.

Figure 2.2 shows how can resulting projection look like. This particular projection shows 273 items (single item-set). Each item is represented as one dot in the image, and its proximity to others represents how similar they are. There are no labels on plot axis as dimensions of projection cannot be named explicitly only the proximity of items is preserved.

As projections are meant to be used for managing item pool, this puts some constraints on how ideal projection should look like.

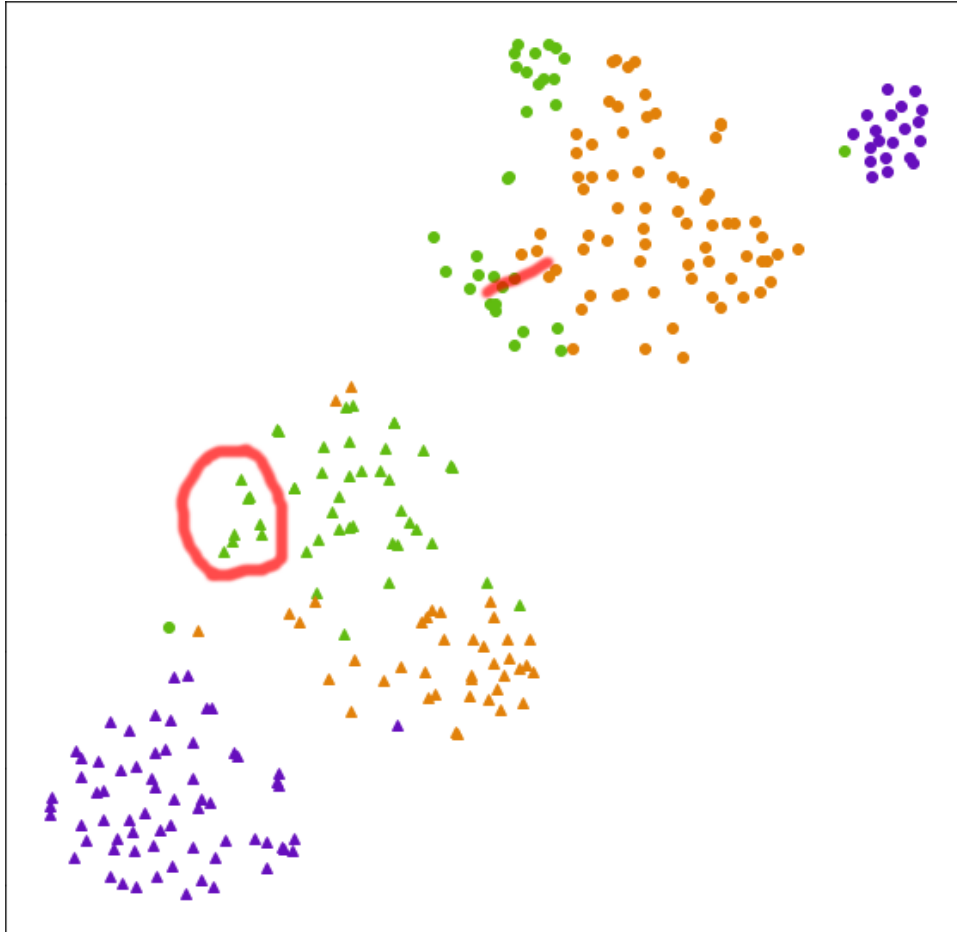


Figure 2.2: Basic projection of item-set “Vyjmenovaná slova po B”. Three used colors distinguish levels and markers correct answers of items. The highlighted group is words starting with prefix “bio” which are correctly considered similar and appear close to each other. On the other hand, two connected items should be considered similar, but they are not because they are in different levels.

We want similar items to end as close to each other as possible. In particular, when using performance as a source of data, we can interpret this as items which require same skills should be projected near each other. For example, when we look at item-set “Vyjmenovaná slova po B” (figure 2.2) there is highlighted group of words starting with common prefix “bio”. It is correct that they are close to each other as they are indeed similar because they are based on a single word. But remember, there is no information about item statement presented to the similarity measure.

### 2.4 Regularities

Projections of item-sets from system Umíme česky (figure 2.2) all display some regularities. Items depicted with same colors and items using same markers form clusters. This section talks about formed clusters in projection and why it is a problem.

#### 2.4.1 Level regularity

Figure 2.2 contains items of three different colors. Item-sets in system Umíme česky are divided into multiple levels of difficulty. For this particular item-set, there are three difficulty levels. The first level is depicted using purple (with 110 items), second orange (with 86 items), and third green (with 77 items). There is a visible pattern in our data. Each cluster consists mostly of items item from one level.

As we mentioned before, only data about user performance (correctness of answers) are used when composing projections. And there is not a direct reason for this clusters of same levels to form as no information about belonging to a particular level is presented to the algorithm.

This phenomenon is not suitable for analysis of item similarity as it can cause misleading results about similarity. One particular example is when similar words are displayed far away from one another just because they belong to different levels. Such example is pair of words “bič” and “bičík” in item-set “Vyjmenovaná slova po B”. They are connected by a line in the figure 2.2.

### 2.4.2 Answers regularity

Another property of items in figure 2.2 is depicted by a marker used. One type of marker (rectangle) shows items with correct answer “i” and another marker (triangle) shows items with correct answer “y”. Again, there is no direct information about this presented to measure of similarity, but such clusters form despite it. So it is useful to explore why is this behavior present at all.

