Measuring Similarity of Educational Items: An Overview

Radek Pelánek

Abstract—A measure of similarity of educational items has many applications in adaptive learning systems and can be useful also for teachers and content creators. We provide a thorough overview of approaches for measuring item similarity. We document the computation pipeline, explicitly highlighting many choices that have to be made in order to quantify item similarity. We also discuss methods for analysis of similarity measures. For illustration and evaluation, we consider items from diverse domains, e.g., mathematics, programming, and language learning. Although there is no ultimate, universal approach to measuring item similarity, our overview leads to guidelines that facilitate computation of item similarity in practical applications.

1 Introduction

A crucial part of learning is active solving of educational items (problems, questions, assignments). For high-quality education, we need large pools of items that learners can solve. Even teachers in a standard educational setting often work with many items. In personalized computerized systems, the need for a large pool of items is even higher—if we want to provide a customized experience for individual learners, we need to be able to choose from an extensive set of items. To use a large item pool efficiently, we need to be able to navigate it. For this, it is very useful to be able to measure the similarity of individual items. How can we measure the similarity of educational items? From a spectrum of similarity measures, how do we pick a suitable one? These are the main questions that we address in this paper.

To put the problem in the broader context, similarity of educational items is a special case of similarity of learning objects. Churchill [1] proposed a classification of learning objects into six classes: presentation objects, practice objects, simulation objects, conceptual models, information objects, and contextual representations. In this work, we deal with just one of these classes—practice objects. Similarity measures have been studied in the general context of learning objects [2]. However, when we consider a restricted type of objects, we can use more powerful techniques by utilizing data specific for a particular type. Particularly, in the case of practice objects, we have data about solutions and learners' performance that are not available for other types of learning objects.

For a specific illustration of educational items and their similarity, let us consider items in a mathematics exercise on the order of operations. A simple similarity measure for such items is Levenshtein edit distance. Fig. 1. provides an example of a projection of such items (from a real application) based on this similarity measure.

Similarity measures have many applications, particularly in adaptive learning systems. Similarity measure can be beneficial for automatic *recommendations* of activities. If a learner solves an item, but with a significant effort, it may be useful to recommend as a next item another very similar item, so that the learner can get more practice. On the other hand, if a learner solves an item easily, it is more meaningful to recommend a dissimilar item. If a learner struggles with an item, the system may provide as a *hint* a suitable worked example based on a similarity measure. Similarity measures can be used in *learner and domain modeling*: based on the similarity between items, we may define knowledge components and estimate the knowledge of learners. Similarity measures may also be used in the *user interface*, e.g., for enabling learners and teachers to navigate the item pool and manually pick an item to solve.

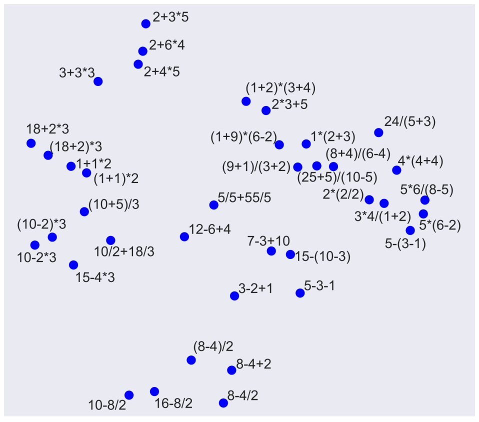

Fig. 1. An illustration of a projection of items based on similarity (the PCA projection based on the Levenshtein edit distance).

In addition to the use in automatic adaptation, similarity measures can also be very useful for empowering humans by providing useful and actionable insight (see [3] for a general discussion of this approach). For developers of learning system and content creators, a similarity measure facilitates the management of an item pool. Using similarity measures we may detect duplicate items and outliers, which should be removed from the item pool. With the use of

• *Radek Pelánek is with the Faculty of Informatics, Masaryk University Brno, Czech Republic. E-mail: pelanek@fi.muni.cz*

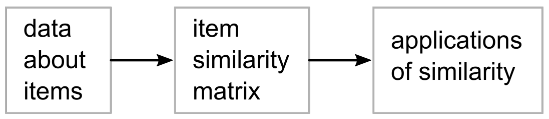

Fig. 2. Computing and applying item similarity.

visualization, it also may be possible to identify "missing items". Suitably presented data on item similarity may be very useful for teachers, instructional designers, or textbook authors. Such data may be useful for example for guiding the choice of items for an exam—typically we want items in an exam to be similar, but not very similar to items practiced during learning. We also want exam items to be rather "typical" (i.e., similar to many other items). Data on item similarity may provide impulses for the organization of classes, instructional materials, or creation of other educational resources (e.g., worked examples). In systems with crowdsourced content creation, the size of an item pool may be extensive, and similarity measures may be fundamental for efficient utilization of available resources.

Fig. 2. presents the basic approach to computing and applying item similarity. We take the available data about items and use it to compute an item similarity matrix (similarity values for each pair of items). This matrix is then used for one of the above-described applications. In these applications, the item similarity matrix is processed by other computational steps, particularly with clustering algorithms and dimensionality reduction algorithms (for creating visualizations). Experience with clustering algorithms suggests that the appropriate choice of similarity measure is more important than the choice of a specific clustering algorithm [4]. Since the choice of similarity measure is domain specific, it is typically not explored in general research on clustering. Therefore, in this work, we focus on the first step—the choice of similarity measure—and explore it in detail for the case of educational data.

Measuring item similarity is not a clearly defined problem. In most domains, there is no single correct measure of item similarity. Consider word problems in mathematics. We obtain very different item similarities depending on whether we consider similarity based on superficial features (a cover story) or deep features (a principle of solution). Similarly, for items illustrated in Fig. 1. we can focus on specific numbers, on arithmetical operations, or on mathematical principles (operator precedence, parenthesis). Should $(8-4)/2$ be considered as more similar to $8-4/2$ or to $(10+5)/3$? This question does not have a simple correct answer. In most applications, we would like to focus on deep features. In some cases, however, it may be useful to consider also shallow features that are more relevant to the perception of similarity by novice learners. For example, in solving programming exercises or mathematics word problems, it may be beneficial for learner engagement to present a series of problems with a similar theme of cover stories.

The absence of clear ground truth makes the analysis and evaluation of similarity measures difficult. A proper evaluation of similarity measures thus has to be connected to a particular application, i.e., we should evaluate the whole pipeline in Fig. 2. together. However, the computation of similarity involves many choices. Exploring all these choices for each possible use of similarity measures is not feasible. It is thus worthwhile to explore similarity measures in general (without a specific application). In this setting, we cannot give verdicts about which measure is better or worse. Nevertheless, we can explore questions like: "Which choices in the similarity computation are the most important?", "Which measures are highly correlated (and thus it is not necessary to consider both of them)?", "How much data do we need for similarity measures to be stable?". This is the basic approach that we take in this paper.

The specific contributions of this paper are the following:

- We provide a systematic overview of approaches to measuring the similarity of educational items.
- We formulate the problem of measuring item similarity in general terms, making the results applicable across a wide range of educational domains.
- We discuss methodical issues of evaluation of similarity measures, and we propose methods for analysis of similarity measures. We illustrate the use of these methods on specific examples.
- We perform an analysis of similarity measures using both real and simulated data. Based on these experiments we formulate recommendations for the use of similarity measures.

This paper extends previous work reported in [5], [6], [7].

2 Basic Approaches to Item Similarity

Fig. 2. shows the basic outline of computing and using item similarity: based on the available data we compute similarity, which can be utilized in many ways. We now provide a more detailed elaboration of the process (illustrated in Fig. 3).

2.1 Terminology

We start by clarifying the terminology. In our discussion we use "similarity measures" (higher values correspond to higher similarity); some related works provide formulas for dissimilarity measures (the distance of items, i.e., lower values correspond to higher similarity). This is just a technical issue, as we can easily transform similarity into dissimilarity by subtraction.

