Masaryk University
Faculty of Informatics

Techniques for measuring similarity of educational items

Bachelor's Thesis

Dominik Gmiterko

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 Gmiterko

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

Acknowledgements

These are the acknowledgements 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 may appear in similarity of items. In observer tutoring system they caused separation of items into surprising clusters. This work focus on 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 system can affect item similarities. E. g., preferred answer of users causing separation by correct answer or user interface affecting structure of missing data. We validated this using simulations. Finding are useful for further usage in analyzed system or replicating similar experiments in other tutoring systems.

Keywords

tutoring system, similarity, educational item, unexpected regularities, similarity measures, exploitative analysis, clustering, projections

Contents

Introduction

Tutoring systems are computer-based systems designed to introduce users into various domains. They usually hold a large number of items which enables them to provide a personalized experience. To maintain this large 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. Similarity is basic metric which can be then used in various use-cases, e.g., projections, clustering, outlier detection. Work discusses different possible similarity measures and differences between them.

Main part of this work consists exploitative analysis of one such tutoring system. We were trying to find different aspects in tutoring system which can affect similarity of items and explain them. This work focuses on tutoring systems with thousands of items as such similarity measure is most useful for them. In particular we used data from tutoring system Umíme česky. Prior to work on this thesis I also contributed to research about similarity of programming problems [1].

Besides Introduction and Conclusion chapters, this thesis is structured into three additional chapters. First chapter talks in general about 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. Later we explain different types of data available in tutoring systems. The second chapter then advances level deeper and describes observed problems specific to data we are using. The last chapter gives an overview of many experiments that were concluded and summarizes results.

This is draft from May 3, 2018

1 Similarity

In this chapter, we talk in general about items (questions) in tutoring systems and computing their similarity. At the beginning of the chapter, we focus on better explaining basic terms like tutoring system, item, and similarity which will be used all through the work. Core of the chapter focuses on description of different kinds of data available about items and possible measures for calculating similarity. Then we report about previous work in this area. And the last section describes why is similarity useful and lists its possible usage.

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. This can be done in some simple way or system may also adapt to each of the users choosing more challenging and interesting items to users who are more skilled in a specific area. While presenting users who struggle with more basic items. This is beneficial for maximizing learning aspect of the system [2]. Another aspect which can tutoring systems solve is providing hints and feedback to students. When there are teachers interacting with the system we can also provide feedback about their students progress and so on.

One closely related area to tutoring systems are recommender systems. Their goal is to recommend some items like movies, music, news, and books that its users will like. They differ from tutoring systems only slightly. Both areas are observing users and items. But the main difference is that our main source of information is a performance of users while solving specific item while recommender systems mostly use rating of the items. Which in fact are still only numbers representing slightly different things so we can still share some of the

techniques. Although different goals bring different problems specific to each area.

One difference is that we can use some additional data about items. There are item statements and solutions of students which can in some contexts give us really useful information about items.

1.2 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 to. Since many aspects of this work are generally applicable we decided to use this broad term. 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. Computing similarity may be performed by 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.

- **Item statement** is some specification of the item for a learner to solve, e.g., a natural language description of the task. 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 provided by the author and we can also utilize all the solutions from users.
- **User's performance** consists of information provided indirectly by users. Performance of items may represent user solving times, correctness of the answers, number of attempts required.

Structure of both item statement and solutions differ greatly based context of tutoring system. However, in general, it is still possible to

convert data into some standard form and use one of a few standard measures despite original representation.

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

The most important property of tutoring systems is that they contain educational items which are solved by users. However, amount of items in systems may vary. Some tutoring systems require a large number of items. This is especially true when they practice simple facts instead of skills. In personalized tutoring systems, the need for a large pool of items is even higher. When providing personalized questions we need a wider set of items to choose from.

Dealing with a large pool of items may not be easy. To maintain a pool of items efficiently we need to be able to easily decide which items are useful and which are not. One possible tool which can help us here is similarity of items.

1.3 Measuring similarity

This section explains possible approaches to measuring similarity of educational items. Following subsections then focuses on specific pipeline we use and comparison of used similarity measures.

In general, we can compute the similarity of items in many ways. We define measures for computing similarity for two items. Then when we want to use similarity for some specific use-case we compute pairwise similarities for all pair of items.

Items in our context are commonly represented as vectors of numeric values. For performance data, this is a vector of correctness or time from all users to given item. Other properties of items like question statement may also be represented as a vector, e.g., by using bag-of-words.