### 2.4.3 Specific questions

There are several main questions we are trying to address:

- Which high-level factor in data can cause these regularities?
- Is there multiple factors causing these regularities?
- Can we explain them sufficiently?
- Are we able to replicate this factors on simulated data?

Following two chapters are focusing on this questions. First, we describe concluded experiments and then summarize conclusions.



## 3 Evaluation

The chapter is explaining factors that can affect resulting projections produced from real-world data. We are focusing on higher level factors that can cause a formation of clusters in projection. On a low level, it is apparent that items which have similar answers will be similar (close to each other) in projection. But we focus on higher-level factors in tutoring systems that can cause this behavior.

### 3.1 Basic

The first section is focusing on experiments which determine a range of this behavior. We ask whether clusters of same answers or levels are still present when altering parameters of pipeline used to calculate projection.

#### 3.1.1 First and last users answer

Our performance matrix contains only one value for each item-user pair. But users can answer to the item multiple times. This raises a question about which of user answers to use.

We tried using both first users answer and last in performance matrix. However, there is no visible difference in results. This is clear when we compare first and last answers of users. There is only 5% difference between them. So at least with a large number of items in the system it does not matter which one is chosen as only a few items are answered multiple times.

#### 3.1.2 Level of regularity on different item-sets

When introducing the problem of unneeded clusters in section 2.4 we showed that one particular item-set shows this behavior. However, we still do not know whether this is true for all the item-sets. We decided it may be useful to quantify how recognizable are this clusters on all item-sets.

Quantification of the regularity of clusters in the projections is executed by comparing its k-means clustering to correct level and

### 3. EVALUATION

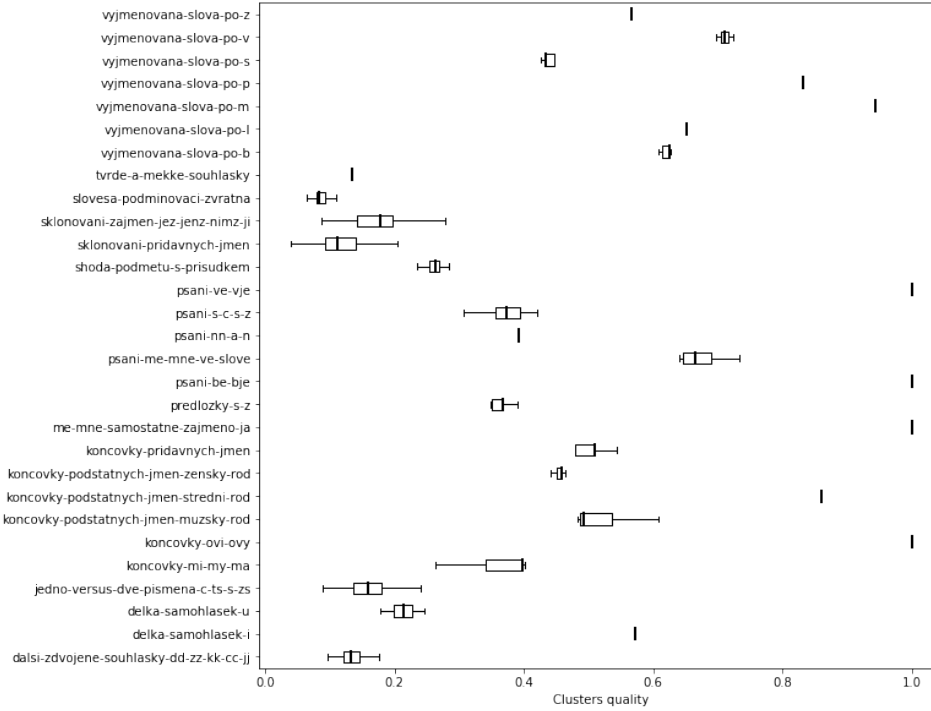


Figure 3.1: Evaluation of cluster quality on different item-sets

answer of items. We use the k-means clustering algorithm [TODO] to find *levels · correct answers* clusters in data about item similarity. Level of regularity is then determined using adjusted Rand index [11]. We compare extracted clustering with the original one (clusters created as a combination of item level and correct answer). Value 1.0 represents that k-means was able to divide all items into same clusters as original classification. On the other hand, lower values show that it is hard to divide items correctly and there are no distinct clusters. This process is repeated multiple times to account for random initialization of k-means algorithm.

Figure 3.1 describes level of regularity of different item-sets. Results show that many item-sets have distinct clusters of different levels and answers. However, there are some item-sets with an obscure structure of similarity. This mostly occurs when item-set have a large number of different answers. It is worth mentioning that even when some item-sets do not have a high score when comparing extracted clusters



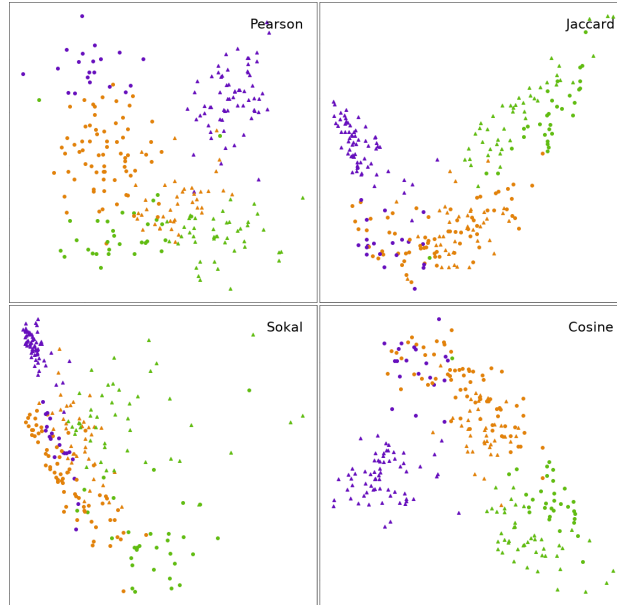


Figure 3.2: Comparison of four similarity measures projected by PCA using the same item-set

to actual but results are stable across runs of k-means. Which also suggests that projections have nicely visible clusters but items are not divided (only) by levels and answers.