Another technical point is that we discuss "measures", not "metrics" in a formal mathematical sense. A distance metric must satisfy several properties: non-negativity, symmetry, the identity of indiscernibles, triangle inequality. Many of the measures that we discuss do not satisfy these properties, particularly the triangle inequality, which requires that for all x, y, z a metric m satisfies: $m(x,z) \leq m(x,y) + m(y,z)$. Note that some applications of similarity measures may require a proper distance metric (e.g., some clustering algorithms). In these cases, it is necessary to either choose a measure that satisfies the metric requirements or to apply additional processing step to obtain a distance metric (e.g., the Euclidean distance over a similarity matrix).

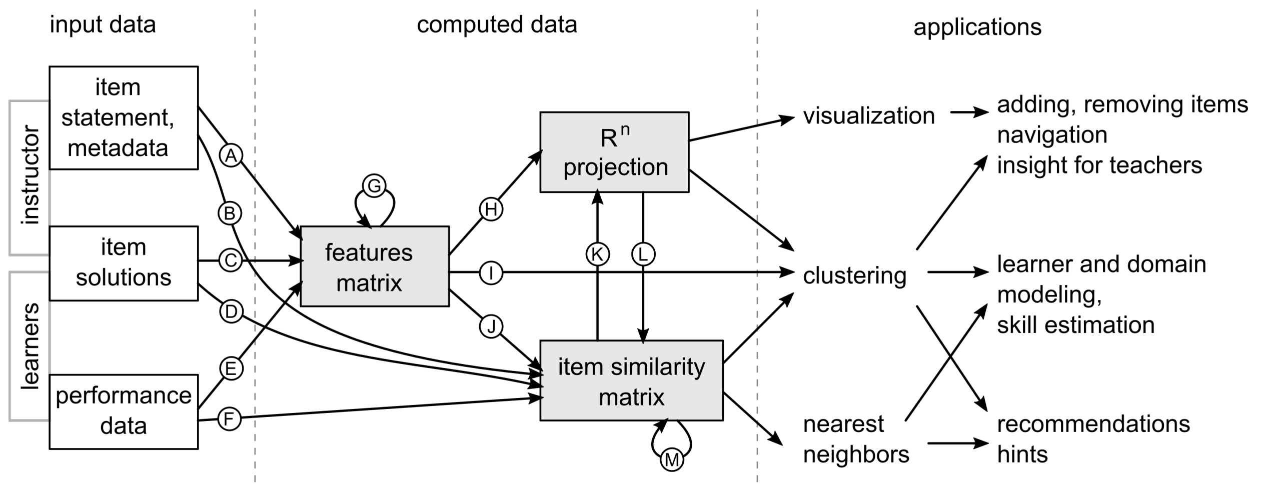

Fig. 3. The general approach to computing and applying item similarity. The arrows that are discussed in more detail in the paper are denoted by letters and referenced in the text as *Arrow X*.

2.2 Input Data

Several sources of data can be used for measuring similarity:

- an item statement and metadata: the content of the item as it is shown to learners and metadata assigned to an item (e.g., tagging with relevant topics),
- item solutions: a correct answer, a sample solution provided by the item author, an explanation, and potentially also data about learners' solutions,
- performance data: the correctness of answers, item solving times, number of attempts needed, hints taken.

Item statement and item solution data are specific to a particular domain and item type. Performance data are similar across domains and item types, but they need to be collected, which leads to a cold start problem.

To provide more specific examples of input data, we consider items from several different domains. A general type of item is a *multiple-choice question*. In this case, the item statement is typically a natural language text (a stem and a set of options), and the item solution is the correct choice (which is not useful for computing similarity). In *mathematics*, an item statement can be given as an expression in a formal syntax ("3-(5+2)") or as a word problem in a natural language text. The core of the item solution is typically a numerical value of the correct answer. It can be supplemented with an explanation (in a natural language) or a derivation of the answer (as a formal expression). In *programming*, an item statement is a specification of the item that a learner should solve, e.g., as a natural language description of the task or an input-output specification. A solution to an item is a program written in a given programming language.

2.3 Data Pipeline

In analyzing and applying similarity, it is useful to distinguish explicitly two matrices that naturally occur in computations:

- a *feature matrix*, in which rows correspond to items and columns to features of items (e.g., keywords occurring in an item statement or an item solution, tags in metadata),
- an *item similarity matrix*, which is a square matrix S, where S_{ij} denotes the similarity of items i and j.

Fig. 3. shows typical steps in the computation and application of similarity. The first step consists of processing the input data and computing either the feature matrix (*Arrows A, C, E*) or directly the item similarity matrix (*Arrows B, D, F*). These computations depend on the characteristics of particular input data. Nevertheless, there are several basic techniques for these computations that are relevant in most domains. We discuss these techniques in Section 4 and Section 5. Once we have the primary matrix, we can process it or combine it with matrices that are based on different input data. These transformations are done using mostly standard machine learning techniques. We discuss the basic approaches in Section 6.

For each step, there are many possible choices for their specific realization. For example, the *Arrow J* (computing a similarity matrix from a feature matrix) can be done using Euclidean distance, Pearson correlation coefficient, cosine similarity, and many other techniques. Analogically, there are many specific ways how to transform an item solution into a feature matrix (*Arrow C*) and many algorithms for performing clustering (*Arrow I*). Moreover, individual steps are mostly independent and can be combined.

In this work we focus on measuring similarity, i.e., constructing the item similarity matrix. We do not discuss in detail different applications since it is necessary at first to adequately clarify how to compute similarity.

3 Related Work

A wide scope of research is related to measuring the similarity of educational items. The problem is a specific instance of a general machine learning approach, and specific similarity measures have been studied in many domains.

The overall approach outlined in Fig. 3. is related to the distinction between pairwise data clustering versus feature vector clustering, which has been studied in the general machine learning research [8], [9]. The similarity of practice items is a special case of the similarity of general learning objects [1]. Previous work explored similarity measures for general objects, e.g., using object content [2] or data about object lifecycle (creation, usage, sharing) [10].

A specific domain, in which item similarities and clusters have been extensively explored, is the field of recommender systems. The basic classification of recommender systems techniques is into content-based and collaborative filtering. Here we can find a close analogy with our approach outlined in Fig. 3. Content-based techniques [11] correspond to computation based on data from the instructor (item statements, metadata, sample solutions). Collaborative filtering methods (e.g., neighborhood-based methods [12], item-item collaborative filtering [13]) correspond to similarity measures based on data from learners (performance data, learners' solutions). In recommender systems, similarity measures are widely applied, e.g., for clustering of items [14], [15] or for clustering of users [16]. A specific example of the evaluation of similarity measures is work by Spertus et al. [17], who compared six similarity measures and evaluated their impact on the studied application.

In both educational data mining and recommender systems, two basic approaches can be used to capture relations between items: a "model-based approach" and an "item similarity approach". In this work, we explore the similarity approach. The alternative model-based approach is based on the idea of constructing a simplified model that explains the observed user data. Using a matrix of learners' answers, we construct a model that predicts these answers. Typically, the model assigns several latent skills to learners and uses a mapping of items to corresponding latent factors. This kind of models can often be naturally expressed using matrix multiplication, i.e., fitting a model leads to matrix factorization. Once we fit the model to data, items that have the same value of a latent factor can be denoted as "similar". The model is typically computed using some optimization technique that leads only to local optima (e.g., gradient descent). It is thus necessary to address the role of initialization, and parameter setting of the search procedure. In recommender systems this approach is used for implementation of collaborative filtering; it is often called "singular value decomposition" [18]. In the educational context, many variants of this approach have been proposed under different names and terminology, e.g., Q-matrix [19], [20], non-negative matrix factorization techniques [21], sparse factor analysis [22], or matrix refinement [23].