When we have vectors for each item we can compare them pairwise to get similarity using some standard similarity measures 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 is called edit distance

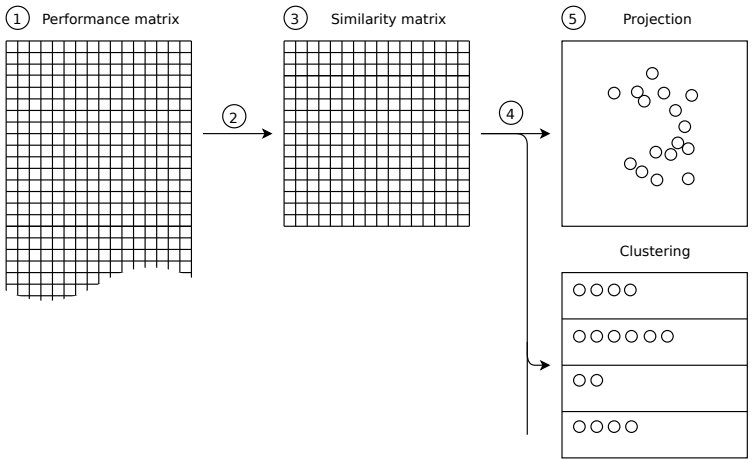

Figure 1.1: Diagram showing stages of standard pipeline

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.

1.3.1 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 cases. When some of the further analysis modify it slightly it is pointed out in the corresponding section. Whenever we don't specify so all projections were produced using this default pipeline. There are a few data structures and few calculations involved:

1. **Feature matrix** is matrix (items $\times$ features) containing source data. As we said previously this can represent any property of the 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.

   In out particular case we call this matrix a **performance matrix** because it contains information about user performance. The first step we have to take is converting raw logged information.

Columns of the matrix are items in the educational system and each row of the matrix contains data about single user's performance. In our case, we used correctness of first users answer to specific item as his performance. This means value 1.0 in case of the correct answer and 0.0 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.

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 was chosen Pearson correlation coefficient 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 as each item is same with itself.

4. **Dimensionality reduction** is used to transform similarity matrix into a projection.

   The similarity matrix is useful for a computer to directly make a decision based on it. 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. This can be achieved by using any dimensional reduction technique.

   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 to results caused

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

5. **Projection** is more compact representation (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 we think it is utilizing data about items which hold most information about their similarity. Other possible choices are using item statement and solutions provided by students. However, they do not hold as much information in our specific tutoring system. Item statements in our observed exercise consist only of few Czech words with one missing spot. And student solution is basically only selection from 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 standard format (performance matrix).

This choice of a pipeline is also relatively easy to understand. As it consists only of few steps which can be studied separately and interchanged.

1.3.2 Similarity measures

There is many possibilities how to calculate similarity of two vectors describing 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 [3]. See table 1.1.

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

Table 1.1: Four parameters of used boolean similarity measures

	incorrect	correct
incorrect	a	b
correct	c	d

- **Sokal** $(a+d)/(a+b+c+d)$
- **Cosine** $a/\sqrt{(a+b)(a+c)}$

1.4 Previous work

Most of the work relevant to our problem comes from single article [4].

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 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 "second level of similarity" improves the stability of results further. However, we choose not to use it as our dataset is large enough to be sable even with a simpler measure. Using straightforward measure is also beneficial when explaining results and this factor is important to our work.

1.5 Why is similarity of items useful

As we mentioned previously key part of learning is solving educational items. Defining some measures for computing similarity of items can then be used for different purposes.

First, most direct, usage is a recommendation of items for a student to solve. We do not want to recommend very similar items to those

that were solved without any problems. However when user struggled 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[5].

Two previous use cases were using problem similarity to automatically make some choices inside tutoring system. Another approach is to bring humans into the decision-making loop [6]. 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 still can not 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 which 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 won't be discussing further. The similarity of items can be utilized for automatic construction of clustering and hierarchical categorization. Even when author already has items categorized he can compare it to computed categorization to verify that groups are formed correctly and refine them if needed.

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 example of the simple projection and description what properties should useful projection have. 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

So when simulating data we are trying to generate results which are similar to real tutoring system. However, we use much simpler model.

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 stored in a users$\times$used skills matrix which is also filled with random values from normal distribution $\mathcal{N}(0, 1)$.

Next, we simulate each user answering to each item. User answers correctly whenever is a random value higher than a logistic function of the difference of item difficulty and user skill. When it is not given answer is incorrect. User skill is one of the skills which corresponds to the skill required for given item.

We will talk more in depth about how we generated simulated data in next chapter when describing how we used them specifically. In all

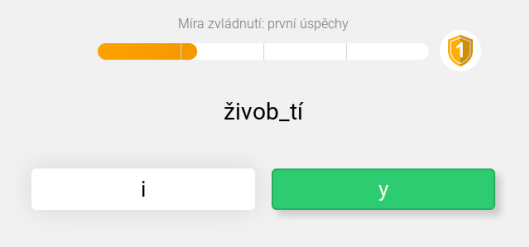

Figure 2.1: Sample question from "fill-in-the-blank" exercise in Umíme česky

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.