We can conclude that regularities are present in some form in all item-sets.

### 3.1.3 Similarity measures

Another question is whether using Pearson correlation coefficient is not cause of some of the regularities. Therefore, we tried exchanging measure used for computing similarity of items to verify that previous results are not specific to a single measure. We evaluated all four previously chosen measures.

Figure 3.2 shows four different similarity measures used on same data. Resulting projections are different. However, they all preserve information which is causing items from the same level and same answer to form clusters. Used colors represent different levels and markers are used to distinguish different answers.

- **Pearson** is most commonly used.
- **Jaccard** differs visibly from other measures. Resulting projection still shows same clusters of levels and items are split based on a correct answer, but the similarity of the items does not correlate with correlation-based measures like Pearson. Specific difference is mentioned later in this chapter.
- **Sokal** shows best that different similarity measures can highlight different aspects of item performance. The similarity of items depends heavily on the performance of items when using Sokal. (Where performance of item is the number of correct answers divided by all answers.) Items with higher performance are much more similar than items with lower performance. This property is causing a packed cluster of items from a first level (easy) and a spread out cluster of items from third level (harder).
- **Cosine** similarity produces projection very similar to Jaccard only mirrored vertically.

We can conclude that clusters are present despite used similarity measure.

## 3.2 Answers regularity

This section describes in detail some of our experiments we concluded which relates to separation of items into clusters of same correct answers. We describe four experiments in total. They brought us to finding one more regularity, and in the end explaining regularity of splitting items by their correct answer.p

### 3.2.1 Total similarity of items

When exploring data, it may be useful to detect outliers. One possible way of how to detect an outlier is looking at the sum of its similarities or distances to other  $k$  nearest neighbors. This was done for example by Zhang; Wang[12]. In particular, this means that an item with a low sum of similarities to other items may be an outlier. At this point we encountered another regularity in data. It was detected that items

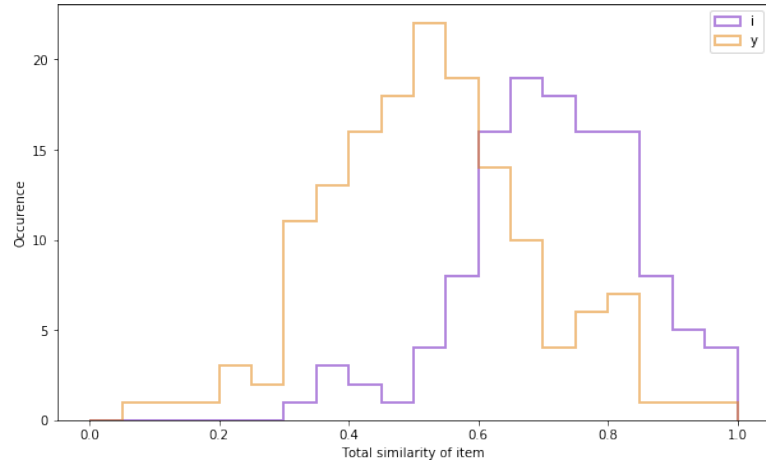


Figure 3.3: Histograms of total similarities of items divided by correct answer on one item-set

with some correct answer might have greater similarity than other answers. This could cause problems with using such outlier detection as items having some correct answers have a higher probability to be selected as an outlier, and it is not true.

Still, it is worth mentioning that this is not true for all similarity measures. It is apparent that Pearson and Cosine have this property, but measures like Jaccard does not. It has different similarity for different levels instead. This is one observed high-level difference between used similarity measures.

For this purpose, we define another property of items. The total similarity is a sum of one items vector of similarities (one column of similarity matrix) to other items. This property is normalized to range 0.0-1.0 and tells us how much is item similar to other items.

This is not specific to only one item-set (which was used in previous images), almost all item-sets display similar pattern. Although for some item-sets, it's more distinct than for others. Sets consisting of items with many possible answers does not behave this way. But that is to be expected.

We picked few item-sets which have very distinct clusters of answers and looked whether the answer with higher similarity is located on left or on the right in the user interface. Our finding was that there are item-set with a preferred answer on both left and right button. In



Figure 3.4: Part of performance matrix with highlighted row contains uneven answers from single user

particular, in concept “Vyjmenovaná slova po B” which have possible answers “i” (left) and “y” (right) the preferred one is the left answer. But for concept “Koncovky přídavných jmen” which has the same possible answers is preferred one “y” on the right.

This experiment tells us only that there is some underlying difference between items with different correct answers but does not tell us what.

#### 3.2.2 Performance matrix

In performance matrix, there are some users which performance differs greatly on different answers. This can be seen in figure 3.4. Each column in image depicts answers to one item. Each row corresponds to one user and colors represent correctness of answer (green is correct, red incorrect) or missing answer (gray). Items are ordered in a way that all items having first of the answers as correct are on the left and items with another correct answer on the right.