Several authors discuss similarity measures specific to a particular application domain. Hosseini et al. [24] consider the similarity of programming items and worked-out examples using introductory programming problems concerning the Java language. Brusilovsky et al. [25] used a content-based similarity measure for adaptive visualization of programming examples. Sahebi and Brusilovsky [26] also analyzed similarity in the programming domain, but they focus on similarity among non-graded items (e.g., an explanatory text, videos). Liu et al. [27] proposed a similarity measure for items involving both text and images that is based on a representation computed by a neural network. The method is relevant for example in mathematics, where items containing both text and images are common. John et al. [28] studied similarity specifically for word problems in mathematics. Käser et al. [29] study similarity and clustering of users (instead of items) in a learning system for mathematics. The work is noteworthy for providing a detailed description of the whole processing pipeline for computing similarity.

Other works discuss problems in different research areas, but with close relation to the similarity of educational items. Similarity measures have been thoroughly studied for text documents [30]; this research is directly relevant to education since many educational items are textual. Specifically, recent approaches based on distributed representations of words, sentences, and paragraphs [31], [32] can be used for computing similarity of textual items. In biology and ecology, similarity measures based on binary data about species presence are often used [33]. Although this may seem like a completely different research area from educational data mining, the binary data about species presence are analogical to binary data about correct and incorrect answers and the same similarity measures can be used. In item response theory, an important assumption is local item dependence [34]; methods for checking the dependence assumption are related to item similarity.

4 Similarity Based on Item Statements, Metadata, and Solutions

We start by discussing techniques for measuring similarity using item statements, metadata, and solutions. There are two natural approaches to computing similarity based on these data. At first, we can compute similarity via a feature matrix. This matrix can be, for example, based on the bag-of-words model (i.e., the occurrence of keywords, ignoring their structure). The similarity matrix is then computed from the feature matrix using some transformation (e.g., Euclidean distance). At second, we can compute similarity by direct computation of similarity for each pair of items—this is typically done by some version of edit distance, and the approach takes the structure of items into account.

4.1 Similarity via Features

The first approach is to take item statements, metadata, or solutions and use them to compute an item feature matrix (*Arrow A* and *Arrow C*). This approach is natural particularly for metadata like tags or topics, which can be treated directly as features. For item statements and solutions, the approach is to a large degree domain specific—we need to select suitable features for a particular domain. The basic approach is to use a bag-of-words model, where the features are "keywords" occurring in items and we ignore the structure of items, using only the number of their occurrences. The choice of "keywords" depends on the type of items:

- items consisting of a natural language text (e.g., multiple choice questions, word problems): the standard bag-of-words model, keywords are just plain words (lemmatized),
- expressions in mathematics: keywords are the used operations and syntactical elements in expressions (e.g., addition, parenthesis, fraction),
- solutions in programming items: keywords are the keywords and operations of the used programming language.

Additional features depend on a specific domain and the type of a problem. For example, in programming we may use features like the use of functions, the nested depth, the length of a code, or description of variable types (e.g., integer, string, list). Items containing natural language text can utilize as features distributed vector representations [31], [32], which are able to capture not just syntax of words, but also their semantics. More complex items may contain not just text, but also images or other media formats. Liu et al. [27] propose to use neural networks in such situations to automatically find a suitable representation (features) of items.

4.2 Direct Computation of Similarity

The second approach is to compute the item similarity matrix directly from item statements (*Arrow B*) or item solutions (*Arrow D*). This can be done by some variation of edit distance [35]. The most commonly used edit distance is the Levenshtein edit distance, which is directly applicable to textual items, for example in language learning. It may be beneficial to preprocess the text with standard natural language processing techniques like lemmatization. An alternative to the edit distance approach, which considers only the syntactical aspect of items, is to consider also semantical similarity [35].

In the case of programming items, the basic Levenshtein edit distance may be still applicable, particularly for programming exercises that use graphical programming (e.g., the Blockly environment), since in this case programs can be relatively easily canonized and linearized. Edit distance can also be applied to the sequence of actions performed by a program. Piech et al. [36] use this approach together with the Needleman-Wunsch global DNA alignment for measuring edit distance. For more complex programming items and expressions in mathematics, it is natural to represent solutions as abstract syntax trees and to utilize tree edit distance [37].

5 Similarity Based on Performance Data

With performance data, we can in principle again either use the feature matrix or directly compute the similarity matrix. However, the use of the feature matrix (*Arrow E*) in this case does not seem very promising. Although there are some natural features describing learners' performance (e.g., success rate, the standard deviation of performance), these natural features are insufficient for measuring item similarity. Nevertheless, they may be beneficial as additional features in another feature-based approach.

We focus on the direct computation of similarities from performance data (*Arrow F*). In this case, the input to item similarity computation is given by data about learners' performance, i.e., a matrix of size $L \times I$, where L is the number of learners and I is the number of items. The matrix values specify the learners' performance. The matrix is typically very sparse (i.e., with many missing values). The output of the computation is an item similarity matrix, which specifies the similarity between each pair of items.

Note that in our discussion we mostly ignore the issue of learning (the change of learners' skill as they progress through items). When learning is relatively slow and items are presented in a randomized order, learning is just a reasonably small source of noise and does not have a fundamental impact on the computation of item similarities. In cases where learning is fast or items are presented in a fixed order, it is necessary to extend described techniques to take learning explicitly into account.

5.1 Correctness of Answers

We start with similarity measures that utilize only dichotomous data about the correctness of learners' answers. The advantage of these measures is that they are applicable to a wide variety of settings since the correctness of answers is the primary type of information available in learning systems.

With dichotomous data, we can summarize learners' performance on items i and j using an agreement matrix with just four values (Table 1). Although we have only four values to quantify the similarity of items i and j, previous research has identified a large number of applicable measures [38], [39], [40]. For example, Choi et al. [38] discuss 76 different measures, albeit many of them are only slight variations on one theme. Similarity measures over dichotomous data are often used in biology with data on co-occurrence of species [33]. A more directly relevant application is the use of similarity measures for recommendations [41]. Recommender systems typically use either Pearson correlation or cosine similarity for computation of item similarities [12], [13], but they consider richer than binary data.

TABLE 1
An agreement matrix for two items.

		item i	
		incorrect	correct
item j	incorrect	a	b
	correct	c	d

TABLE 2
Definitions of similarity measures based on the agreement matrix.

measure	formula	range
Pearson	$(ad - bc)/\sqrt{(a+b)(a+c)(b+d)(c+d)}$	$[-1, 1]$
Yule	$(ad - bc)/(ad + bc)$	$[-1, 1]$
Cohen	$(P_o - P_e)/(1 - P_e)$ $P_o = (a+d)/n$ $P_e = ((a+b)(a+c) + (b+d)(c+d))/n^2$ $n = a+b+c+d$	[0, 1]
Ochiai	$a/\sqrt{(a+b)(a+c)}$	[0, 1]
Sokal	$(a+d)/(a+b+c+d)$	[0, 1]
Jaccard	$a/(a+b+c)$	[0, 1]

Table 2 provides definitions of 6 measures that we have chosen for analysis. Following previous research (e.g., [33], [38]), we call measures by names of researchers who proposed them. We selected the measures in such a way as to cover measures used in the most closely related work and measures which achieved good results (even if the previous work was in other domains). We also tried to cover different types of measures.

Pearson measure is the standard Pearson correlation coefficient evaluated over dichotomous data. The Pearson correlation coefficient is defined as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

When the vectors $\vec{x}$ and $\vec{y}$ are vectors of answers, the expression simplifies to the formula given in Table 2. For example, the left part of the denominator (variance of x values) becomes $(a + c)(b + d)/n$. In the context of dichotomous data, this measure is also called the Phi coefficient or the Matthews correlation coefficient.

Yule measure achieved good results in previous work in the field of recommender systems [41]. The measure is a special case of Goodman and Kruskal's gamma, which is a rank correlation metric given as $(N_s - N_d)/(N_s + N_d)$, where N_s is the number of pairs that are ranked in the same order, and N_d is the number of pairs that are ranked in reversed order.

Cohen measure is typically used as a measure of inter-rater agreement (it is more commonly called "Cohen's kappa"). It considers the observed agreement between two ratings (P_o), taking into account the agreement occurring by chance (P_e). In our setting, it makes sense to consider this measure when we view learners' answers as "ratings" of items. Relations among Pearson, Yule, and Cohen measures are discussed in [42].