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 item-sets are arranged into hierarchical categorization as seen in the user interface. As our analysis works only with item-sets separately this categorization into different aspects is important to our work. We consider results from each item-set as it has uniform items, but we do not compare them against each other.

Items in item-sets are then 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

Table 2.1: Basic statistics of dataset

	Items	Users	Answers	Item-user answers
Whole exercise	6 037	46 128	10 421 521	7 264 763
One item-set	273	14 207	1 216 403	888 748

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. In "item statement" group it is a statement of the question with one missing spot and two possible answers to fill in there. We also know answers from all the users. Then in performance group of information, it is correctness of user answers and response time.

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 logarithm of time instead of time itself. This is shown by Niznan; Pelánek; Řihák[7]. Only then it would be useful to use data about response times. This is a reason why we are using mostly correctness of user answers.

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 one selected item-set which is the one most commonly used for analysis. This item-set has most answers from users, therefore, it is ideal for analysis as results are more stable. We used this item-set by default but we also confirmed observed behavior on other item-sets. The last column ("Item-user answers") contains a count of unique item-user pair which were answered. This number differs from all logged answers because users can answer the

same item multiple times. In this case, we use only one of the answers for further analysis.

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 2 thoughtfully chosen dimensions to simplify them.

In general, we want projection which puts similar items together. This 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 prior to 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 can not be named explicitly only 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.

We want similar items to end as close to each other as possible. When using performance as source of data we can interpret it as items which require same skills are projected near each other. For example, when we look at our selected item-set we can see that all words starting with “bio” are close to each other. This group consists of words which are indeed similar because they are based on a single word with added

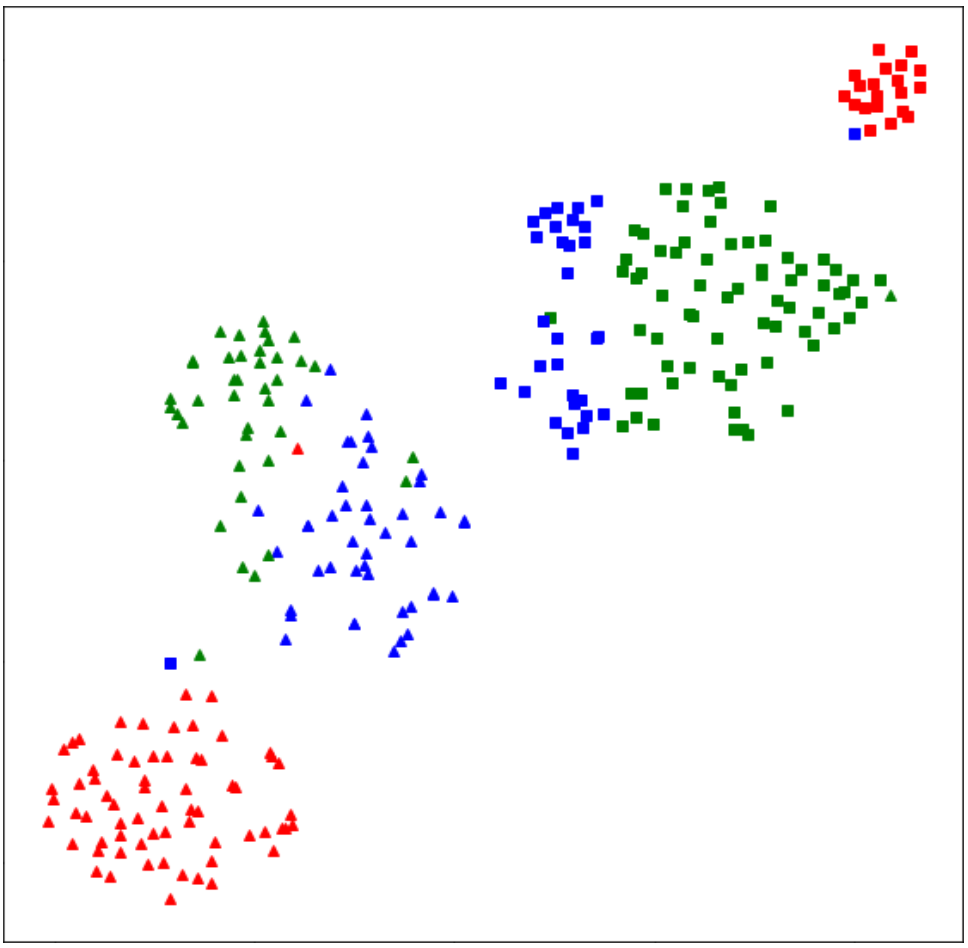

Figure 2.2: Basic t-SNE projection of item-set “Vyjmenovaná slova po B”; colors distinguish levels and markers correct answers of items

suffix. 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 (look back at Figure 2.2) all display some regularities. Items depicted with same colors and items using same markers form clusters. But so far we have not explained what properties of items colors and used markers represent. This section talks about formed clusters in projection and why it may be 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 red (with 110 items), second green (with 86 items), and third blue (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".