After exploring performance matrix some more, we can see that there are users who have much higher performance on items with one answer than another. There are even users who almost always use one of the answers. Such users are highlighted in the image. They answered almost all items with one answer (right) correctly and items with another answer (left) incorrectly.

### 3.2.3 Default answer

We guessed that there are users who prefer using one answer by default when they do not know the answer. Firstly we wanted to determine whether preferring one answer over another can affect projection and create clusters of different answers.

We used simulation for this. Simulation is extended over basic simulation described in second chapter 2.1. There are a few differences over basic simulation. For each item, we also choose a correct answer. This is used to offset logistic function higher or lower to simulate higher chance of succeeding when solving items which correct answer is the users preferred one. In this experiment, we used offset 0.2 which corresponds to 20% higher chance to answer correctly one answer and 20% lower on items with another answer. Also, there are no missing values in performance matrix as simulated users answer to all items.

This specific simulation is using two uncorrelated skills and two answers. They are distributed in a way that there is the same amount of each combination (1/4 of items). We included different skills in this simulation to better illustrate conditions of real data because such clusters of levels exist there.

When we look at resulting projection 3.5 of simulated performance data we can see that it has same properties as our real data. There are clusters (in this case caused by uncorrelated skills) which have items separated by the correct answer (represented by different colors).

We conclude that users proffering one answer over another can cause separation of items by correct answer in projection.

### 3.2.4 Filter users

The previous simulation showed that preferred answer could cause separation of items. Now we have to show that users like this exist in our real dataset.

We observed before that whole dataset uses both answers approximately the same, but for a single user, this may not be true. In this experiment, we want to quantify whether each user prefers some answer or not. For this, we used a difference of user answer performances.

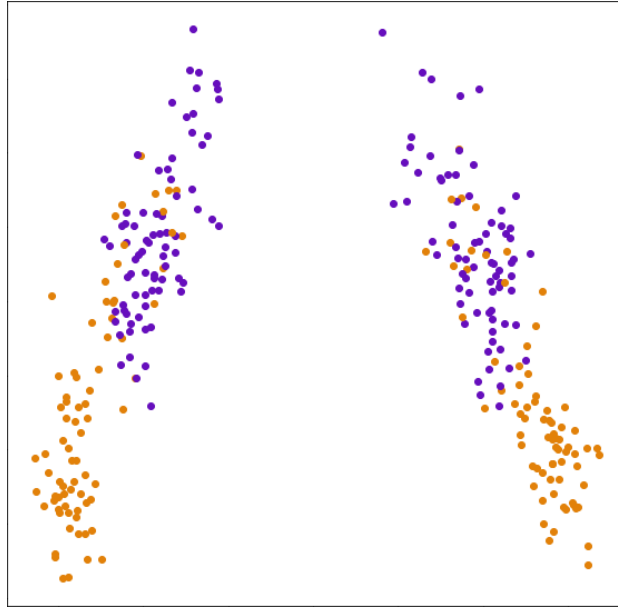


Figure 3.5: Projection of simulated data using default answers when user does not know how to answer

User answer performance is calculated as a ratio of items which user answered correctly to all items when looking only at items with a particular correct answer. We calculate this value for each possible correct answer and each user. Then we combine this set of user answer performances for each user to one value using their difference. This gives us single value representing whether the user uses some answer by default. Value ranges from 0.0 which represents no preferred answer at all to 0.5 which is largest possible preference.

Figure 3.6 shows histogram of this values calculated for each user on one item-set. There are few visible groups of users. The largest group have uniform performance on all answers (represented by values close to 0.0). On the other end of the spectrum are users who use only one answer (values close to 1.0). And in the middle, there are users who probably prefer one answer but use both answers. It is visible that there is quite a large amount of users with a small preference for one of the answers. (It does not matter which one.)

It is worth mentioning that it is normal that there is some difference between user answer performance. Even "correct" system would have