Ochiai coefficient is typically used in biology [33]. It is also equivalent to the cosine similarity measure, which for two vectors $\vec{x}, \vec{y}$ is given as the cosine of the angle between the two vectors: $\cos(\alpha) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|}$. When the cosine similarity is evaluated over dichotomous data, it leads to the Ochiai formula given in Table 2. Cosine similarity is often used in recommender systems for computing item similarity, albeit typically over interval data [13].

Finally, we consider two very simple measures. *Sokal* measure is also called Sokal-Michener or "simple matching". It is equivalent to the accuracy measure used in information retrieval. Together with *Jaccard* measure, they are often used in biology, but they have also been used for clustering of educational data [39].

Note that some similarity measures are asymmetric with respect to 0 and 1 values. These measures are typically used in contexts where the interpretation of binary values is the presence/absence of a specific feature (or observation). In the educational context, it is more natural to use measures which treat correct and incorrect answers symmetrically. Nevertheless, for completeness, we have also included some of the commonly used asymmetric measures (Ochiai and Jaccard). In these cases, we focus on incorrect answers (value a as opposed to d) as these are typically less frequent and thus bear more information.

5.2 Additional Performance Data

The correctness of answers is the primary source of information about item similarities, but not the only one. The second major type of performance data is response time. Particularly in domains like programming, the response time is a key performance indicator.

The basic approach to utilization of response time is to combine it with the correctness of an answer. Given the correctness value $c \in \{0, 1\}$ and response time $t \in \mathbb{R}^+$, we combine them into a single score r. Typically we want to take into account the labor-intensity of a particular item, e.g., by considering the median of all response times τ. Examples of specific formulas for computing the score r are:

- linear transformation for correct answers only: $r = c \cdot max(1 - t/2\tau, 0)$,
- linear transformation for both correct and incorrect answers ("high speed, high stakes scoring rule" used in the Math Garden software [43]): $r = (2c - 1) \cdot max(1 - t/2\tau, 0)$,
- exponential discounting (used in [44]): $r = c \cdot min(1, 0.9^{t/\tau - 1})$.

The scores obtained in this way are real numbers. Given the scores, it is natural to compute the similarity of two items using the Pearson correlation coefficient of scores.

It is also possible to utilize specific wrong answers for computation of item similarity. Wrong answers typically have a very skewed distribution of their occurrence dominated by a few common mistakes. These common wrong answers may be indicative of item similarity [45]. For example, if two words are repeatedly confused by learners in vocabulary learning, this can be interpreted as an evidence of the similarity of these words (we illustrate this approach in our analysis, specifically in Fig. 4.).

6 Data Transformations

Once we compute the item features or basic item similarities, we can process them using many transformations. In contrast to the above-described processing of input data, which necessarily involves details specific for a particular type of items, the data transformation steps are rather general—they can be used for arbitrary feature matrices and are covered by general machine learning techniques. We discuss techniques that are the most relevant for educational items.

6.1 Feature Transformations and Combinations

The primary feature matrix obtained by data processing contains for each feature raw counts, e.g., the number of occurrences of a keyword. Before computing similarity it is beneficial to normalize the values in the feature matrix (*Arrow G*), i.e., to perform transformations that take the feature matrix and produce a new, modified feature matrix. Examples of such transformations are:

- binarization by thresholding (a very coarse-grained normalization),
- normalization by dividing by a maximal value for each feature to get values into the $[0, 1]$ interval,
- log transform (used particularly to limit the influence of outlier values),
- TF-IDF (term frequency–inverse document frequency) transformation, used particularly for the bag-of-words features.

Often we can obtain several feature matrices (or item similarity matrices) corresponding to different data sources or multiple solutions. We can combine these matrices in different ways, e.g., by *average* (assuming additive influence of data sources), *min* (assuming conjunctive influence of data sources), or *max* (assuming disjunctive influence of data sources).

6.2 From Features to Similarity

Once we have the features matrix, we want to compute the item similarities (*Arrow J*). In this step, we have a vector of real values for each item (weights of individual features), and we compute the similarity of a pair of items as the similarity of their vectors. Computing the similarity of vectors is a common operation in machine learning, with many choices available. The common choices are cosine similarity, the Pearson correlation coefficient, and Euclidean distance (transformed into a similarity measure by subtraction). In the case of binarized features, the measures described in Table 2 are applicable.

These similarity measures are used widely in recommender systems, with the experience that the choice of a suitable measure depends on a particular data set [12]. The choice of a similarity measure also depends on the steps used to compute the feature matrix and on the purpose of the similarity measure. As an example, consider two programming problems, for which solutions use the same concepts (keywords), but one of the solutions is longer and uses each keyword multiple times. If we use normalization, the feature vectors will be (nearly) the same and the items will end up as very similar for any similarity measure. If we do not use normalization, the items will end up as very similar when we use cosine similarity and correlation coefficient, but as different when we use Euclidean distance. We cannot give a simple verdict, which one of these approaches is better, since this may depend on the intended application.

6.3 Projections

From the feature matrix or the item similarity matrix, we can compute projection to $\mathbb{R}^n$ (*Arrow H* and *Arrow K*). Such projection is typically used for applications, particularly for visualization of items. It can, however, also be a useful processing step in the computation of item similarities. For example, in the case of correlated features, we can use the principal component analysis (PCA) for decorrelating features (*Arrow H*) and then compute similarities based on the principal components (*Arrow L*).

There are many techniques for computing low dimensional projections. For the feature matrix (*Arrow H*), the popular choices include the basic linear PCA and the nonlinear t-SNE [46]. For similarity data (*Arrow K*), the basic technique is the (non-metric) multidimensional scaling.

6.4 Second Level of Item Similarity

The basic computation of item similarities computes the similarity of items i and j using only data about these two items. To improve a similarity measure, we can employ a "second level of item similarity" that is based on the computed item similarity matrix and uses information on all items (*Arrow M*). Examples of such a second step are Euclidean distance or correlation. The similarity of items i and j is given by the Euclidean distance or Pearson correlation of rows i and j in the similarity matrix. Note that the Euclidean distance may be used implicitly when we use standard implementations of some clustering algorithms (e.g., k-means) or projections (e.g., the PCA projection in Fig. 1. was done in this way).

With the basic approach to item similarity, we consider items similar when the performance of learners on these items is similar. With the second step of item similarity, we consider two items similar when they behave similarly with respect to other items. The main reason for using this second step is the reduction of noise in data by using more information. This approach may be useful particularly for dealing with learning. Two very similar items may have rather low direct similarity because getting feedback on the first item can strongly influence the performance on the second item. However, we expect both items to have similar similarities to other items.

A more technical reason for using the second step (particularly the Euclidean distance) is to obtain a measure that is a distance metric. The measures described above mostly do not satisfy triangle inequality and thus do not satisfy the requirements on a distance metric.

The basic idea of the second level similarity is related to the idea behind the SimRank algorithm [47]: "two objects are similar if they are related to similar objects". The SimRank algorithm works with relational data (represented using graphs). It is based on a recursive definition of similarity, which further generalizes the principle of second level similarity. Experience suggests that the recursive definition stabilizes very quickly. Thus for practical purposes, the second level similarity should be sufficient.

7 Analysis

In preceding sections, we provided an overview of a wide range of methods that can be used to compute the similarity of items. What are the relations among these methods? Which one should we choose for a particular application? To get insight into these questions, we now discuss methods for analyzing similarity measures and provide an illustrative analysis of some specific measures. In our analysis, we focus mainly on measures based on performance since these are to a large degree independent of a particular domain. Measures based on item statements and solutions are domain dependent, and thus it is hard to study them in general. For these, we provide several illustrative observations and examples.