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.

In our particular context of Czech language items should be similar only when they have same correct answer. So this regularity should not cause any problems with projection directly but it is still useful to explore why is this behavior present at all.

2.4.3 Specific questions

There is several main question we are trying to address:

- Which high-level factor in data can cause this regularities?
- Is there multiple factors causing this regularities?
- Can we explain them sufficiently? item 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 performance correlates 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 possibly 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% different answers between them. So at least with a large number of items in the system it does not really matter which one is chosen as only a few items are answered multiple times.

3.1.2 Quality of clusters 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 quality of clusters in the projections is executed by comparing its k-means clustering to correct level and answer of

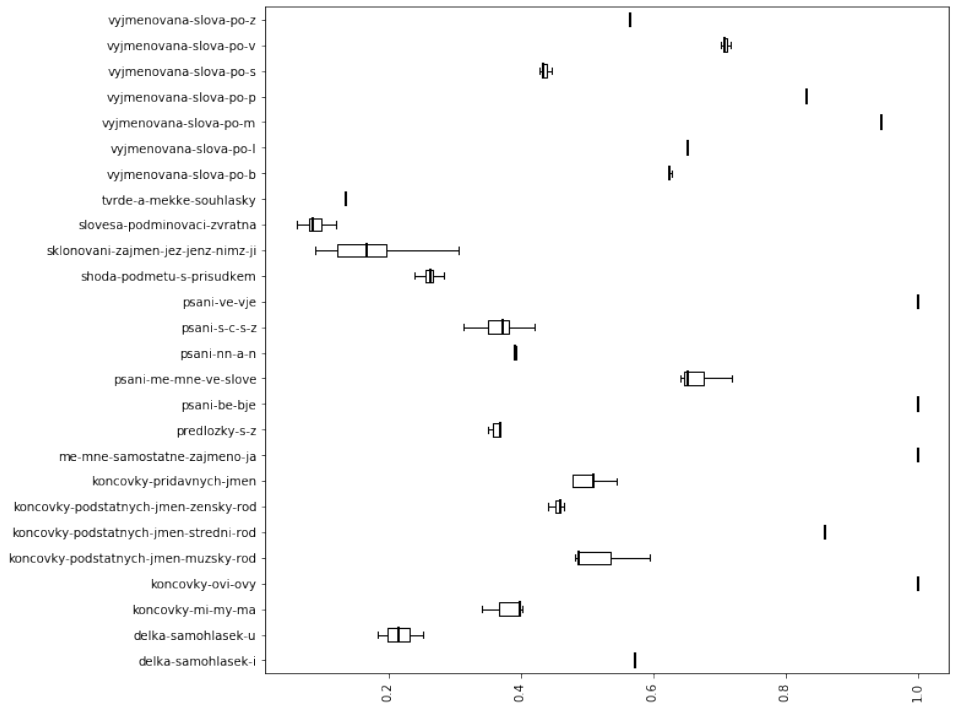

Figure 3.1: Evaluation of cluster quality on different item-sets

items. We ask k-means clustering algorithm to find *levels* $\times$ *correct answers* clusters. Quality of clustering is then evaluated using Rand index. So value 1.0 represents that k-means divide all items into same clusters as correct classification (clusters created as a combination of item level and correct answer). On other hand, lover values represent 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 depicts quality of clusters on different item-sets. Results show that many item-sets have distinct clusters of different levels and answers. But there are some item-sets with an obscure structure of projection. This mostly occurs when item-set have a large number of possible answers. It is worth mentioning that even when some item-sets do not have high score when comparing extracted clusters 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 were looking at both clusters of answers and levels simultaneously. Another possible choice would be to somehow quantify both qualities of levels and answers separately but this straightforward approach would not work there. Simplest example showing why would consists of two levels and two correct answers. Where correct answers are divided evenly into each level. When we ask k-means to create two clusters for levels and two clusters for answers it will return the same clustering in both cases because a number of clusters is the only used parameter. This will result in one metric returning high value but other one small even when clusters are really distinct.

So we can conclude that this behavior is present in all item-sets.

3.1.3 Similarity measures

In another experiment we exchanged measure used for computing similarity of items to verify that previous results aren't specific to a single measure. Especially whether they all preserve smaller similarity of items with different levels and correct answer. 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 answer to form clusters. Different colors represent different levels and markers different answers.

- **Pearson** is commonly used. It correlates well with visibly simpler measure **Yule** ($(ad - bc)/(ad + bc)$).
- **Jaccard** differs visibly from other measures. Resulting projection still shows same clusters of levels and items are split based on a correct answer but similarity of the items does not correlate 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.