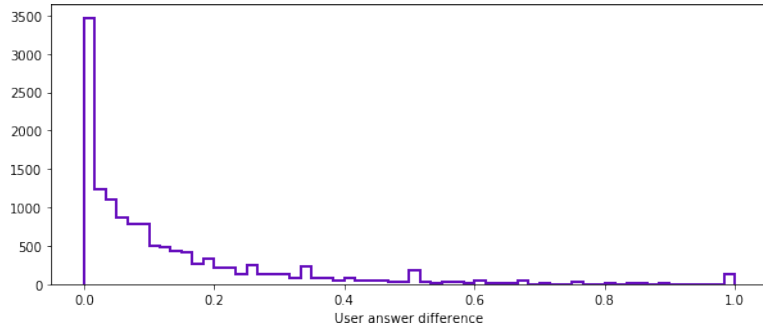


Figure 3.6: Histogram of user answer performance difference

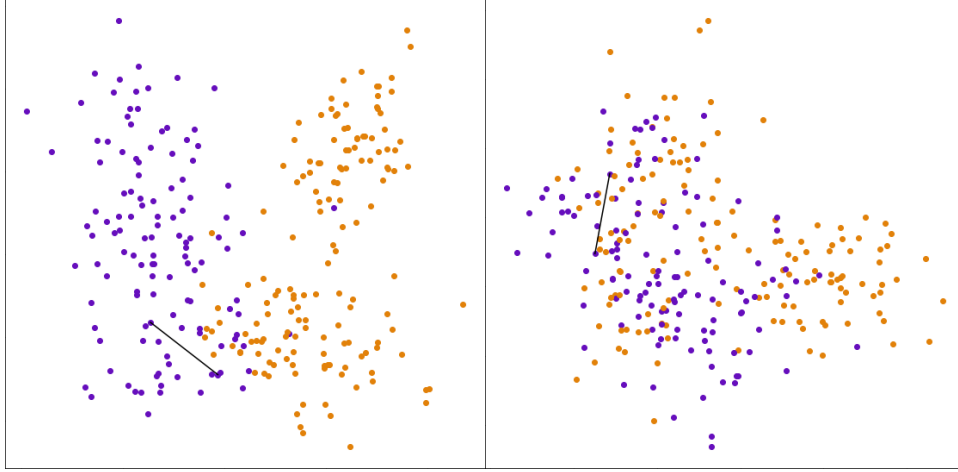


Figure 3.7: Effect of filtering out users

some non-zero differences, but they should be only close to zero. We can use this to filter out users which difference in performance is too large. We thought that this would remove the separation of items based on a correct answer from a projection. And it does. Figure 3.7 shows projection before and after filtering users based on their answer performance difference. In this particular case, we consider that user should be filtered out when his difference on answers is greater than 0.2 was chosen based on comparing our histogram with simulated data. Colors in this particular image represent correct answers to items. It is visible that separation by answer is less visible after filtering out part of the users.

In section 2.4.1 of the second chapter we also mentioned that there are words in different levels which should appear closer to each other than they do use standard pipeline without filtering of users. Such pair of words is shown in figure 3.7 with the black straight line connecting them. The distance between them shrank. But this also caused that all possible answers are now mixed. One could argue whether this is better or worse than before as questions with different answers in our particular tutoring system probably should not be as similar. But this depends on the goal and specific use-case for which is similarity used.

We choose to use a difference of performance instead of how much user uses each answer because we do not have to normalize it in any way. If we choose to use usage of answers we have to compare this to correct ratio as it is not guaranteed that item-set contains the same amount of items for all correct answers. Using difference of performance resolves this for us.

To sum it up. When we simulated users, we gave them all habit to use one answer more commonly. This divided items by their corresponding correct answer. But not all users have a habit of using one answer by default in real-world data. Yet there is quite a large amount of users who do. We can filter them out to stop them affecting projections.

## 3.3 Levels regularity

This section describes in detail some of our experiments we concluded which relates to the formation of items into clusters of the same level. The section contains two experiments sowing some factors which can affect similarity in such manner that clusters forms.

### 3.3.1 Missing values

As we mentioned before, performance matrix is relatively sparse. Missing values are not distributed randomly, and they form a pattern. Our question is whether this pattern of missing values can affect similarity of items and projections.

Items in each item-set of the system are divided into up to three levels. The difficulty of the levels differs. That is why users in the system



Table 3.1: Groups of users

1	2	3	Users
no	no	no	35%
yes	no	no	31%
no	yes	no	9%
no	no	yes	8%
yes	yes	no	7%
no	yes	yes	2%
yes	no	yes	2%
yes	yes	yes	7%

usually do not solve all of the available levels. Less experienced users tend to solve only first or first two levels. Still, more experienced users solve only higher levels. This is causing visible pattern in performance matrix. Some user rows contain information only about specific levels and are missing all values of other levels.

Based on this we can divide users into eight groups. All groups are listed in table 3.1. First three columns say whether this group contains users who solved particular levels. We say that user solved level when he answered at least 30 items (which is  $1/3$  of questions for observed item-set). Groups are not represented in data uniformly. Most of the users fall into two groups. These two groups are users who either solve only first or only second level.

Once again, we used simulated data to gather more information about this. A simulation was concluded in the same way as we explained before in section 2.1. The only difference is that resulting performance matrix will now contain missing values. We wanted to recreate missing data similarly that this occurs in real performance matrix. This is achieved by simulated users not solving all items. Each user starts with solving one level and then with some probability continues to another. So most users solve only one level, some users solve two levels, and only a few users solve all three levels. Order in which they answer levels is chosen at random as users are not required

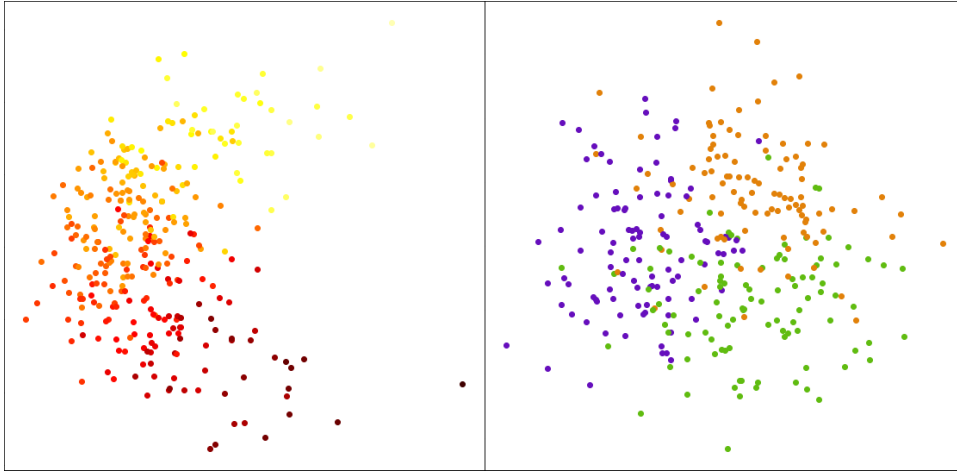


Figure 3.8: First and second (left) and second and third (right) principal components of PCA on simulated data; image on left is colored by performance of items

to continue chronologically. This is also visible in real data. There are users who solve only second or only highest difficulty level.