To evaluate techniques on realistic and diverse educational data, we use data from several sources:

- fill-in-the-blank questions with two options about Czech grammar and orthography from the system `umimecesky.cz` (further denoted as "Czech"),
- constructed response questions from mathematics (mostly arithmetic) from the system `umimematiku.cz` (further denoted as "Mathematics"),
- Blockly programming problems from the system `robomise.cz` (further denoted as "RoboMission"),
- multiple-choice questions for English vocabulary from the system `umimeanglicky.cz`,

- Python programming problems from an introductory university course at Masaryk University.

7.1 Methods

Evaluation of similarity measures is difficult—there is no clear criterion of quality of measures since we do not have access to the ground truth (the "correct" similarity). Moreover, the desirable properties of a similarity measure may depend on a particular application. Nevertheless, it is very useful to get an insight into similarity measures in an application-independent way.

One useful type of analysis is the study of agreement of different measures. Let us consider two similarity measures S_1 and S_2. By analyzing their agreement, we cannot decide which one is better, but the analysis is still instrumental. If we find out that S_1 and S_2 are highly correlated, then we know that from a practical perspective we can pick one of them and ignore the other one. If they are highly uncorrelated, we should perform further exploration before choosing one of them for use in our application. Note that with this approach we encounter some "meta-problems": How do we quantify the agreement of two measures? Are there differences between different methods for measuring agreement? In this work, we quantify agreement using the Pearson correlation coefficient over the values in the (flattened) similarity matrix. In previous work [6], we measured agreement also using ranking (agreement on the top N most similar items); this change did not significantly influence results.

Another practical analysis is the internal stability of measures, specifically for those based on performance data. For these techniques to provide meaningful results, we need to have enough data. How much is "enough"? How do we tell whether we already have enough data? A simple check is a variation of the previous analysis: We divide the available learner data into two datasets (learner-level division), we compute the item similarity measure for each dataset and then analyze their agreement. If the agreement over two independent datasets is high, we can consider the measure to be stable.

As a complement to the analysis with real data from learning systems, it is also useful to perform analysis using simulated data. Simulated data provide a setting that is in many aspects simplified, but allows easier evaluation thanks to the access to the ground truth.

7.2 Measures Using Different Input Data

A critical aspect of the computation of item similarity is the choice of input data: item statements, item solutions, or performance data. Each of these input sources leads to a different view of similarity, and the appropriate choice depends on a particular application.

For an intuitive, high-level illustration, we consider visualizations of item similarities for groups of words in English vocabulary practice. Fig. 4. shows PCA projections of three groups of words (animals, colors, months) based on three similarity measures. The first measure uses Levenshtein distance of words, i.e., it corresponds to direct computation of similarity based on item statements (*Arrow B*). The second measure is also based on item statements (words), but uses features (*Arrow A* and *Arrow J*) which are given by a distributed representation of words based on processing a large corpus of natural language data; for the analysis, we use precomputed 300-dimensional vectors [48]. The third approach uses learner performance data (*Arrow F*). Specifically, we use data on mistakes in multiple-choice questions. The similarity measure corresponds to the ratio of mistakes for the given pair of words. Note that mistakes in vocabulary learning to a certain degree depend on the learners' first language (in our analysis we use data from Czech native speakers). Clearly, the measures based on item text and learner performance are quite different. The distributed representation of words, which captures the semantic context of words, is closer to learner mistakes than the purely syntactical approach, but there are still some non-trivial differences.

For direct analysis of item similarity matrices, let us consider the domain of programming. Fig. 5. shows two similarity matrices for 72 introductory programming problems in Python. One matrix is based on item solutions (sample programs provided by the item author); similarity is computed using a feature matrix based on programming keywords occurring in sample solutions. This matrix is dense since many keywords (e.g., `print`, `for`) are shared by many items. The second similarity matrix is based on item statements (natural language descriptions of the programming task); similarity is computed using a bag-of-words representation of the text. The item statements are brief (typically one sentence), and thus items share words only with several other items. The resulting matrix is sparse and quite different from the first one.

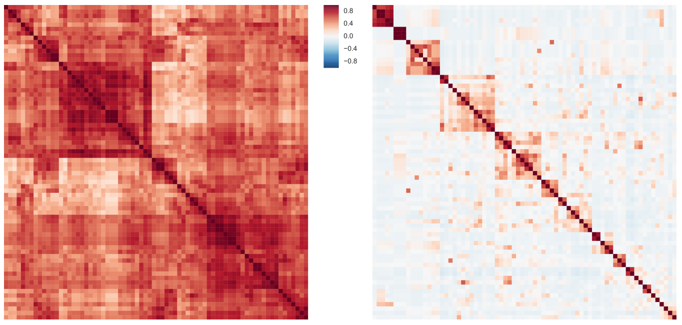

Fig. 5. Item similarity matrices for 72 Python programming problems. Left: similarity computed using features based on keywords in problem sample solutions, logarithmic transformation, and correlation. Right: similarity computed using features based on words in natural language item statement, TF-IDF transformation, and correlation. Note that items are ordered by hierarchical clustering and although the matrices show same items, each uses different ordering.

For each data source we need to pick a processing pipeline—specific choice of features and their transformation. The importance of these choices depends on a particular dataset. In general, our experience suggests that the choice of specific processing pipeline does not have a fundamental impact on the resulting similarity values, particularly when compared with the impact of the choice of input data. Some choices of a processing pipeline can lead to significantly different results—particularly when the computation becomes overly sensitive to some aspect of items, e.g., computing similarity with the use of Euclidean

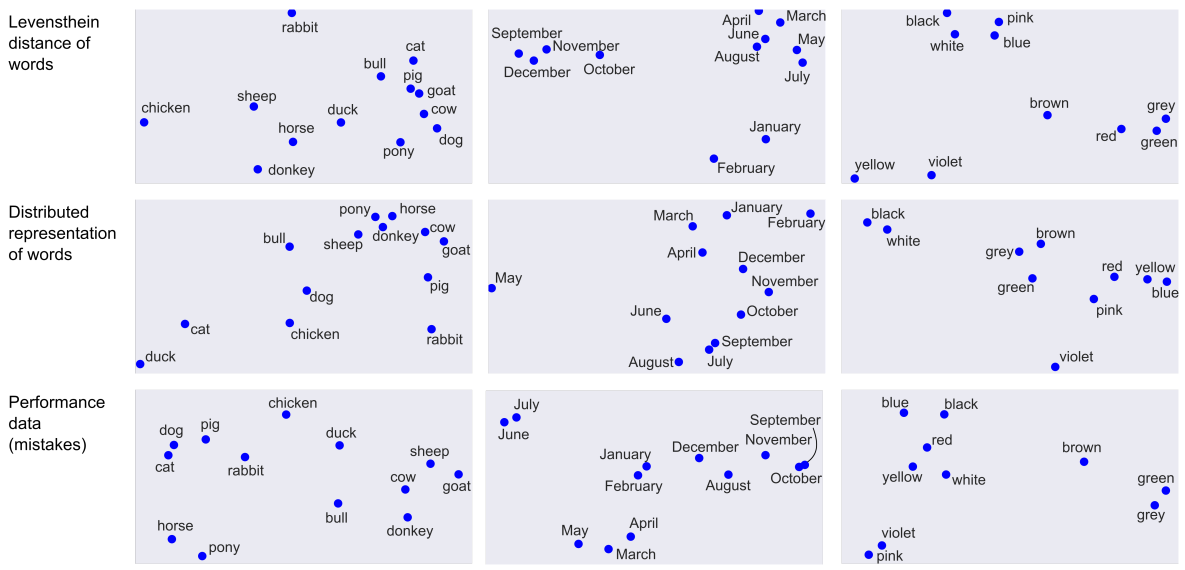

Fig. 4. PCA projections of groups of words in English vocabulary practice based on three different similarity measures.

distance over unnormalized counts of keyword occurrences.

Fig. 6. illustrates these general observations on data from the RoboMission programming problem. It shows the agreement among measures that use different input data and processing pipelines.