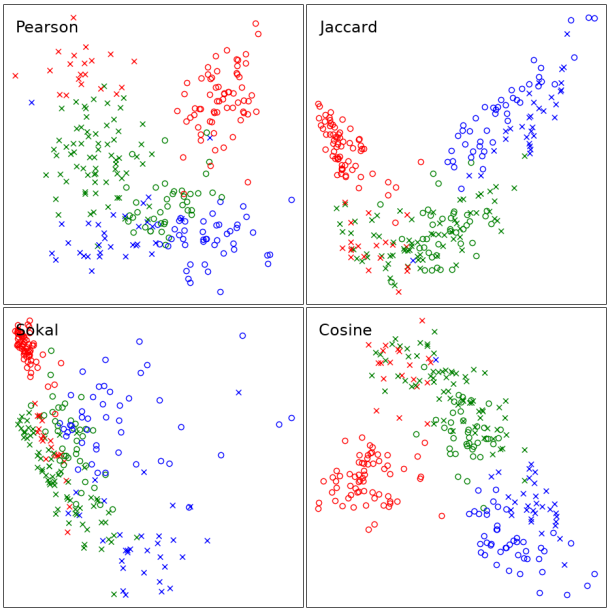

Figure 3.2: Comparison of four similarity measures on same item-set

(Where performance of item is a number of correct answers divided by all answers.) Items with higher performance are much more similar than items with lower performance. This 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.

So 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 which brought us from finding more properties of this regularity to determining cause in real dataset.

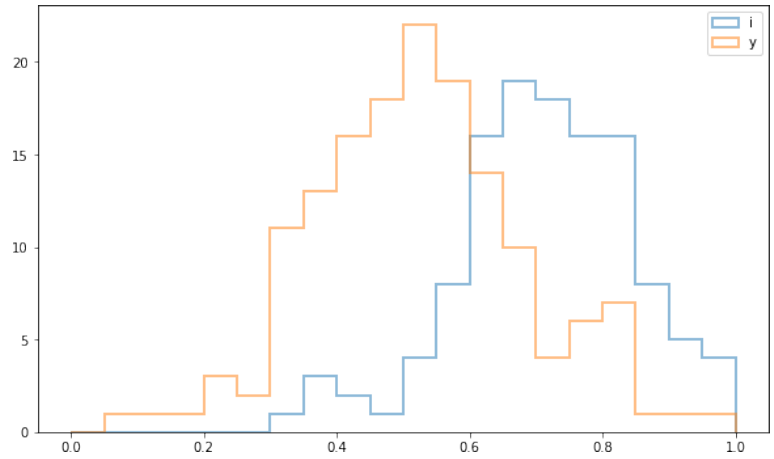

Figure 3.3: Histograms of total similarities of items divided by correct answer on one item-set

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 sum of its similarities or distances to other k nearest neighbors. This was done for example by **mitchell2014survey**[**mitchell2014survey**]. In particular, this means that an item with a low sum of similarities to other items may be an outlier. This is where we encountered another regularity in data. It was detected that items with some correct answer may have greater similarity than other answers. This could cause problems with using such outlier detection as items having some correct answers have higher probability to be selected as outlier and its apparently 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 another difference between used similarity measures.

For this purpose we define another property of items. 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

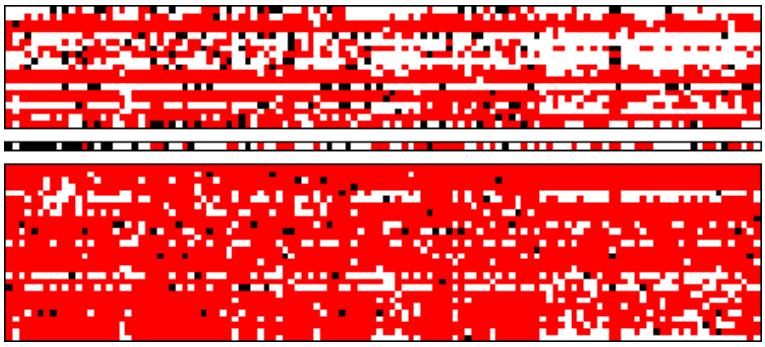

Figure 3.4: Part of performance matrix with highlighted row contains uneven answers from single user

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 right in the user interface. Our finding was that there are item-set with a preferred answer on both left and right button. In particular in concept "Vyjmenovaná slova" 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

After looking at performance matrix we can see that there are some rows which behavior 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 (white is correct, black incorrect) or missing answer (red). 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 anyways 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

Our guess was 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).

TODO better reasoning for this, explain why this happens

So users proffering one answer over another can cause items to separate by correct answer in projection.