Figure 3.8 shows a projection of our simulated data with missing answers. Projection on the left shows first two principal components of PCA, projection on the right shows second and third component of PCA. So there are three dimensions shown effectively. Colors in the first image depict performance of items (dark colors represent small performance and lighter colors higher performance). The second image is colored by levels as usual.

In this case, the first dimension corresponds pretty well with the performance of items and next two dimensions (second image) distinguish belongingness to each level. Such easy explanation of axis is possible only for simulated data. When looking at projections of real data, it is much harder to say anything about its axis.

After looking at similarity matrix, we saw that there is a difference between similarities in the same level and similarities of items in different levels. For given amount of items, mean levels solved by a single user, and a number of users there is a visible difference in stability of similarity results. Computed similarities for items at the

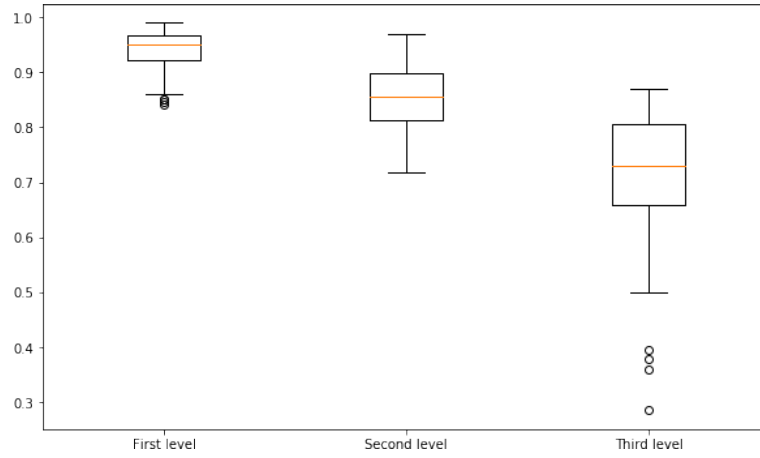


Figure 3.9: Mean performance of items in three levels of single item-set

same level are much more stable than in between levels. This may cause projection to consider items on the same level as more similar.

This is not true in general only if an amount of users who solved items in more than one level is small enough. If we increase the number of simulated users by ten times, we get stable similarity and no clusters.

### 3.3.2 Item performance

Figure 3.9 shows mean items performance (ratio of correct answers to all answers) from one item-set. Horizontal axis contains items sorted by level and mean their performance. The vertical axis shows a performance of each item. Given item-set contains three item-sets with mean performance 94%, 86%, 71% respectively.

Items are divided in such manner that difficulty of levels raises. However, this division is not strict. It is visible on Figure 3.9 that difficulty of items in levels overlap. Question is whether the difficulty of items is somehow projected into similarity of items.

For this simulation, we variate difficulty of questions. In the standard simulation, we use the difficulty of items drawn from same normal distribution  $\mathcal{N}(0, 1)$ . The only difference is that we use different means of distributions for each simulated level. For this particular experiment, we fitted this shifts to correspond to performances observed in real item-set.

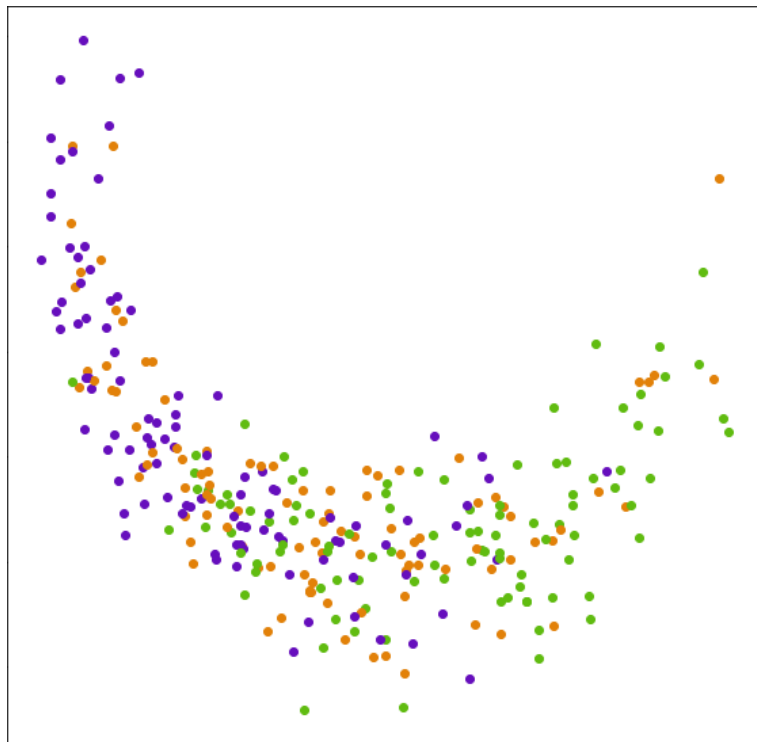


Figure 3.10: Simulated levels with different performance

Results are visible in Figure 3.10. We concluded that this factor does not cause such clusters of items from the same level as we want to recreate. We included this experiment as an example of a negative answer because we expected another result.