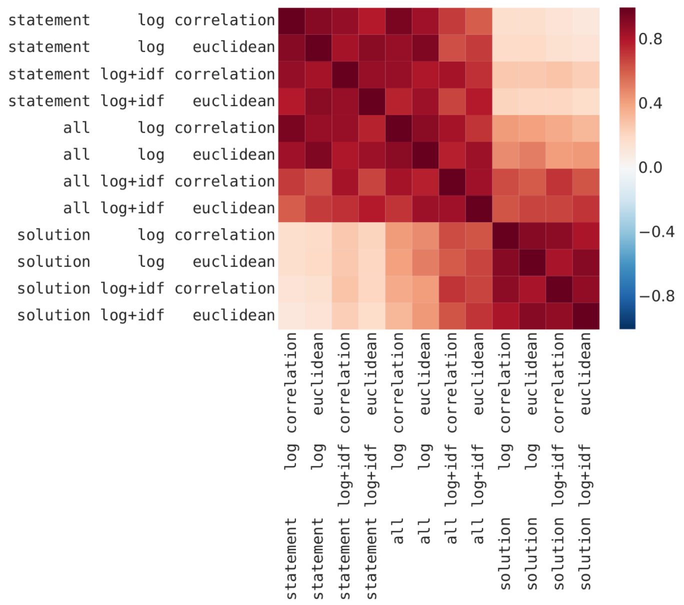

Fig. 6. Agreement among 12 similarity measures for problems in RoboMission [7]. Measures use bag-of-words features from either problem statement, sample solution, or both. Transformation is either log, or log+IDF; similarity function is either correlation, or (subtracted) Euclidean distance.

7.3 Measures Based on Performance

We analyze in more detail measures based on performance since these measures are to a large degree domain independent and can thus be studied in a more general way then content specific measures. A disadvantage of these measures is that they suffer from the cold start problem. Therefore, we pay attention to the stability of these measures depending on the amount of available data.

7.3.1 Simulated Data

We start with experiments with simulated data where we can utilize the ground truth. For generating simulated data we use a simple approach with a minimal number of assumptions and ad hoc parameters. Each item belongs to one of k knowledge components. Each knowledge component (KC) contains n items. Knowledge components correspond to the ground truth item similarities (items within one KC are "truly similar"). Each item has a difficulty generated from the standard normal distribution $d_i \sim \mathcal{N}(0,1)$. Skills of learners with respect to individual knowledge components are independent. A skill of a learner l with respect to a knowledge component j is generated from the standard normal distribution $\theta_{lj} \sim \mathcal{N}(0,1)$. We assume no learning (constant skills). Answers are generated as Bernoulli trials with the probability of a correct answer given by the logistic function of the difference between the relevant skill and the difficulty of item j (a Rasch model): $p = exp(\theta_{lj} - d_i)^{-1}$. The used approach is quite standard; closely related procedures for generating simulated data have been used by several authors [34], [39], [49].

For a good similarity measure, we expect high values for items in the same KC (within-KC values) and low values for items from different KC (between-KC values). Fig. 7. shows the distribution of similarity values for selected measures. To quantify the differences among within-KC and between-KC distributions we calculate a discrimination factor d that expresses how often a score sampled from one distribution is greater than a score sampled from another distribution. This factor is closely related to the AUC metric or the "common language effect size indicator" used in psychology [50]. The results show that for Jaccard and Sokal measures the values overlap to a large degree, whereas Pearson and Yule measures provide better results. Adding the second step—Pearson correlation in this example—to the similarity mea-

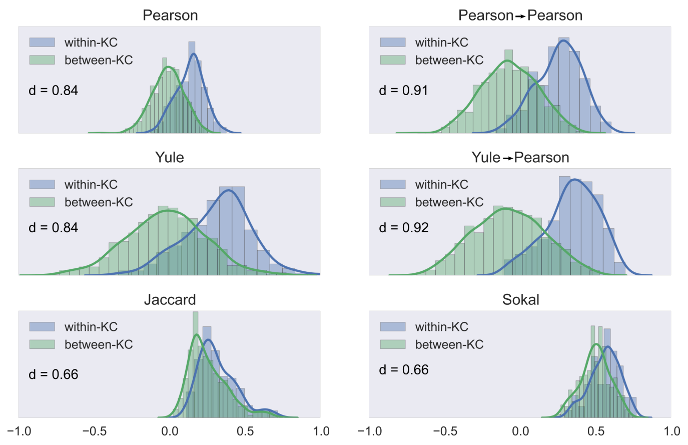

Fig. 7. Differences between similarity values inside knowledge components and between them. The experiment uses 100 simulated learners and 5 knowledge components with 10 items in each of them.

sure separates within-cluster and between-cluster values better.

Fig. 8. shows the quality of different measures depending on the number of learners. Sokal and Jaccard are poor measures, and their performance does not improve with additional data. Pearson and Yule have very comparable performance. The second step in similarity improves the quality of a measure. The figure also shows that with 200 learners the best measures achieve quite good results. This result should, however, be interpreted as an "optimistic bound", since our setting is simplified, e. g., the individual knowledge components are entirely independent and data are complete (no missing values). For practical applications, we should expect the necessary number of learners to be higher.

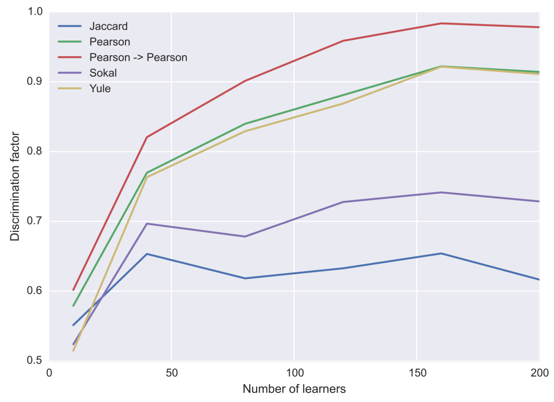

Fig. 8. The impact of the used data size on the quality of different measures (quantified by the discrimination factor). The experiment uses 5 knowledge components with 20 items in each of them.

For a better insight into the role of the second step of similarity, Fig. 9. shows the impact of the second and the third step of similarity for our simulated data. We see that further steps amplify the signal in data (the true within-KC similarities). However, we also see that the further steps also amplify noise in data (between-KC similarities, which are not present in the ground truth model generating the data). This amplification of noise is not a problem when we are concerned with detecting the most similar items. However, it has an impact of the internal stability of the measure when the stability is quantified by correlation of similarity values; we observe this behavior also over real data.

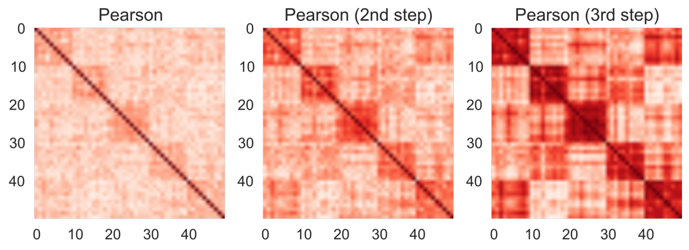

Fig. 9. Illustration of the impact of the second and the third step of similarity. The experiment uses 200 learners and 5 knowledge components with 10 items in each of them.

7.3.2 *Real Data*

For data coming from real systems, where we do not have access to the ground truth, we focus on the analysis of agreement between measures. We have analyzed the agreement among the six measures described in Table 2. Fig. 10. shows averaged results for five frequently solved knowledge components from the Czech and Mathematics datasets. As can be seen, the results are very similar for these two datasets, even though they correspond to very different items. The results are also quite stable across knowledge components. The results show that Pearson and Cohen measures are very highly correlated across all data sets and have nearly the same values (although not exactly the same). Larger differences in similarity values (but only up to 0.1) can be found when one of the values in the agreement matrix is small, which happens only for poorly correlated items with the resulting similarity value around 0. The second pair of highly correlated measures is Ochiai and Jaccard, which are both asymmetric with respect to the agreement matrix. The correlation between these two pairs of measures varies depending on a dataset and in some cases drops to 0.5. These two measures are also highly correlated with Pearson and Cohen. Yule measure is somewhat different from other measures. Sokal is the most outlying measure with small correlation with all other measures.