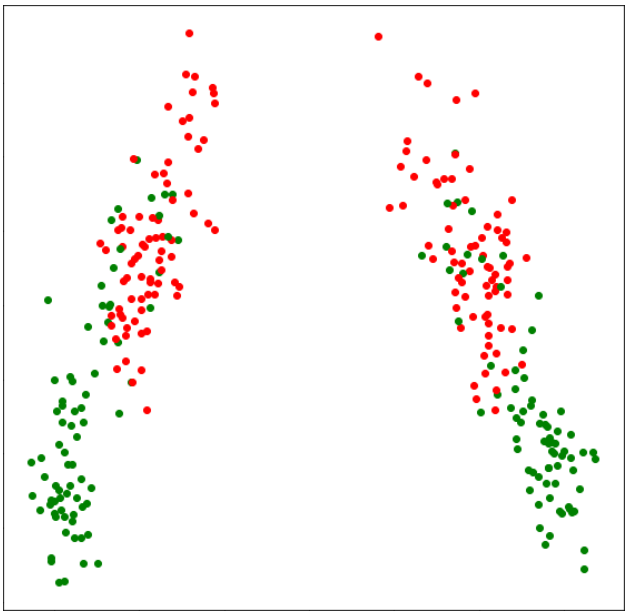

Figure 3.5: Projection of simulated data using default answers when user does not know how to answer

3.2.4 Filter users

The previous simulation showed that preferred answer can 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.

User answer performance is calculated as ratio of items which user answered correctly to all items when looking only at items with 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 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.

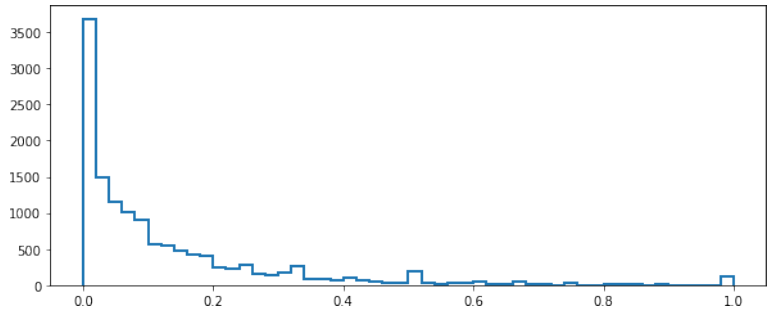

Figure 3.6: Histogram of user answer performance difference

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 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 will 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 of 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 be appear closer to each other than they do using simple pipeline. Such pair of words is shown in figure 3.7 with the black straight line connecting them. Distance between them shrank. But this also caused that that all possible answers are now mixed together. One could argue whether

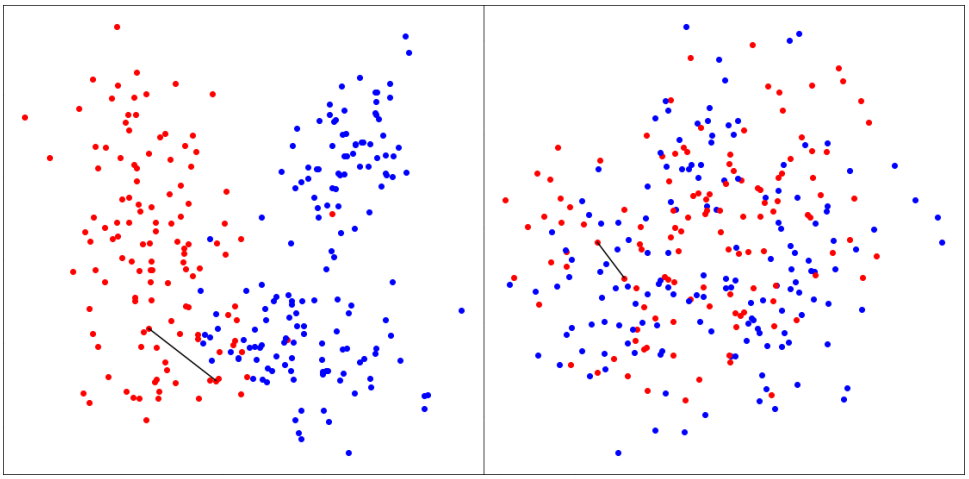

Figure 3.7: Effect of filtering out users

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 goal of use-case for which is similarity used.

We choose to use 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 same amount of items for all correct answers. Using difference of performance resolves this for us automatically.

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 formation of items into clusters of same level. The section contains two experiments sowing some factors which can affect similarity in such manner that clusters forms.