## 4 Recommendations

This short chapter summarizes recommendations gathered after looking at results of our experiments. We list both general recommendations which I think are useful and practical recommendations for future work on this specific dataset.

As we have learned, there are a few stages used in our pipeline for computing similarity. We come from raw logs through calculated similarity to some its use-case. Diagram 4.1 depicts the possible effect of single value changed in a log of users performance. Apparently, in performance matrix, this is only single value. When we continue with computing similarity matrix single row and single column can be affected in some way. On this stage, it is still straight as answers to item affect only similarities which include this item. Although, when we apply some additional level of processing like clustering, dimensionality reduction, or second level of similarity each changed value can affect all of the items. At this stage, we compare each item to each item.

It is apparent that single changed value does not change much. It is the underlying truth about items and users that affect similarity measures. This results also show that chosen pipeline is robust and complex as it allows us to uncover hidden information in performance data.

This work consists mostly of exploitative analysis of similarity measures and their use cases on a dataset from Umíme česky. Especially we focused on using data about user performance from single exercise. In the beginning, we knew about the existence of some regularities in data that caused clusters of items that were same in some way that should not be present in the data we used. On the way for explaining them, we found multiple high-level factors that can affect results of calculated similarity and projections of items in tutoring systems. We also encountered some problems and therefore can give some recommendations for future work in this or comparable area.

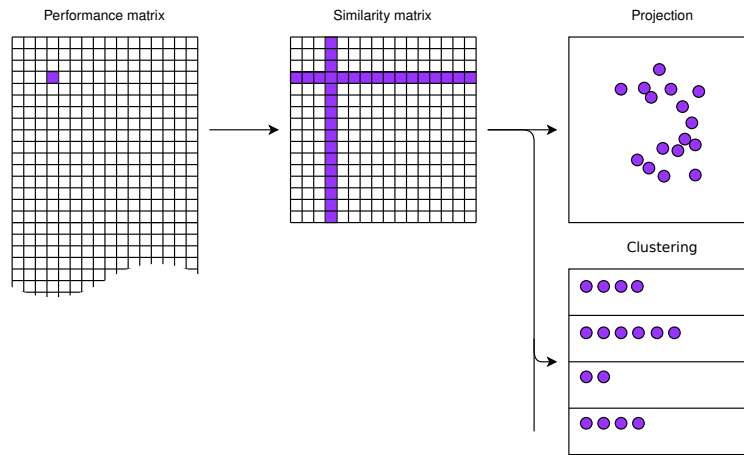


Figure 4.1: Diagram of possibly affected values on different stages of processing

### 4.1 Factors affecting different stages

In third chapter 3 we described a few high-level factors which can cause clusters of items that we found. We described only a few factors affecting tutoring systems that we tested, but there may as well be other that we did not imagine. We can already divide them into categories based on the stage of pipeline they affect. Possible choices here are similarity matrix or some specific use cases.

First described factor was the usage of some answer by default. This regularity is visible even directly in similarity matrix. Typically items with one of the correct answers have higher total similarity than items in another group. Therefore this factor falls into the first category. It is revealed in similarity matrix.

On the other hand, regularity caused by missing data is not present in similarity matrix. This factor reveals only after some usage of similarity, e.g., projection or clustering.

It is worth knowing which high-level factor can affect which stages of similarity pipeline to be prepared they can appear there. For example factor of missing answers does not affect techniques which use similarities of a single item. Only if technique compares them to each other, this factor affects the result.



## 4.2 Practical recommendations

It is important to choose appropriate pipeline for calculating similarity. One we experimented with is a good choice. Because it is utilizing data about performance and we understand it. However, there are some unneeded regularities we have to think about when using it.

For developers of the system it is worth knowing that structure of user interface can affect data in some unexpected way (refer to section 2.4). In case that similarity would be used extensively it may be useful to implement some changes to the user interface to prevent such regularities.

Another possible solution is to normalize data in some way when processing them. For example, because item-sets with a large number of correct answers are not affected by preferred answers, we can use some modified pipeline only for item-sets with a small number of answers.

We did not find any difference in the usability of observed similarity measures. Therefore we recommend using any measure that is good performing and convenient to use. Based on previous research this can be Pearson correlation coefficient applied two times sequentially. This measure has good stability of results even on smaller datasets and is easy to implement.

Most straightforward visualization of item similarity is projections. Visualizing some known property like belongingness level or place of an item in taxonomy can be used to verify whether it is good or may need some refinement. Another possible use is interactively displaying single item-set and observing particular items manually.

Projections and clustering usually suffer from same problems. E. g., Pattern of missing answers affects both of them. This problem cannot be easily solved. But as this regularity is caused by instability of items similarity, it can be improved by using more stable similarity measure.

But projections and clustering are also affected by the formation of clusters of same items with a same correct answer. When using them, we have to be aware of this.

For outlier detection, we can recommend using method that does not depend on the global distribution of values, only on the local neighborhood as items with some answers have higher similarity. Another possibility is to normalize data in some way. One example

has proposed the removal of users with a great difference of answer preference. But other less drastic normalization methods can also be found we only used this one to show that these users are a cause for this separation of items by correct answer.

We also found multiple properties that do not affect anything. Different performance of items in levels does not cause clusters, a ratio of correct answers in item-set does not affect similarity.

### 4.3 General recommendations

As we saw, when using similarity with some particular dataset problems specific to it may arise. While trying to explain this patterns in our particular data we used we run into multiple situations which can repeat even when explaining some different data using similarity techniques.