In Fig. 8. we analyzed the impact on the amount of data on the quality of measures using simulated data. In Fig. 11. we perform an analogical analysis for real data,

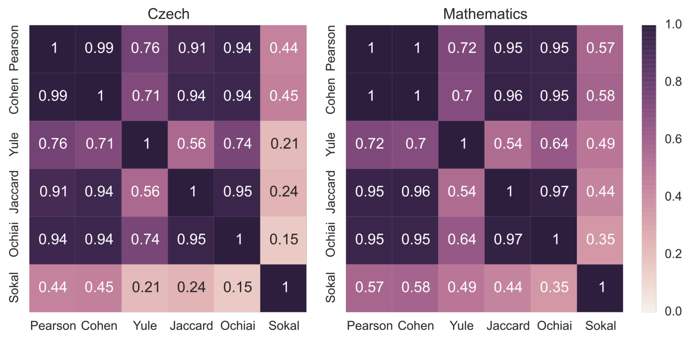

Fig. 10. The agreement of similarity measures. Reported values were computed as average correlations computed over 5 knowledge components from each domain.

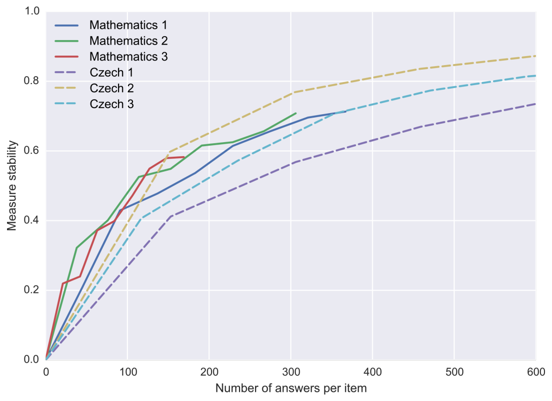

Fig. 11. The internal stability of measures depending on the amount of data.

reporting the internal validity of a measure using the two halves experiment. In this case, we use only the Pearson measure and show results for six knowledge components of different types. The results are quite stable across knowledge components—they show that to get good stability of a measure we need over 500 answers per item.

A detailed analysis of similarity values over real data shows that the values can be influenced by the way data are collected. For example, attrition bias (data not missing at random) or details of user interface (leading to bias in answers for some users) can lead to artifacts in computed similarity values [51]. These issues need more attention in future research.

8 Discussion

We conclude our overview of similarity measures for educational items with a high-level discussion of the most important issues in the use of similarity measures. We also provide recommendations for practitioners.

8.1 Choices and Their Evaluation

Measuring the similarity of items is a complex problem because it can be tackled in many ways and it is hard to evaluate the quality of measures. We propose a systematic approach to studying similarity measures (outlined in Fig. 3.), which explicitly highlights many choices that we need to make to specify a similarity measure. In practical applications of similarity measures, it can easily happen that we make some of these choices implicitly, e.g., by using a default setting in a used software package in *Arrow J*. Such implicit choices can influence results of experiments based on similarity measures. It is thus important to be at least aware of them.

Ideally, we would like to compare different choices and use the best similarity measure. Unfortunately, the evaluation of similarity measures is difficult. Without going into the details of a specific application, we cannot objectively compare measures. However, it is not reasonable to do the evaluation only with respect to the final application. Consider, for example, the use of similarity measures for recommending items in a learning system. Evaluating the quality of recommendations is a complex task even if we compare just a single version of the recommendation algorithm to a control group, particularly for a learning system which aims to balance several goals like learning and engagement [52]. It is not feasible to evaluate variants of the recommendation algorithm for many similarity measures.

We thus need to analyze similarity measures even without considering specifics of a particular application. Such evaluation cannot give us verdicts about which measures are good or bad. However, we can evaluate which decisions in the similarity computations are critical—which computation pipelines lead to different results. In this way, we can narrow the number of measures that need to be explored for a particular application from hundreds to a few cases. In the current paper, we use for this purpose the basic analysis of the correlation of similarity values of the two measures.

8.2 Importance of Input Data

The pipeline for computing similarity measure involves many steps. However, not all of the involved decisions have the same importance. Typically, the most important decision is the choice of input data. Different types of input data (item statements, metadata, item solutions, performance data) lead to significantly different aspects of similarity. Which of these aspects is the right one depends on a particular use case of similarity. Decision within the computation pipeline (e.g., the choice of specific features or normalization methods) are comparatively less important. There is a wide set of choices, and some of them can lead to significantly different results than others. Nevertheless, once we filter out outlying techniques, we usually find a set of closely correlated measures. Within them, the impact of the choice is not fundamental.

Since the choice of similarity measure depends on a particular application and its specific settings, we cannot give universal recommendations about the choice of input data. Based on our experience and the results of the analysis, we can, however, provide some basic guidelines for the choice. For applications of similarity measures in learner modeling, it is natural to use measures based on performance data and learners' solutions since these correspond to the actual behavior of learners. On the other hand, if we want to use similarity for guiding recommendations or hints, we may prefer to use measures based on item similarity, which leads to more understandable similarity results.

When using similarity measures for obtaining insight for teachers and authors of educational content, it is useful to highlight surprising, robust similarity results, specifically by focusing on the disagreement of measures. If similarity based on item statement is high, but the one based on learner performance is low (or the other way around), it may be useful for teachers to consider these items and use them during lectures. Developers of learning systems can use several types of measures for item management: detecting duplicated items using similarity based on item statements, detecting outlying items using similarity based on learner performance, or checking the correctness of metadata by looking for differences between similarities based on metadata and learner performance.

8.3 Recommendations for Similarity Computation

As mentioned above, the most important decision is the choice of input data. This choice differs depending on a domain and a particular application. For example, in programming the basic choice is to use item solutions—sample programs are typically available and carry the most important information about the item. For word problems in mathematics, it is natural to focus on item statements. For other kinds of items in mathematics (evaluating expressions like $3 \times (4 - 6)$ or $\frac{1}{2} + \frac{1}{3}$) it may be better to focus on performance data. For these items, the item statement and solution carry limited information, whereas performance data for these items can be quite quickly collected and used.

When using item statements and solutions, our explorations and experience suggest that a reasonable default pipeline is:

1) Compute feature matrix based on the data using the basic bag-of-words approach (computing number of occurrences of natural keywords).
2) Normalize the feature matrix, specifically using some variant of the TF-IDF transformation.
3) Compute item similarity using the Euclidean distance of vectors in the normalized feature matrix.

When using performance data, Pearson, Yule, and Cohen measures lead to better results than Ochiai, Sokal, and Jaccard measures. It is also beneficial to use the second step of item similarity. The exact choice of details does not seem to make a fundamental difference (e.g., Pearson versus Yule in the first step, the Euclidean distance versus Pearson correlation in the second step). The Pearson correlation coefficient is a good default choice since it provides quite robust results and is applicable in several settings and steps. It also has the pragmatic advantage of having fast, readily available implementation in nearly all computational environments, whereas measures like Yule may require additional implementation effort.

With performance data, it is essential to pay attention to the amount of available data. With small datasets, the computed similarity values are unstable and meaningless. It is hard to say in general what is a large enough dataset. Results of our analysis suggest that at least a few hundreds of answers per item are necessary. The specific number, however, depends on properties of a particular dataset. It is thus necessary to check the stability of similarity measures before we start using them.

Acknowledgments

The author thanks colleagues from the Adaptive Learning group (particularly Jiří Řihák, Tomáš Effenberger, Matěj Vaněk, Dominik Gmiterko, Vojtěch Sassmann, and Daniel Homola) for discussions and cooperation on previous research that provided a base for this overview.

References

[1] D. Churchill, “Towards a useful classification of learning objects,” *Educational Technology Research and Development*, vol. 55, no. 5, pp. 479–497, 2007.