Table 3.1: Groups of users

1	2	3	Color	Users
no	no	no	black	35%
yes	no	no	red	31%
no	yes	no	green	9%
no	no	yes	blue	8%
yes	yes	no	yellow	7%
no	yes	yes	cyan	2%
yes	no	yes	magenta	2%
yes	yes	yes	light gray	7%

3.3.1 Missing values

As we mentioned before, performance matrix is relatively sparse. Yet 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. As the difficulty of levels is differs users in the system 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 8 groups. All groups are listen 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). We added color to each group so we can distinguish them in following plots easily. 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

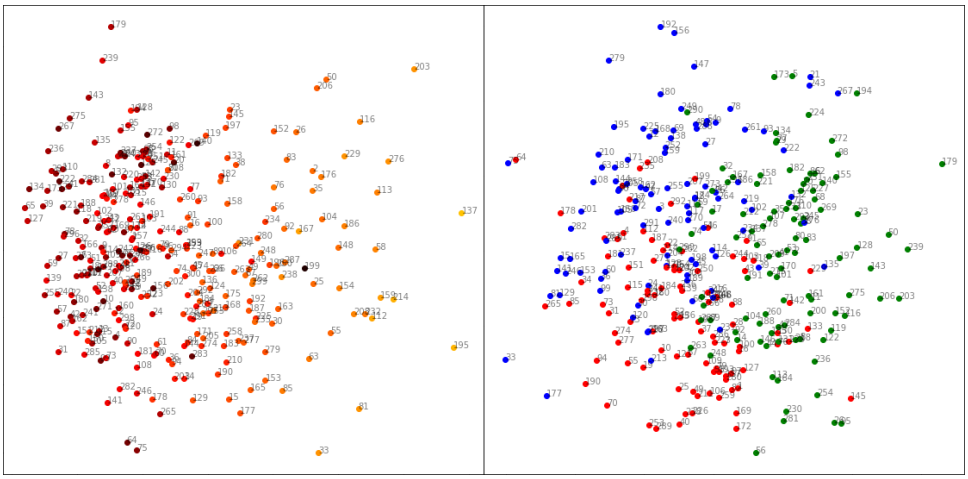

Figure 3.8: First and second (left) and second and third (right) principal components of PCA on simulated data; image on left is colored by performance of items

performance matrix will now contain missing values. We wanted to recreate missing data in a similar way 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 2 levels and only a few users solve all 3 levels. Order in which they answer levels is chosen at random as users are not required 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 projection of our simulated data with missing answers. Projection on left shows first two principal components of PCA, projection on right shows second and third component of PCA. So there are 3 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 first dimension corresponds pretty well with performance of items and next two dimensions (second image) distinguish belongingness to each level. Such easy explanation of axis is possible

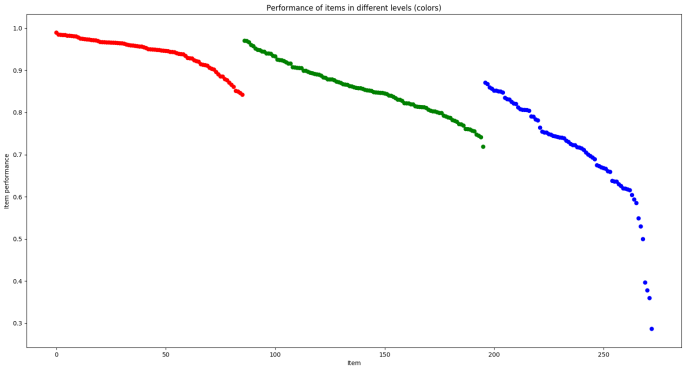

Figure 3.9: Mean performance of items in three levels of single item-set

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 difference between similarities in same level and similarities of items in different levels. For given amount of items, mean levels solved by single user and number of users there is visible difference in stability of similarity results. Computed similarities for items in same level are much more stable than in between levels. This may cause specific used use-case (in this case projection) to consider items in same level as more similar.

Yet this is not true true in general only if amount of users who solved items in more than one level is small. If we increase number of simulated users by 10 times we will 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 into 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 difficulty of items is somehow projected into similarity of items.