I think it is always a good idea to first determine the range of effect. E. g., whether this problem also appears when we use some different measure or source of data. I know it is basic. But it can be easily forgotten when facing some specific problem. And it was in great help to us. Such experiments were explained in first section [3.1](#) of the previous chapter.

I found it useful to use simulations when we are dealing with techniques from which we obtain some results, but it is hard to explain why (projection). Altering inputs and observing output is a way of explaining what is happening inside.

Usually, when using PCA only first two principal components are used, but I found it useful to look at more some other dimensions as well. This approach was used in section [3.3.1](#) when the first dimension correlated with the performance of items and did not give us any useful information.

## 5 Conclusion

We explored differences in using different methods for measuring the similarity of educational items based on data about the correctness of users answers. Before starting this work, we knew there were some unclear circumstances considering the formation of clusters in projection and therefore the distribution of similarities of items. All the experiments we concluded were focused on this behavior. Now we understand how the structure of system may affect performance data and similarity of items itself.

We determined whether is this behavior present in whole dataset and all similarity measures. Multiple experiments using both real and simulated data were concluded. And as a result, we were able to answer questions proposed in section 2.4.3. We found multiple high-level factors that can cause regularities which were validated using simulations. We are pretty sure about the cause of one of the regularities and described some factors that can cause other observed regularity. We think that in real data there may be multiple factors causing level regularity as it is not straightforward to explain clusters of items from the same level.

Most of the findings discovered in this dataset may not be directly transferable to another tutoring system. But used analysis can also help understand other tutoring systems. Alternatively, maybe even systems which are not used for learning but contains items with similar properties.

Before beginning work on this thesis, I also contributed to research about the similarity of programming problems [5] which can also benefit from some of the results.

This work was concluded as exploration analysis, and hence its results consist mostly of gathered information and recommendations for future work. In particular, implementing proposed recommendations when using similarity of items in mentioned tutoring system. The similarity of items is currently not used in the system Umíme česky but increasing number of items will soon call for some automated management of problem pool. For this reason, it will be useful to analyze methods for detecting problems with item pool further.

## 5. CONCLUSION

---

In particular, compare the usefulness of different methods of item visualizations for human judgment. Also, implementation of automatic detection of duplicate and outlier items may be useful. In some special cases, it may be even possible to recommend kinds of items that are missing from the system using similarity of items.

It also may be useful to look at the same data from another perspective and observe whether there are similar regularities when we calculate similarities of users instead.

However, we already fulfilled our goals for this work and answered questions we asked.

## **A An appendix**

### **A.1 Notebook**

Concluded experiments are collected in Jupyter Notebooks which also use scripts written in python to simplify our work and make notebooks cleaner. There is also some additional experiment not described in this text as we chose to include here only experiments with somehow surprising results or which gave us the most insight into problems we were solving. Providing all the experiments in this way gives everyone possibility to alter and re-execute them. More information about launching this environment is present in enclosed files.



## Bibliography

1. PAPOUŠEK, Jan; PELÁNEK, Radek. Impact of adaptive educational system behaviour on student motivation. In: *International Conference on Artificial Intelligence in Education*. 2015, pp. 348–357.
2. BAKER, RSJD et al. Data mining for education. *International encyclopedia of education*. 2010, vol. 7, no. 3, pp. 112–118.
3. HOSSEINI, Roya; BRUSILOVSKY, Peter. A study of concept-based similarity approaches for recommending program examples. *New Review of Hypermedia and Multimedia*. 2017, pp. 1–28.
4. BAKER, Ryan S. Stupid tutoring systems, intelligent humans. *International Journal of Artificial Intelligence in Education*. 2016, vol. 26, no. 2, pp. 600–614.
5. PELÁNEK, Radek; EFFENBERGER, Tomáš; VANĚK, Matěj; SASSMANN, Vojtěch; GMITERKO, Dominik. Measuring Item Similarity in Introductory Programming. In: *Proc. of Learning at Scale*. ACM, 2018. To appear.
6. KÄSER, Tanja; BUSETTO, Alberto Giovanni; SOLENTHALER, Barbara; KOHN, Juliane; ASTER, Michael von; GROSS, Markus. Cluster-based prediction of mathematical learning patterns. In: *International Conference on Artificial Intelligence in Education*. 2013, pp. 389–399.
7. CHOI, Seung-Seok; CHA, Sung-Hyuk; TAPPERT, Charles C. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*. 2010, vol. 8, no. 1, pp. 43–48.
8. PELÁNEK, Radek; ŘIHÁK, Jiří. Measuring Similarity of Educational Items Using Data on Learners' Performance. 2017.
9. PIECH, Chris; BASSEN, Jonathan; HUANG, Jonathan; GANGULI, Surya; SAHAMI, Mehran; GUIBAS, Leonidas J; SOHL-DICKSTEIN, Jascha. Deep knowledge tracing. In: *Advances in Neural Information Processing Systems*. 2015, pp. 505–513.
10. NIZNAN, Juraj; PELÁNEK, Radek; ŘIHÁK, Jiří. Using problem solving times and expert opinion to detect skills. In: *Educational Data Mining 2014*. 2014.

## BIBLIOGRAPHY

---

11. SANTOS, Jorge M; EMBRECHTS, Mark. On the use of the adjusted rand index as a metric for evaluating supervised classification. In: *International Conference on Artificial Neural Networks*. 2009, pp. 175–184.
12. ZHANG, Ji; WANG, Hai. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*. 2006, vol. 10, no. 3, pp. 333–355.