[2] X. Ochoa and E. Duval, “Relevance ranking metrics for learning objects,” *IEEE Transactions on Learning Technologies*, vol. 1, no. 1, pp. 34–48, 2008.

[3] R. S. Baker, “Stupid tutoring systems, intelligent humans,” *International Journal of Artificial Intelligence in Education*, vol. 26, no. 2, pp. 600–614, 2016.

[4] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[5] J. Řihák and R. Pelánek, “Measuring similarity of educational items using data on learners’ performance,” in *Educational Data Mining*, 2017, pp. 16–23.

[6] R. Pelánek, T. Effenberger, M. Vaněk, V. Sassmann, and D. Gmiterko, “Measuring item similarity in introductory programming,” in *Proc. of Learning at Scale*. ACM, 2018.

[7] R. Pelánek, T. Effenberger, M. Vaněk, V. Sassmann, and D. Gmiterko, “Measuring item similarity in introductory programming: Python and robot programming case studies,” 2018, arXiv:1806.03240.

[8] T. Hofmann and J. M. Buhmann, “Pairwise data clustering by deterministic annealing,” *Ieee transactions on pattern analysis and machine intelligence*, vol. 19, no. 1, pp. 1–14, 1997.

[9] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann, “Optimal cluster preserving embedding of nonmetric proximity data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1540–1551, 2003.

[10] X. Ochoa and E. Duval, “Use of contextualized attention metadata for ranking and recommending learning objects,” in *Proceedings of the 1st international workshop on Contextualized attention metadata: collecting, managing and exploiting of rich usage information*. ACM, 2006, pp. 9–16.

[11] P. Lops, M. De Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[12] C. Desrosiers and G. Karypis, “A comprehensive survey of neighborhood-based recommendation methods,” in *Recommender systems handbook*. Springer, 2011, pp. 107–144.

[13] M. Deshpande and G. Karypis, “Item-based top-n recommendation algorithms,” *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[14] M. O’Connor and J. Herlocker, “Clustering items for collaborative filtering,” in *Proc. of the ACM SIGIR Workshop on Recommender Systems*, vol. 128. UC Berkeley, 1999.

[15] Y.-J. Park and A. Tuzhilin, “The long tail of recommender systems and how to leverage it,” in *Proc. of Recommender systems*. ACM, 2008, pp. 11–18.

[16] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering,” in *Proc. of Computer and Information Technology*, vol. 1, 2002.

[17] E. Spertus, M. Sahami, and O. Buyukkokten, “Evaluating similarity measures: a large-scale study in the orkut social network,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 678–684.

[18] Y. Koren and R. Bell, “Advances in collaborative filtering,” *Recommender Systems Handbook*, pp. 145–186, 2011.

[19] K. Tatsuoka, “Rule space: An approach for dealing with misconceptions based on item response theory,” *Journal of Educational Measurement*, vol. 20, no. 4, pp. 345–354, 1983.

[20] T. Barnes, “The q-matrix method: Mining student response data for knowledge,” in *Educational Data Mining Workshop*, 2005.

[21] M. C. Desmarais, “Mapping question items to skills with non-negative matrix factorization,” *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 2, pp. 30–36, 2012.

[22] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk, “Sparse factor analysis for learning and content analytics.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1959–2008, 2014.

[23] M. C. Desmarais, B. Beheshti, and P. Xu, “The refinement of a q-matrix: Assessing methods to validate tasks to skills mapping,” in *Proc. of Educational Data Mining*, 2014, pp. 308–311.

[24] R. Hosseini and P. Brusilovsky, “A study of concept-based similarity approaches for recommending program examples,” *New Review of Hypermedia and Multimedia*, pp. 1–28, 2017.

[25] P. Brusilovsky, J.-w. Ahn, T. Dumitriu, and M. Yudelson, “Adaptive knowledge-based visualization for accessing educational examples,” in *Proceedings of Information Visualization*. IEEE, 2006, pp. 142–150.

[26] S. Sahebi and P. Brusilovsky, “Student performance prediction by discovering inter-activity relations,” in *Educational Data Mining*, 2018, pp. 87–96.

[27] Q. Liu, Z. Huang, Z. Huang, C. Liu, E. Chen, Y. Su, and G. Hu, “Finding similar exercises in online education systems,” in *Proc. of Knowledge Discovery & Data Mining*. ACM, 2018.

[28] R. J. L. John, R. J. Passonneau, and T. S. McTavish, "Semantic similarity graphs of mathematics word problems: Can terminology detection help?" *International Educational Data Mining Society*, 2015.

[29] T. Käser, A. G. Busetto, B. Solenthaler, J. Kohn, M. von Aster, and M. Gross, "Cluster-based prediction of mathematical learning patterns," in *International Conference on Artificial Intelligence in Education*. Springer, 2013, pp. 389–399.

[30] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, 2008, pp. 49–56.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[32] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[33] D. A. Jackson, K. M. Somers, and H. H. Harvey, "Similarity coefficients: measures of co-occurrence and association or simply measures of occurrence?" *American Naturalist*, pp. 436–453, 1989.

[34] W.-H. Chen and D. Thissen, "Local dependence indexes for item pairs using item response theory," *Journal of Educational and Behavioral Statistics*, vol. 22, no. 3, pp. 265–289, 1997.

[35] W. H. Gomaa and A. A. Fahmy, "A survey of text similarity approaches," *International Journal of Computer Applications*, vol. 68, no. 13, 2013.

[36] C. Piech, M. Sahami, D. Koller, S. Cooper, and P. Blikstein, "Modeling how students learn to program," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 2012, pp. 153–160.

[37] P. Bille, "A survey on tree edit distance and related problems," *Theoretical computer science*, vol. 337, no. 1, pp. 217–239, 2005.

[38] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43–48, 2010.

[39] H. Finch, "Comparison of distance measures in cluster analysis with dichotomous data," *Journal of Data Science*, vol. 3, no. 1, pp. 85–100, 2005.

[40] S.-F. M. Liang and L.-W. Tzeng, "Assessing suitability of similarity coefficients in measuring human mental models," in *Network of Ergonomics Societies Conference*. IEEE, 2012, pp. 1–5.

[41] E. Şenyürek and H. Polat, "Effects of binary similarity measures on top-n recommendations," *Anadolu University Journal of Science and Technology – A Applied Sciences and Engineering*, vol. 14, no. 1, pp. 55–65, 2013.

[42] M. J. Warrens, "On association coefficients for 2× 2 tables and properties that do not depend on the marginal distributions," *Psychometrika*, vol. 73, no. 4, pp. 777–789, 2008.

[43] S. Klinkenberg, M. Straatemeier, and H. Van der Maas, "Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation," *Computers & Education*, vol. 57, no. 2, pp. 1813–1824, 2011.

[44] J. Rihák, "Use of time information in models behind adaptive system for building fluency in mathematics." in *Proc. of Educational Data Mining*, 2015.

[45] R. Pelánek and J. Řihák, "Properties and applications of wrong answers in online educational systems," in *Proc. of Educational Data Mining*, 2016.

[46] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[47] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 538–543.

[48] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[49] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in Neural Information Processing Systems*, 2015, pp. 505–513.

[50] K. O. McGraw and S. Wong, "A common language effect size statistic." *Psychological bulletin*, vol. 111, no. 2, p. 361, 1992.

[51] D. Gmiterko, "Techniques for measuring similarity of educational items," Bachelor thesis, Masaryk university, 2018.

[52] J. Papoušek, V. Stanislav, and R. Pelánek, "Evaluation of an adaptive practice system for learning geography facts," in *Proc. of Learning Analytics & Knowledge*, D. Gasevic, G. Lynch, S. Dawson, H. Drachsler, and C. P. Rosé, Eds. ACM, 2016, pp. 40–47.

Radek Pelánek received his Ph.D. degree in Computer Science from Masaryk University for his work on formal verification. Since 2010 his research interests focus on areas of educational data mining and learning analytics. Currently, he is the leader of the Adaptive Learning group at Masaryk University and is interested in both theoretical research in user modeling and practical development of adaptive educational systems.