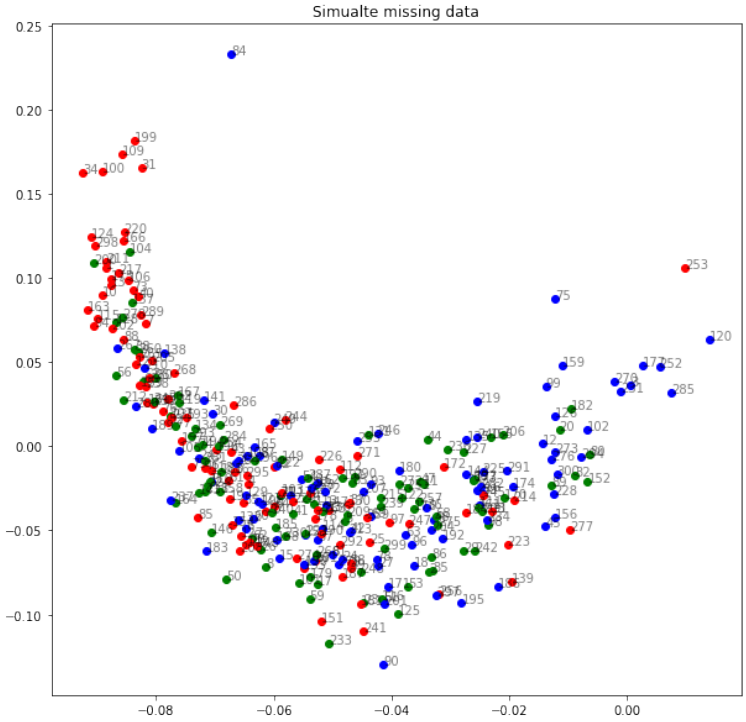

Figure 3.10: Simulated levels with different performance

For this simulation we variate difficulty of questions. In standard simulation we for all items use difficulty drawn from normal distribution $\mathcal{N}(0, 1)$. Only difference we put into this experiment is shifting mean of distribution for each simulated level. For this particular experiment we fitted this shifts to correspond to performances observed in real item-set.

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

3.4 Notebook

Concluded experiments are collected in Jupyter Notebooks which also use scripts written in python to simplify our work and make notebooks

cleaner. There are 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 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.

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 possible effect of single value changed in 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 level it is still straightforward 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 we compare each item to each item.

Yet, as we know, single changed value does not change much. It is underlying truth about items and users that affect similarity measures. This also shows 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 dataset from Umíme česky. Especialy we focused on using data about user performance from single exercise. At the beginning we knew about existence of some regularities in data that caused clusters of items that were same in some way that should not be present in the used data. On a way for explaining them we found multiple high-level factors that can possibly 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.

4.1 General recommendations

As we saw, when using similarity with some specific dataset problems specific for it may arise. While trying to explain this patterns in our

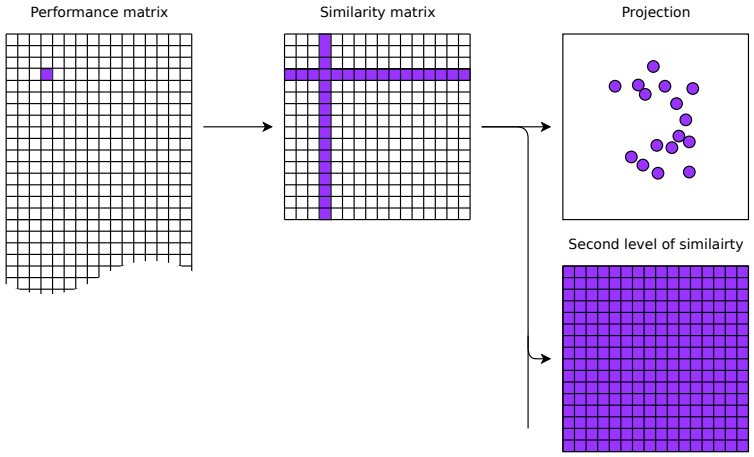

Figure 4.1: Diagram of possibly affected values on different stages of processing

particular used data we run into multiple situations which can repeat even when explaining some different data using similarity techniques.

I think it is always good idea to first determining range of effect. E. g., whether this problem also appears when we use some different measure or 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 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 for 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 first dimension correlated with performance of items and did not give us any useful information.

4.2 Practical recommendations

This work did not focus on different possible visualizations of items, but as byproduct of concluded experiments we still run into some.

projection simplest one for each item-set seaprately

For outlier detection we can recommend

using method that does not depend on global distribution of values, only on local neighborhood or use same normalization, for example proposed removal of users with great preference for one answer causing this separation of items. But other less drastic normalization method can also be found we only used this one to show that this is a cause for this clusters.

4.3 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, there may as well be other that we did not think of. Yet we can already divide them into categories based on stage of pipeline they affect. Possible choices here are similarity matrix or some specific use cases.

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

On the other hand regularity caused by missing data is not present in similarity matrix. Only after some usage of similarity is this factor revealed. For projection and second level of similarity this is visible.

5 Conclusion

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

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

This work was concluded as exploration analysis and hence its results consists mostly of gathered information and recommendations for future work. In particular implementing proposed recommendations when using similarity of items in mentioned tutoring system. 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 further analyze methods for detecting problems with item pool.

In particular compare 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.

Bibliography

1. 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.
2. 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.
3. 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.
4. PELÁNEK, Radek; ŘIHÁK, Jiří. Measuring Similarity of Educational Items Using Data on Learners' Performance. 2017.
5. 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.
6. BAKER, Ryan S. Stupid tutoring systems, intelligent humans. *International Journal of Artificial Intelligence in Education*. 2016, vol. 26, no. 2, pp. 600–614.
7. NIZNAN, Juraj; PELÁNEK, Radek; ŘIHÁK, Jiří. Using problem solving times and expert opinion to detect skills. In: *Educational Data Mining 2014*. 2014.